



**Burroughs**

---

*Student Guide*

**Introduction to A Series and  
B 5000/B 6000/B 7000 Concepts  
Mark 3.6  
EP 4195**

**Preliminary Material  
April 1986**

**"The names used in this publication are not of individuals living or otherwise. Any similarity of likeness of the names used in this publication with the names of any individual living or otherwise is purely coincidental and not intentional."**

**Burroughs believes that the information described in this publication is accurate and reliable, and much care has been taken in its preparation. However, no responsibility, financial or otherwise, can be accepted for any consequences arising out of the use of this material, including loss of profit, indirect, special, or consequential damages. There are no warranties which extend beyond the program specification.**

**The customer should exercise care to assure that use of information in this publication will be in full compliance with laws, rules, and regulations of the jurisdiction with respect to which it is used.**

**The information contained herein is subject to change. Revisions may be issued from time to time to advise of changes and/or additions.**

**Correspondence regarding this document should be forwarded directly to Burroughs Corporation, Room 205, Education Development, 2611 Corporate West Drive, Lisle, IL 60532-3697.**

# A SERIES AND B 5000/B 6000/B 7000 CONCEPTS

## TABLE OF CONTENTS

### SECTION 1: Railroad Diagrams Overview

Objectives .....	1-2
Unit 1 Railroad Diagrams .....	1-3
Practice .....	1-6

### SECTION 2 : Families, Files and Disk Directories

Objectives .....	2-2
Unit 1 Families .....	2-3
Unit 2 File Names .....	2-10
Practice .....	2-15
Unit 3 Disk Directories .....	2-17
Unit 4 Family Substitution .....	2-24
Practice .....	2-28
Unit 5 File Attributes .....	2-30
Practice .....	2-36

### SECTION 3: InterPro Overview

Objectives .....	3-2
Unit 1 InterPro Introduction .....	3-3
Unit 2 MARC Usage .....	3-7

### SECTION 4: System Initialization

Objectives .....	4-2
Unit 1 System Initialization .....	4-3
Unit 2 Changing MCP Files .....	4-9
Practice .....	4-12

### SECTION 5: Hardware and I/O Overview

Objectives .....	5-2
Unit 1 Hardware .....	5-3

### SECTION 6: Data Communications Concepts

Objectives .....	6-2
Unit 1 Data Communications Hardware .....	6-3
Unit 2 Data Communications Software .....	6-10
Practice .....	6-17

# A SERIES AND B 5000/B 6000/B 7000 CONCEPTS

## SECTION 7: MCP and MCP/AS Overview

Objectives .....	7-2
Unit 1 MCP and MCP/AS Differences .....	7-3
Unit 2 MCP MCP/AS Functional Areas .....	7-10

## SECTION 8: Basic CANDE Commands

Objectives .....	8-2
Unit 1 CANDE Overview .....	8-3
Unit 2 CANDE Editing Commands .....	8-6
Unit 3 CANDE Control Commands .....	8-16
Practice .....	8-22
Unit 4 COMS Windows and Dialogs .....	8-23
Unit 5 Additional CANDE Commands .....	8-26

## SECTION 9: Stack Architecture Concepts

Objectives .....	9-2
Unit 1 Object Code File Layout .....	9-3
Unit 2 Memory Structures for Program Execution .....	9-7
Practice .....	9-14
Unit 3 Word Formats and Data Representation .....	9-15
Unit 4 Control Words .....	9-22
Practice .....	9-30
Unit 5 Display Registers .....	9-31
Unit 6 Top-Of-Stack Registers .....	9-41
Practice .....	9-53

## SECTION 10: SYSTEM/DUMPALL - Listing Disk Files

Objectives .....	10-2
Unit 1 SYSTEM/DUMPALL - Listing Disk Files .....	10-3
Practice .....	10-8

## SECTION 11: Libraries Overview

Objectives .....	11-2
Unit 1 Libraries Overview .....	11-3

## SECTION 12: Memory Management Overview

Objectives .....	12-2
Unit 1 Memory Management Overview .....	12-3



# A SERIES AND B 5000/B 6000/B 7000 CONCEPTS

## SECTION 13: Work Flow Management System

Objectives .....	13-2
Unit 1 Work Flow Management System Overview .....	13-3
Unit 2 Automatic Display Mode .....	13-15
Practice .....	13-20
Unit 3 Basic WFL Syntax and Statements .....	13-21
Unit 4 WFL File Equations and Compiling .....	13-26
Practice .....	13-32
Unit 5 LIBRARY/MAINTENANCE .....	13-33
Practice .....	13-40

## SECTION 14: Security Overview

Objectives .....	14-2
Unit 1 Security Overview .....	14-3

## SECTION 15: Software Products Overview

Objectives .....	15-2
Unit 1 Printing Subsystem Overview .....	15-3
Unit 2 Utilities Overview .....	15-9
Unit 3 DMS Overview .....	15-13
Unit 4 Other Software Products .....	15-20
Practice .....	15-25

## SECTION 16: Software Installation (Optional)

Objectives .....	16-2
Unit 1 Software Installation .....	16-3
Practice .....	16-8

## SECTION A 3: A 3 Hardware Overview

Objectives .....	A3-2
Unit 1 A 3 Hardware Overview .....	A3-3
Practice .....	A3-9

## SECTION A 9: A 9 Hardware Overview

Objectives .....	A9-2
Unit 1 A 9 Hardware Overview .....	A9-3
Practice .....	A9-10

# A SERIES AND B 5000/B 6000/B 7000 CONCEPTS

## SECTION A 10: A 10 Hardware Overview

Objectives .....	A10-2
Unit 1   A 10 Hardware Overview.....	A10-3
Practice .....	A10-10

## SECTION A 15: A 15 Hardware Overview

Objectives .....	A15-2
Unit 1   A 15 Hardware Overview.....	A15-3
Practice .....	A15-10

## SECTION B 5900: B 5900 Hardware Overview

Objectives .....	B5900-2
Unit 1   B 5900 Hardware Overview.....	B5900-3
Practice .....	B5900-7

## SECTION B 6900: B 6900 Hardware Overview

Objectives .....	B6900-2
Unit 1   B 6900 Hardware Overview.....	B6900-3
Practice .....	B6900-7

## SECTION B 7900: B 7900 Hardware Overview

Objectives .....	B7900-2
Unit 1   B 7900 Hardware Overview.....	B7900-3
Practice .....	B7900-12

## SECTION C: Compile Listings and Program Dumps

Sample ALGOL Payroll Program used in Section 9 .....	C-2
Compile Listing .....	C-3
Dump at Line 2650 .....	C-7
Dump at Line 1950 .....	C-8
Sample Payroll Program from Section 9, Converted to COBOL74 .....	C-11
Compile Listing .....	C-12
Dump at Line 3100 .....	C-22
Dump at Line 3700 .....	C-24
Sample Library Program in ALGOL .....	C-26
Compile Listing .....	C-27
Sample Library User Program In ALGOL .....	C-28
Compile Listing .....	C-29
Dump of User Program .....	C-30
Dump of Library .....	C-32

Sample Library Program in COBOL74 .....	C-33
Compile Listing .....	C-34
Sample Library User Program in COBOL74 .....	C-36
Compile Listing .....	C-37
Dump of User Program .....	C-39
ALGOL Program to Illustrate Simple Calculation .....	C-41
Compile Listing .....	C-42
Dump .....	C-44
COBOL74 Program to Illustrate Simple Calculation .....	C-45
Compile Listing .....	C-46
Dump .....	C-52
DUMPALL Listing of a Data File .....	C-53
COBOL74 Program to Show Dump with Disk File Buffers .....	C-55
Compile Listing .....	C-56
Dump .....	C-58
ALGOL Program with Procedures for Optional Dump Lab .....	C-64
Compile Listing .....	C-65
Dump .....	C-67

## LAB EXERCISES

MARC Lab .....	Lab-3
CANDE Lab .....	Lab-5
CANDE/DUMPALL Lab .....	Lab-9
Dump Lab (Optional) .....	Lab-11
WFL Lab .....	Lab-13

## BIBLIOGRAPHY

Resource List .....	Bib-2
---------------------	-------

# A SERIES AND B 5000/B 6000/B 7000 CONCEPTS

## Purpose of the Course

The purpose of this course is to introduce the basic concepts, terms, and software of the A Series and B 5000/B 6000/B 7000 as preparation for programming and managing the systems.

## Course Objectives

On successful completion of the course, the student will be able to:

- Identify the functional use of major hardware and software components of A Series , and B 5000/ B 6000/ B7000 systems
- Create, edit, and compile source files using CANDE
- Write basic WFL jobs using common file attributes
- Compile programs using WFL
- Use standard MARC menus to run programs and do inquiries
- Manage multiple CANDE sessions with COMS windows
- List data files using SYSTEM/DUMPALL
- Recognize data representation formats used in programs and files
- Identify major functions of the MCP
- Identify major characteristics of the stack architecture

## Intended Audience

Programmers, analysts, programming managers, and operations managers

## Prerequisites

- Six months computer programming experience

## Duration

- Five days

# A SERIES AND B 5000/B 6000/B 7000 CONCEPTS

## Introduction

The focus of this course is to introduce software concepts for the A Series/B 5000/B 6000/B 7000 Systems. The course begins and ends with software concepts. Some hardware concepts are covered to enhance the software information presented.

This course is designed in a modular format which allows the material to be presented in different sequences if necessary.

The topics marked as **optional** may be omitted at the discretion of the instructor, depending on the interests and needs of the students.

The student guide is to be used to supplement the class presentations and is not to be used as text book.

The notation **A Series Systems** used in the text of the student guide includes the A Series/B 5000/B 6000/B 7000 Systems, unless otherwise specified.

Throughout the student guide ODT commands have been included when applicable. The course does not provide details on ODT commands, but emphasizes communicating with the system through a programmer's terminal. For detailed information on ODT commands, refer to the A Series ODT Reference Manual.

Each unit has resource manuals listed for your further research following the completion of the class. Some of these manuals may be available for your use during class. The bibliography at the end of the Student Guide lists the titles and form numbers for all the resource manuals.

## A SERIES AND B 5000/B 6000/B 7000 CONCEPTS

This page left blank for formatting.

**SECTION 1**  
**RAILROAD DIAGRAMS OVERVIEW**

# A SERIES AND B 5000/B 6000/B 7000 CONCEPTS RAILROAD DIAGRAMMS OVERVIEW

## *INTRODUCTION*

### **Section Objective**

Read railroad diagrams.

### **Purpose**

In order to read A Series documentation, you must be able to read railroad diagrams.

### **Unit Objectives**

Read railroad diagrams.



# **A SERIES AND B 5000/B 6000/B 7000 CONCEPTS RAILROAD DIAGRAMS OVERVIEW**

## **UNIT 1**

### **RAILROAD DIAGRAMS**

#### **Objective**

Read railroad diagrams.

#### **Purpose**

You must know how to read railroad diagrams to use the A Series documentation.

#### **Resources**

A Series Systems An Introduction, Section 7 - Where To Find More Information

A 9 System Software Installation Guide, Appendix B - Railroad Diagram Description

A Series CANDE Reference Manual, Section - Understanding Railroad Diagrams

A Series System Software Site Management Reference Manual, Section - Understanding Railroad  
Diagrams

# A SERIES AND B 5000/B 6000/B 7000 CONCEPTS RAILROAD DIAGRAMS

## What is a Railroad Diagram?

Burroughs convention for software documentation is the use of railroad diagrams to show syntax.

A **railroad diagram** is a graphic representation of the syntax of a statement or command.

A railroad diagram:

Consists of horizontal and vertical lines, keywords, symbols, upper and lower case words, and special characters.

Indicates which items are required or optional.

Shows the order in which the items must appear in the statement or command to be accepted by the system.

Tells how often items can be repeated.

Indicates the required punctuation.

Valid statements and commands are constructed by following the rules for reading railroad diagrams.

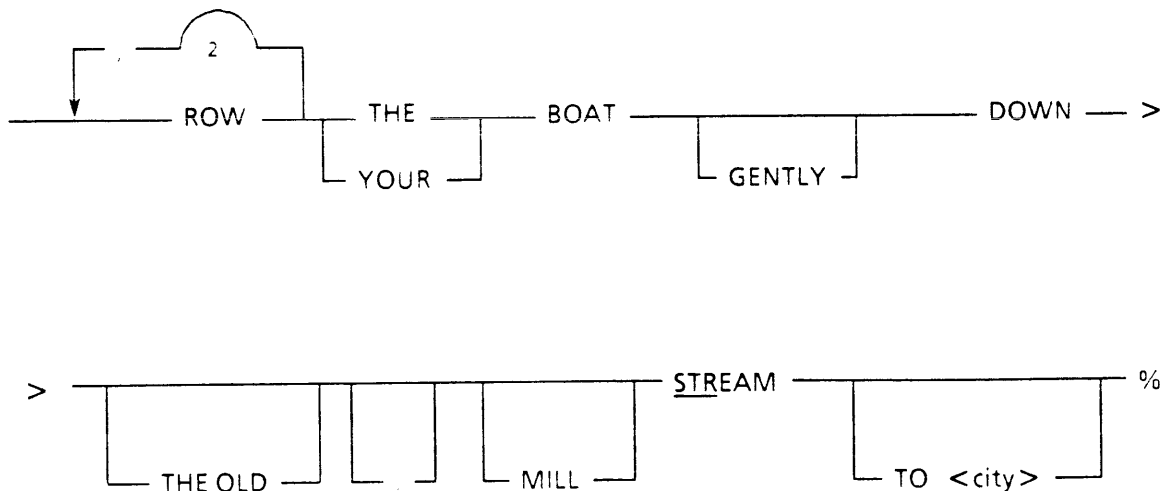


Figure 1-1 Sample Railroad Diagram

## A SERIES AND B 5000/B 6000/B 7000 CONCEPTS RAILROAD DIAGRAMMS

Some valid statements from Figure 1-1 are:

ROW THE BOAT DOWN STR

ROW, ROW, ROW YOUR BOAT GENTLY DOWN THE OLD MILL STREAM

ROW, ROW, ROW THE BOAT GENTLY DOWN THE OLD, MILL STREAM TO CHICAGO

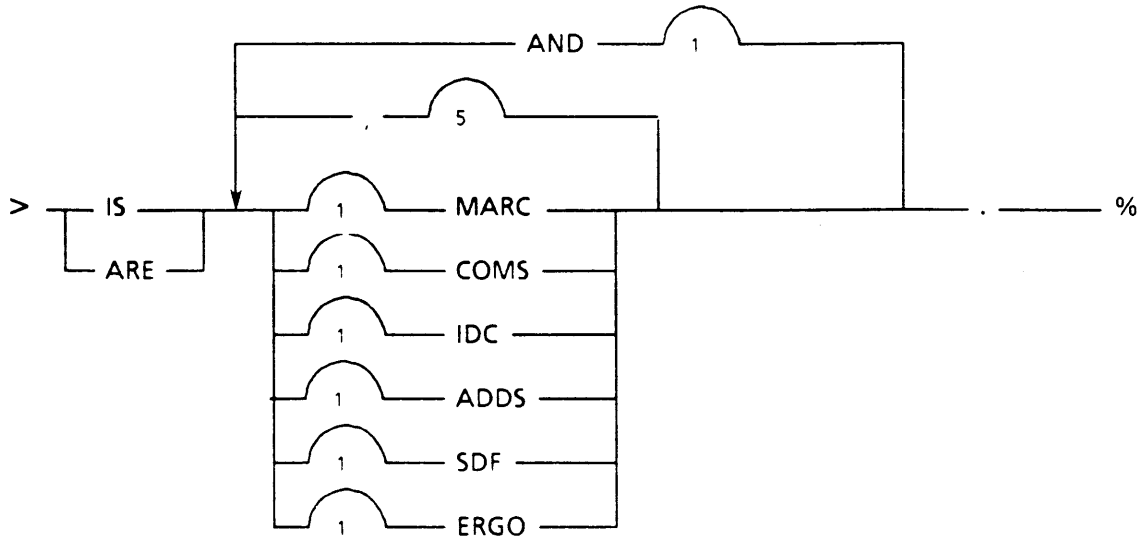
### *Railroad Diagram Rules:*

1. Read left to right except where indicated by arrows pointing right to left.
2. A loop is an item or group of items that can be repeated.
3. A bridge shows the maximum number of times you can take this path.
4. If diagram will not fit on one line, a right arrow ( > ) appears at the end of the first line. Another right arrow will appear at the beginning of the continuation line.
5. The end of the diagram is denoted by a vertical bar (|) or a percent sign (%).
6. A vertical bar means the command can be followed by a semicolon and another command.
7. A percent sign indicates that nothing else is to follow.
8. Upper case words must be spelled as they appear. You may use the acceptable abbreviation, which is underlined.
9. At least one blank must appear between words.
10. Blanks are optional around special characters.
11. Brackets with lower case words inside indicate that this a user-supplied variable.
12. Brackets are omitted.
13. A bridge with an integer only indicates the number of times that path can be traveled.
14. A bridge with an integer \* indicates that this path must be traveled that number of times.

# A SERIES AND B 5000/B 6000/B 7000 CONCEPTS RAILROAD DIAGRAMS

## Practice

\_\_\_\_\_ <number> \_\_\_\_\_ OF THE INTERACTIVEPRODUCTIVITY PRODUCTS \_\_\_\_\_ >



<number>

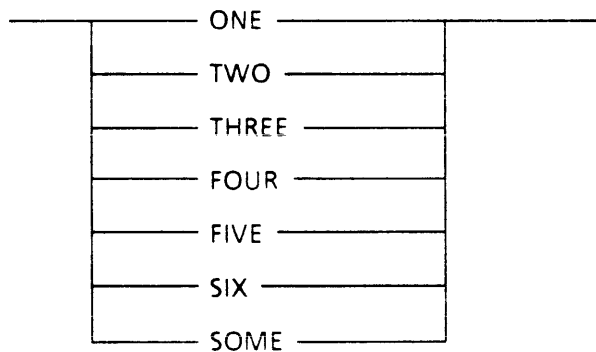


Figure 1-2 Practice Railroad Diagram

Using the railroad diagram Figure 1-2, indicate if the following statements are Valid or Invalid.

- \_\_\_\_\_ 1. ONE OF THE INTERPRO PRODUCTS IS MARC.
- \_\_\_\_\_ 2. SOME OF THE INTERPRO PRODUCTS ARE COMS, MARC AND IDC.
- \_\_\_\_\_ 3. ONE OF THE INTERPRO PRODUCTS ARE ERGO.
- \_\_\_\_\_ 4. FIVE OF THE INTERPRO PRODUCTS ARE MARC, COMS, IDC, SDF ERGO.
- \_\_\_\_\_ 5. THREE OF THE INTERPRO PRODUCTS ARE MARC, ADDS AND SDF.
- \_\_\_\_\_ 6. ALL OF THE INTERPRO PRODUCTS ARE MARC, COMS, IDC, ADDS, SDF AND ERGO.

**SECTION 2**  
**FAMILIES, FILES AND DISK**  
**DIRECTORIES**

## ***INTRODUCTION***

### **Section Objective**

Access Files on the system.

### **Purpose**

To access disk files, you must know how families, disk directories and file attributes function.

### **Unit Objectives**

Recognize a Family and related terms.

Identify Mirror Disk.

Specify the file naming convention for the system.

Identify the Disk Directories used by the system.

Write Family Substitution statements.

Recognize some common file attributes.

# A SERIES AND B 5000/B 6000/B 7000 CONCEPTS FAMILIES, FILES AND DISK DIRECTORIES

## **UNIT 1**

### **FAMILIES**

#### **Objective**

Recognize a Family and related terms.

Identify Mirror Disk.

#### **Purpose**

In order to access disk files, you must be aware of how families and mirror disks function.

#### **Resources**

A Series Systems An Introduction, Section 3 - A User's View of System Functions

A Series Disk Subsystem Software Overview, Section 2 - Disk Subsystem Concepts  
Section 7 - Mirrored Disk

A Series I/O Subsystem Reference Manual, Section 1 - Introduction  
Section 3 - Device Dependencies  
Section - Format of External File Name

# A SERIES AND B 5000/B 6000/B 7000 CONCEPTS FAMILIES

## What Is A Family?

A **Family** is a single disk device or a collection of disk devices which are logically grouped together, have the same name and a common file directory (list of all files).

The system treats these physical disks as a single entity and can spread files destined for that family across all the members of the family. A family can consist of different types of disk hardware.

Most systems will have many families and the family names usually correspond to some function performed or to a group of users.

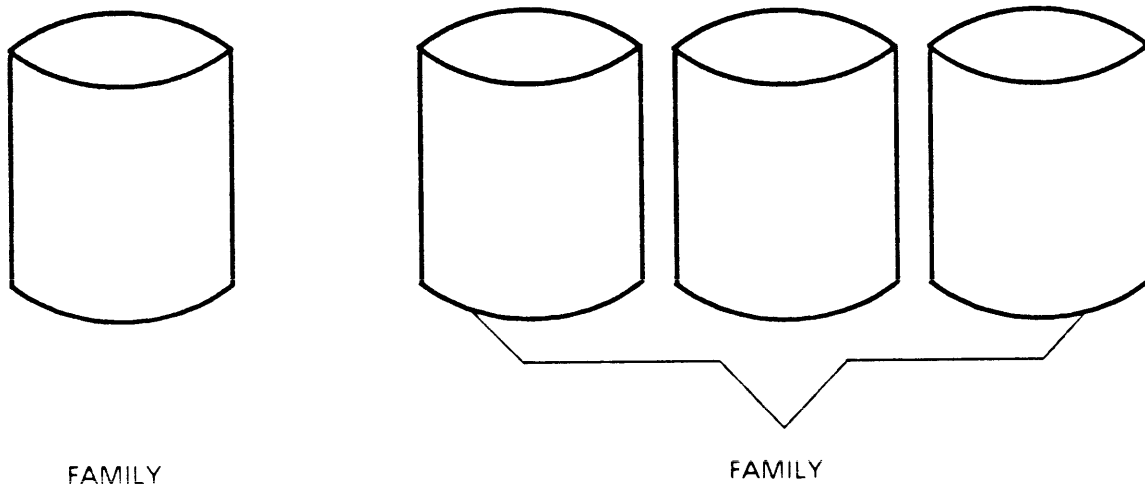


Figure 2-1 Families

Since a family can have many physical packs, all with the same name, each physical pack is assigned a unique number known as the **Family index**. To identify a particular pack in the family, you would use the family name and the family index assigned to that pack.



## A SERIES AND B 5000/B 6000/B 7000 CONCEPTS FAMILIES

Each family must have a single **Base Pack**. The base pack will always have a family index of 1. It will also contain the common file directory which is called the **Flat Directory**. The flat directory contains the file headers (contains file names, disk addresses, file characteristics) for every file that is stored on any member of the family. The flat directory itself is a file and is named SYSTEMDIRECTORY/001.

All other packs that make up the family are known as **Continuation Packs**. Continuation packs are optional. The maximum number of continuation packs that a family can have is 254. The **family indexes for continuation packs** are 2 through 255.

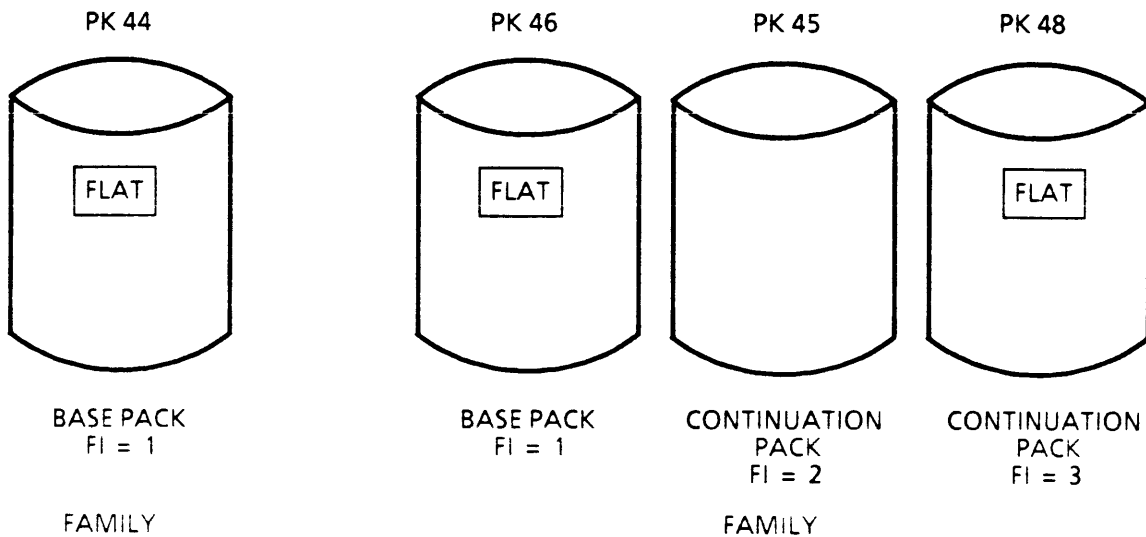


Figure 2-2 Continuation Packs

Since access to all files requires the flat directory, the flat directory located on the base pack can be duplicated and placed on the continuation packs for that family. The system will allow a maximum of 3 flat directories (1 on base pack, 1 on each of 2 continuation packs). Duplicate flat directories are created using the ODT command **DD** (Directory Duplicate).

The **DD** command requires the family name and family index where the duplicate flat directory is to be stored. Its file name will be SYSTEMDIRECTORY/<3 digit familyindex>. The **DD** command is also used to remove duplicate flat directories from a family.

# A SERIES AND B 5000/B 6000/B 7000 CONCEPTS FAMILIES

## *Mirror Disk*

The Mirror Disk feature provides you with the ability to have from 2 to 4 disks maintained as exact copies of each other (Mirrored Set).

Mirror Disk is available for systems that have Data Link Processors (DLP) hardware. For more information on DLPs, see Section 5 - Hardware Overview.

### Mirror Disk Features

#### Increase System Availability

Backup and audit requirements should be re-evaluated since mirrored sets reduce the possibility of data loss. Processing time previously used for backup and auditing is now available for some other type of processing.

#### Improves Data Integrity

If one copy of the data in a mirrored set is destroyed or has irrecoverable errors, another set will allow normal functions to proceed.

#### Possible Improvement of I/O Throughput

Improvements in I/O throughput will vary depending on the individual characteristics of the installation.

If your system has a high ratio of reads versus writes, your throughput could be improved. The MCP will distribute reads equally to all mirrored sets. All writes must be issued to all members of the mirrored set.

Disk mirroring is completely transparent to the application program. Applications will continue processing normally if an error on one mirror disk occurs. Special programming is not required in the application to handle mirrored sets. Applications issue only one read or write as they are currently doing, and the MCP handles the control of the other members of the mirrored set.

Two tables are maintained by the MCP.

**OWL** - Outstanding Write List keeps track of writes in process to mirrored sets. The OWL is kept in memory.

**MIT** - Mirror Information Table contains information concerning all mirrored sets on the system.

Disk Mirroring is an optional function of the MCP controlled by the ODT command **OP** (Options) and the **MIRRORING** option. Once the mirroring option has been turned on, mirrored sets are established by the ODT command **MIRROR**. The **MIRROR** command allows for creation/deletion and maintenance of mirrored sets.

# A SERIES AND B 5000/B 6000/B 7000 CONCEPTS FAMILIES

## *Family Names*

An **Identifier** is 1 to 17 alphanumeric characters. The recommended characters for an identifier are 0 thru 9 and A thru Z.

A **Family Name** is an identifier.

Many installations establish 2 special purpose families.

### **DISK**

The family named DISK is sometimes called the "System Disk". This family will normally contain the Master Control Program (MCP). In addition to the MCP itself, many files used by the MCP and a special directory called SYSTEM/ACCESS are located here.

### **PACK**

The family named PACK is sometimes called the "System Resource Pack" because the files contained here are usually files that are accessed by all users of the system. Some of these files are compilers, utilities and printer backup files.

This pack family has become less common at installations than in the past. If family PACK is not present, the files that would have been located on PACK are divided between DISK and other families on the system.

Your system will have **other families** for file storage.

The files stored on other families usually are the user data bases, data files and programs (both source and object code). The family name must be an identifier and should have some meaning for your installation. The number of physical disk devices that make up your families will differ depending on the storage needs for different departments or functions. Your site should establish some type of family naming convention at the time the system is installed.

## A SERIES AND B 5000/B 6000/B 7000 CONCEPTS FAMILIES

You can use the ODT command PER (Peripheral Status) to examine the family name, serial number, and family index for disk devices. To use this ODT command and certain other ODT commands, you are required to enter a unit mnemonic for the disk instead of the family name. **PK** is the disk drive mnemonic. All hardware types have mnemonics which are unique. Some examples of mnemonics are:

PK	disk drives
MT	magnetic tape drives
LP	line printers

Example PER command:

PER PK

Sample Response:

```
----PK STATUS---  
44*B [000100] (MCP) #1 DISK (1)  
45*C [002000:001000:46] #2 EDUCATION (24)  
46*B [001000] #1 EDUCATION (12)  
48*C [002005:001000:46] #3 EDUCATION (10)
```

The response from PER PK includes a unit number. A **Unit Number** is a unique number assigned to a particular peripheral device, used to identify the device.

To distinguish a particular disk drive on your system using mnemonics, you include a unit number that represents that device.

Examples: PK 46, PK 48.

Certain ODT commands require a mnemonic and a unit number, such as **OL** (Display Label and Paths).

Example: OL PK 44

# A SERIES AND B 5000/B 6000/B 7000 CONCEPTS FAMILIES

## Rows/Areas

A **Row** or **Area** is a contiguous space on disk allocated for a portion of a file.

When a file is created on a family, space for the file is allocated one row at a time. The MCP determines which member of the family the row is placed on. The MCP uses a rotational manner to allocate the rows of files. A single file can be spread across all members of the family. The **SINGLEUNIT** option can be selected for a file, to force the MCP to place the entire file on one member of the family.

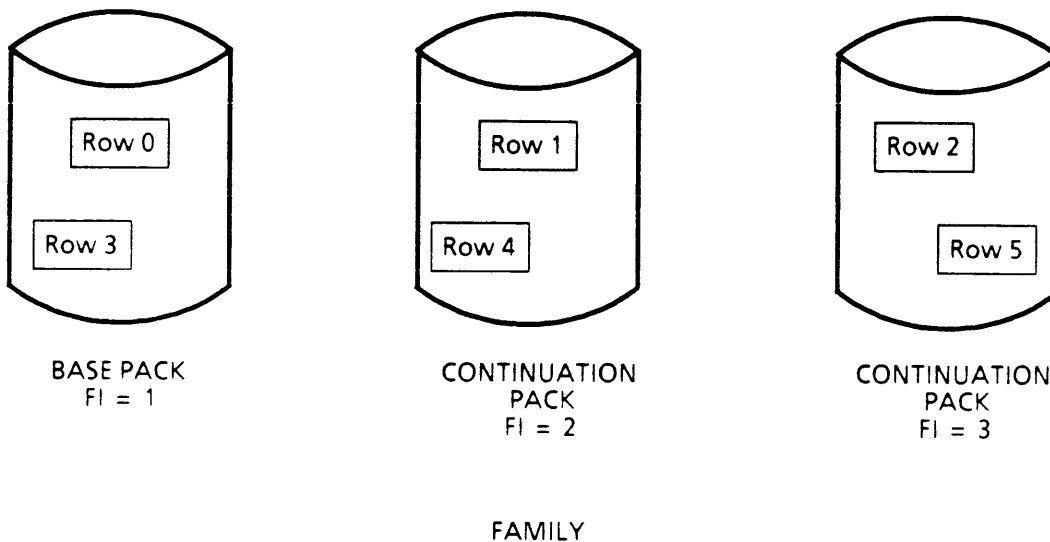


Figure 2-3 Row Allocation

If a disk becomes checkerboarded, the ODT command **SQUASH** can be used to correct the problem.

# A SERIES AND B 5000/B 6000/B 7000 CONCEPTS FAMILIES, FILES AND DISK DIRECTORIES

## *UNIT 2*

### *FILE NAMES*

#### **Objective**

Specify the file naming convention for the system.

#### **Purpose**

In order to access disk files, you must know how files are named.

#### **Resources**

A Series Systems An Introduction, Section 3 - A User's View of System Functions

A Series I/O Subsystem Reference Manual, Section 8 - Format of External File Names

# A SERIES AND B 5000/B 6000/B 7000 CONCEPTS FILES

## File Naming Convention

Each file which is to be stored on media is required to have a name so that it can be identified.

A **File Name** consists of 1 to 12 identifiers. Each identifier is separated by a slash (/). If the file name is to have a usercode associated with it, the system will allow the usercode to be a thirteenth identifier.

Examples:           A/B  
  
                      A/B/C/D/E/F/G/H/I/J/K/L

As for family names, an identifier is 1 to 17 alphanumeric characters. File names will allow special alphanumeric characters which must be enclosed in quotes, such as ".".

Each of the 12 identifiers must start with a letter or a digit.

The rest of the characters that make up the identifier, can be letters (A-Z), digits (0-9), hyphen or underscore without using quotes. Any other characters must have quotes surrounding them.

Examples:  
  
                      A"/.B"/C  
  
                      1A/2\_\_B/C3

The thirteenth node of the file name is for the usercode specification. The usercode must be placed before all of the other nodes of the file name, enclosed in parentheses, and not followed by a slash. The usercode node is either a valid usercode for the system or an \* (**Non-Usercoded** files).

Examples:  
  
                      (USER1)A"/.B"/C  
  
                      (ACCT)1A/2\_\_B/C3  
  
                      \*A/B/C  
  
                      \*SYSTEM/DUMPALL  
  
                      (PAYROLL)EMPLOYEE/HISTORY

## A SERIES AND B 5000/B 6000/B 7000 CONCEPTS FILES

The use of Usercodes as a node of the file name allows the system to:

Provide for segregation of files by user.

Files can be created with the same name by different users. The usercode will make the file name unique and allow the system to access the correct file per user.

Establish security boundaries.

Usercodes are associated with a level of system security. Some usercodes are **Privileged** and others are **Non-Privileged**.

**Non-Privileged** usercodes can access only files that the usercode owns and files that are non-usercoded. This is the system default.

**Privileged** usercodes can have access to all users' files. This includes both usercoded and non-usercoded files.

Establish the user identity.

To segregate files by user and enforce security restrictions, the system must be able to identify its users. All users must enter a valid usercode before doing any type of production. The valid usercodes are stored in a disk file named **SYSTEM/USERDATAFILE** which is maintained by the program called **SYSTEM/MAKEUSER**.

The ODT is a privileged user and is not required to log on (enter a valid usercode) before a command can be processed. To access usercoded files, the ODT is required to supply the usercode as part of the file name.

All files that are stored on a disk pack will have the family name where they are stored associated with the file name. The family name is appended to the file name following the word **ON**. When the file name includes the word **ON** and the family name, this name is now called a **File Title**.

Examples:

(STUDENT)A/"B"/C ON EDUCATION

(USER2)1A/2\_\_B/C3 ON PAYROLL

\*A/B/C ON DISK

\*SYSTEM/DUMPALL ON DISK

The system supplies a family name of DISK unless it has been explicitly set to another value.



# A SERIES AND B 5000/B 6000/B 7000 CONCEPTS FILES

## *Nodes of a File Name*

Each identifier that is part of the file name is also called a **Node**. The nodes of a file name also have special titles.

The rightmost node is called the **File Header** or **File Identifier**.

When the usercode is present in the file name, it is the leftmost node.

The other nodes are called **Directories**. Directories are further subdivided.

The first node of the directories is called **Directory**.

The other nodes of the directory nodes are called **Subdirectories**.

## *Tree Structure*

Since files can be grouped by similarities in the beginning parts of their names, a tree structure can be established.

The top of the tree is the family name. The next level down is the usercode. The next lower level is the directory, followed by the subdirectories until you reach the file identifier.

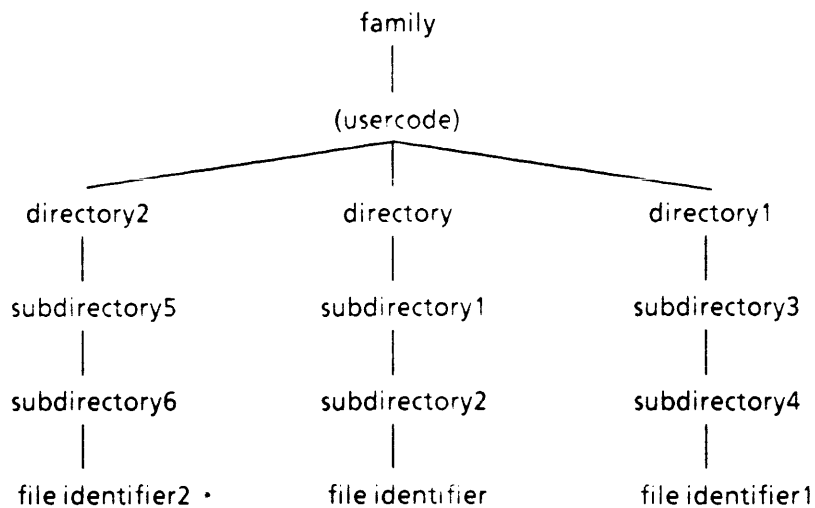


Figure 2-4 Tree Structure

## A SERIES AND B 5000/B 6000/B 7000 CONCEPTS FILES

Groups of files whose names begin with common nodes can be referred to by using the equal sign (=). The equal sign (=) must appear after a slash (/) or after the parenthesis of the usercode.

Examples:

(STUDENT)=

(USER2)EMPLOYEES/=

(ACCTS)PAYABLE/JANUARY/= ON ACCTPACK

The ODT command PD (Print Directory) allows you to inquire as to the existence of a file or files. Your entry depends on the tree structure.

Examples:

PD A/=

PD (USER1)=

PD (ACCTS)PAYABLE/JANUARY/= ON ACCTPACK

PD (STUDENT)= 2

PD (STUDENT)LAB/TEST/DATA

**A SERIES AND B 5000/B 6000/B 7000 CONCEPTS  
FILES**

**Practice**

Part A: Match the terms on the left with the descriptions on the right.

- |           |                |    |                                                                                               |
|-----------|----------------|----|-----------------------------------------------------------------------------------------------|
| _____ 1.  | FAMILY         | a. | Composed of 1 to 17 alphanumeric characters.                                                  |
| _____ 2.  | DIRECTORY      | b. | Composed of 1 to 13 identifiers.                                                              |
| _____ 3.  | FLAT DIRECTORY | c. | An identifier used to establish user identity and can be either privileged or non-privileged. |
| _____ 4.  | USERCODE       | d. | An exact copy(ies) of a family.                                                               |
| _____ 5.  | FILE NAME      | e. | A file that contains the file headers.                                                        |
| _____ 6.  | PACK           | f. | A unique number assigned to each member of a family.                                          |
| _____ 7.  | IDENTIFIER     | g. | Node of a file name.                                                                          |
| _____ 8.  | FAMILYINDEX    | h. | Contains the Master Control Program and can be called the System Disk.                        |
| _____ 9.  | MIRROR DISK    | i. | A family normally known as the System Resource Pack.                                          |
| _____ 10. | DISK           | j. | A collection of disk devices with a common name and file directory.                           |

Part B: See next page.

# A SERIES AND B 5000/B 6000/B 7000 CONCEPTS FILES

Part B: Given the information below, draw the logical tree-structured diagram that represents the files listed, by placing the correct identifier in the space provided.

- a. The usercode is USER1.
- b. The family is named EDUCATION.
- c. The files are:
  - A/C/F/L
  - A/C/F/K
  - A/C/E/J
  - A/C/E/I
  - A/B/D/H
  - A/B/D/G

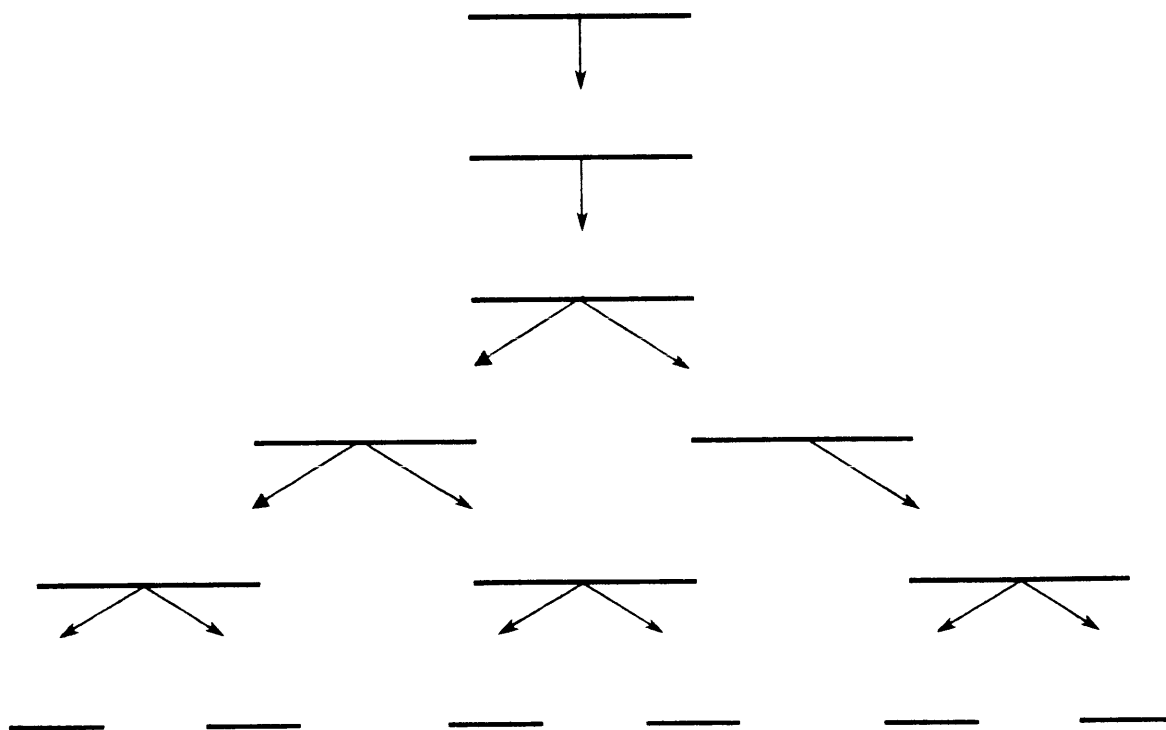


Figure 2-5 Tree Structure

**A SERIES AND B 5000/B 6000/B 7000 CONCEPTS  
FAMILIES, FILES AND DISK DIRECTORIES**

***UNIT 3***

***DISK DIRECTORIES***

**Objective**

Identify the Disk Directories used by the system.

**Purpose**

You should be aware of what directories are involved and what the role of each directory is when you access a disk file.

**Resources**

A Series Systems An Introduction, Section 3 - A User's View of System Functions

A Series Disk Subsystem Software Overview, Section 2 - Disk Subsystem Concepts

# A SERIES AND B 5000/B 6000/B 7000 CONCEPTS DISK DIRECTORIES

## Directories

A directory is a list of files organized into a hierarchy according to similarities in the first parts of the names. Directories are structures maintained by the MCP that are used to locate files.

### SYSTEM/ACCESS Directory

The system has a directory called SYSTEM/ACCESS/<family index>, located by default on the family DISK. This directory is maintained for the whole system and is divided into two parts.

#### PAST (Pack Access Structure)

The PAST contains an entry for every family that is active on the system. Each entry contains the family name and a pointer into the FAST. The pointer indicates where the entry for the family's files are stored in the FAST.

#### FAST (File Access Structure)

The FAST contains pointers to each disk's file headers, sorted by tree structure order. The FAST contains an entry for each file on the system. The entry is a pointer into the Flat directory (pointer to file header).

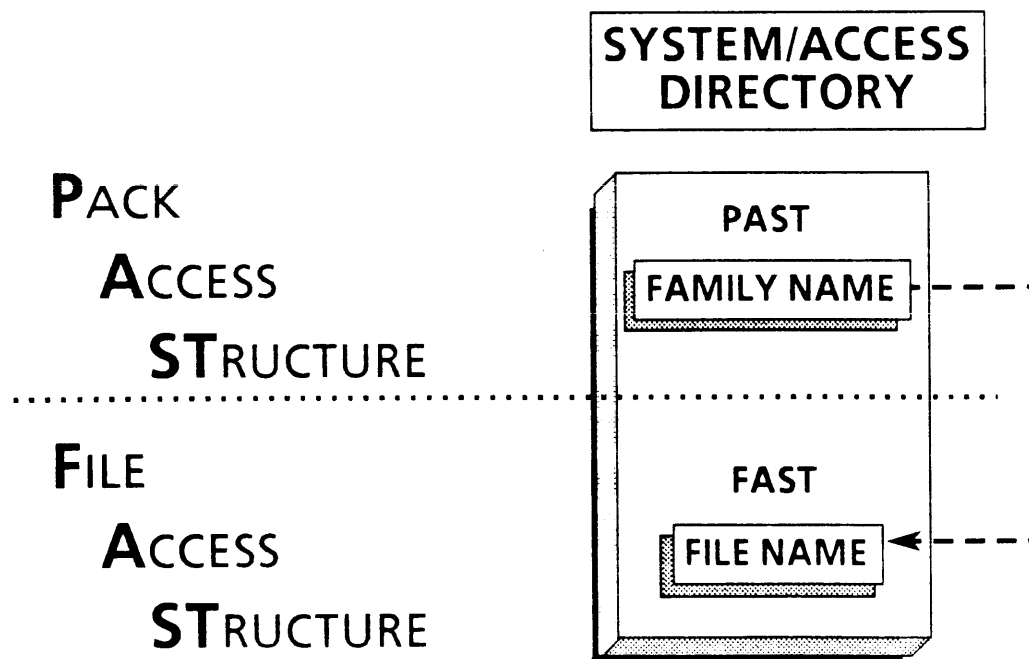


Figure 2-6 SYSTEM/ACCESS Directory

**A SERIES AND B 5000/B 6000/B 7000 CONCEPTS  
DISK DIRECTORIES**

*Flat Directory*

The Flat directory is the local directory for the family and is unsorted. It is located on the family's base pack. It contains a file header for every file that is stored on some member of the family. It received its name because the FAST points directly to the desired file header.

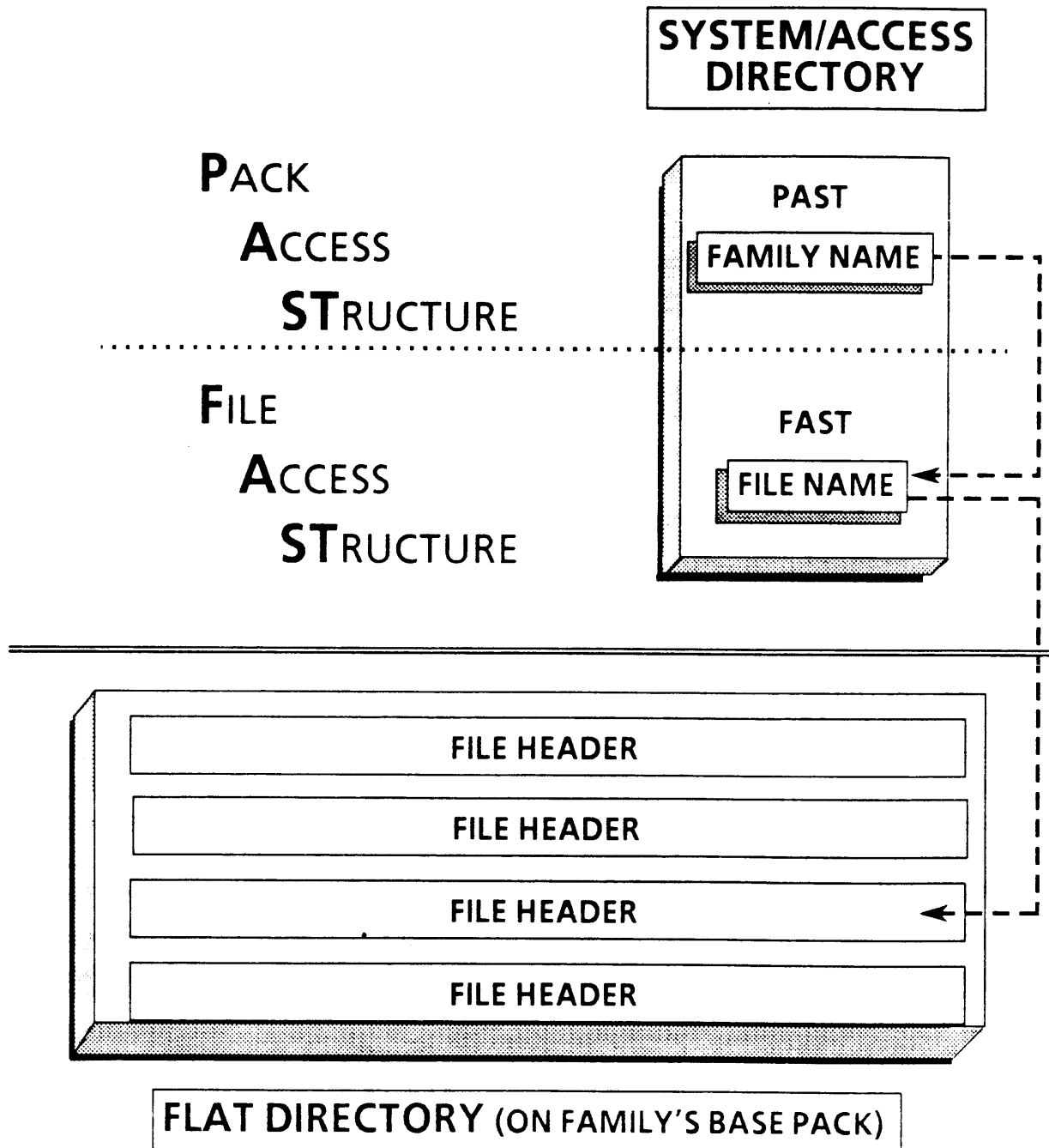


Figure 2-7 SYSTEM/ACCESS and Flat Directories

# A SERIES AND B 5000/B 6000/B 7000 CONCEPTS DISK DIRECTORIES

## File Headers

A File Header is a data structure that contains information about a disk file. File headers are part of the Flat directory. The file header contains the disk addresses of the rows of the file. It also contains information about the characteristics of the file.

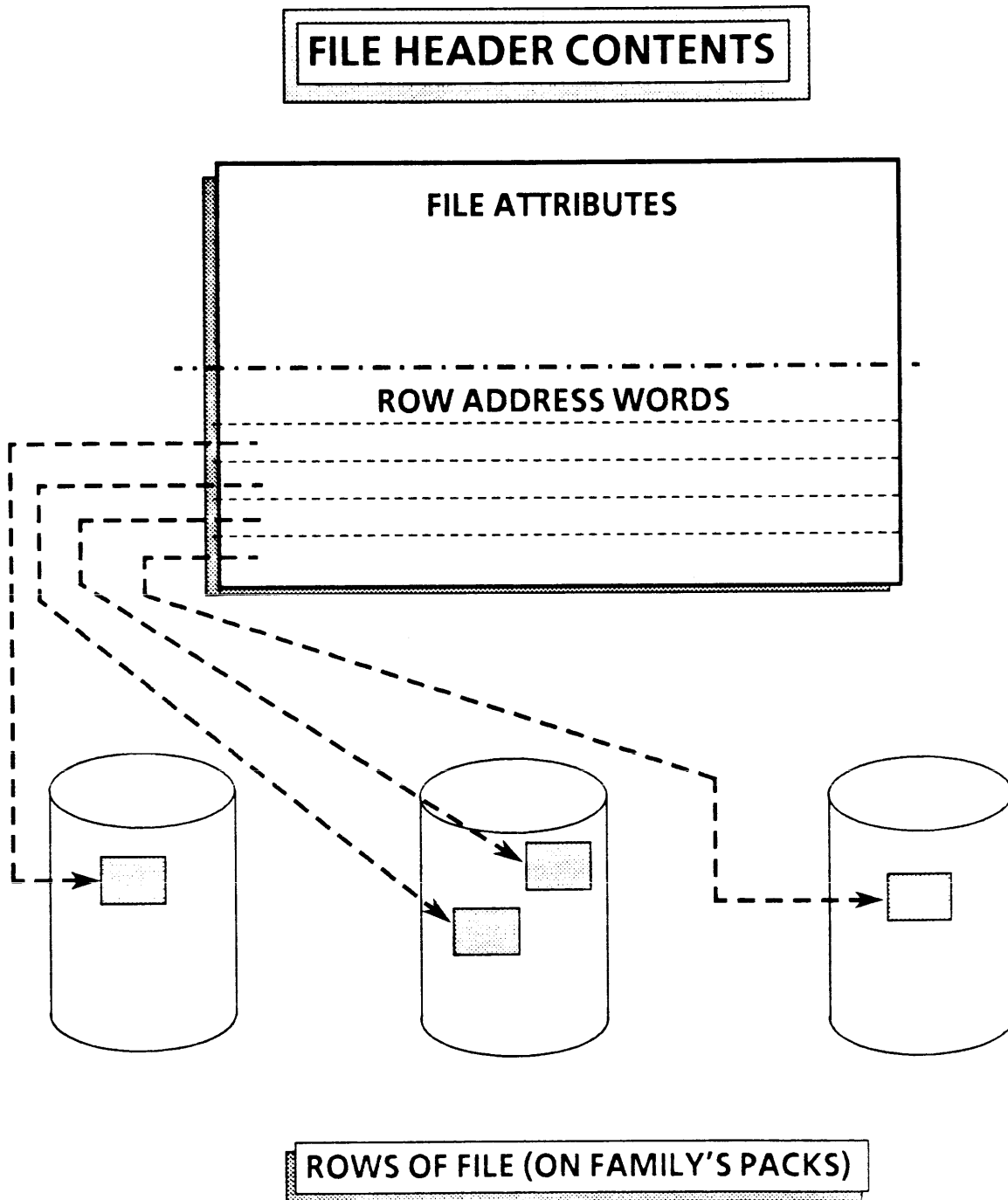


Figure 2-8 File Header Contents



## **A SERIES AND B 5000/B 6000/B 7000 CONCEPTS DISK DIRECTORIES**

### *Process of Accessing Disk Files*

Below are the steps followed by the MCP when a file is requested by file title (file name and family name). Refer to Figure 2-9 which follows.

1. The MCP uses the PAST to locate the proper family entry in the FAST.
2. The MCP reads the file entries for the family in the FAST and locates the pointer to the disk file header for the file with that file name.
3. The MCP reads the disk file header in the Flat directory.
4. The MCP obtains from the header the physical disk addresses of the area or areas that contain the disk file.
5. The MCP accesses the file's data at those physical addresses.

The PAST is not accessed each time a file is to be located. The MCP reads the PAST entries into a table in main memory when the system is initialized. When the PAST is updated, the MCP table is updated also. When a file is to be located, the MCP reads the table in memory to get the pointer to the FAST.

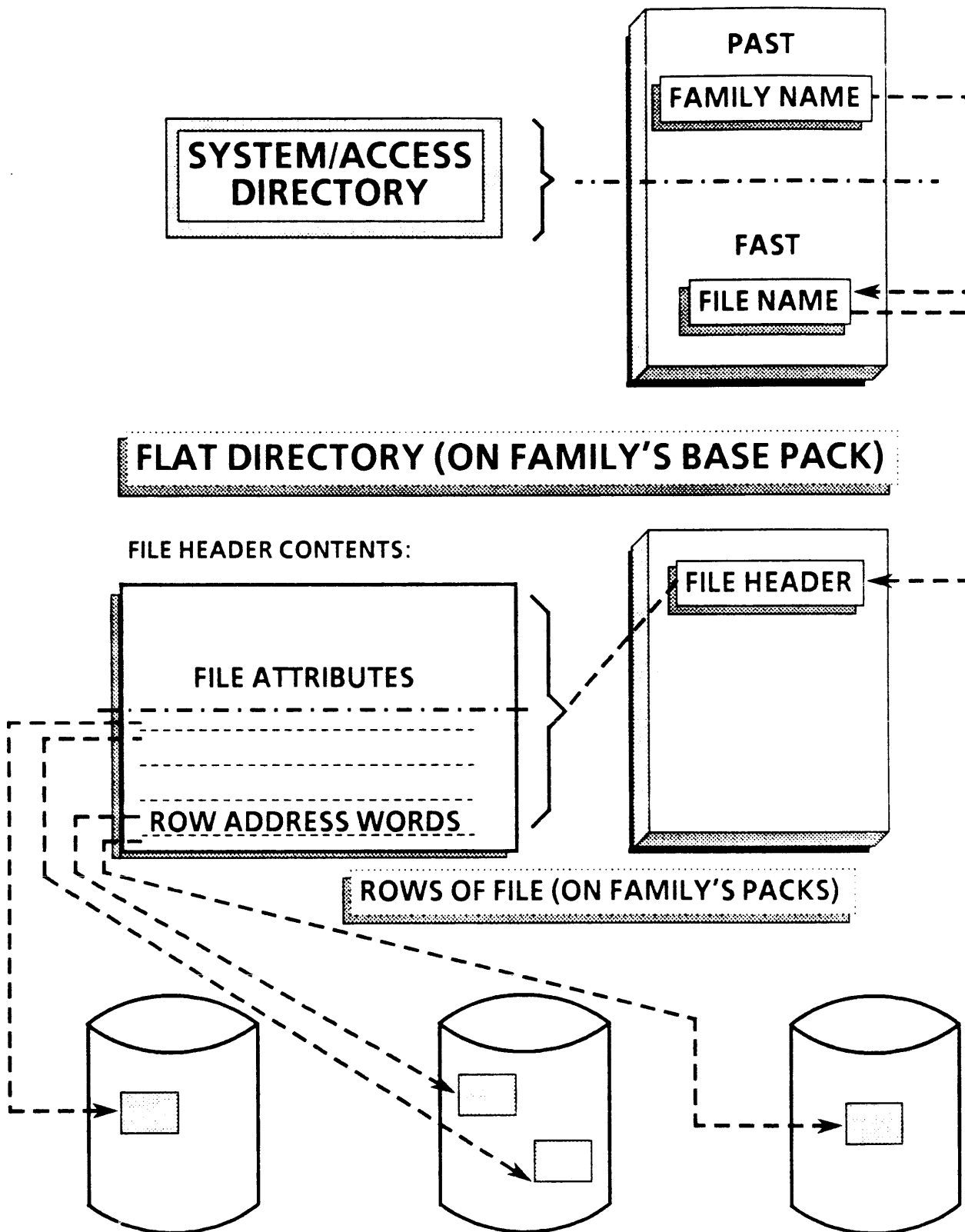


Figure 2-9 Directories Flow

## A SERIES AND B 5000/B 6000/B 7000 CONCEPTS DISK DIRECTORIES

### *Directory Errors*

Since the FAST is logically organized as a tree structure, it is totally unusable if an error is encountered at the high level of the structure. The FAST can be rebuilt by using the Flat directories for each family. A rebuild of the FAST will also be done if the ODT command RB (ReBuild Access) is entered.

### **LAST**

**LAST** is the Local Access Structure Table. It is a copy of the FAST and is located on the base pack of the family only.

The LAST eliminates the need for a rebuild of the FAST when a pack is brought on-line. The FAST is restored by using the LAST. If the LAST is not present or an error occurs when attempting to copy the LAST, the MCP will cause a rebuild of the FAST to occur.

The LAST is not always present on the base pack.

The LAST is written to the base pack of a family when 1 of the ODT commands CLOSE, FREE or PO (Power Off) is requested and performed by the MCP. If the pack is manually powered off, the MCP will not attempt to write a LAST to the family.

If an error occurs during the MCP's attempt to write the LAST to the base pack, or the MCP determines that there is insufficient space on the base pack to place the LAST, the MCP will discontinue the operation.

If the system is using the MCP cataloging feature, a LAST will not be generated.

If the pack is mirrored, no LAST will be generated when the pack is removed from the system..

The SYSTEM/ACCESS directory is created the first time the system is initialized. Each time a family is added to the system, the PAST and FAST are updated using the LAST or a rebuild operation.

**A SERIES AND B 5000/B 6000/B 7000 CONCEPTS  
FAMILIES, FILES AND DISK DIRECTORIES**

**UNIT 4**

**FAMILY SUBSTITUTION**

**Objective**

Write Family Substitution statements.

**Purpose**

Family substitution is involved when you attempt to access a file. In order to know what information must be provided to the MCP to have the correct file located, you must know how family substitution works.

**Resources**

A Series Disk Subsystem Software Overview, Section 5 - Planning and Installation

## A SERIES AND B 5000/B 6000/B 7000 CONCEPTS FAMILY SUBSTITUTION

### What Is Family Substitution?

**Family Substitution** is an automatic substitution of a user-specified family location for another family. It causes the system to attempt to locate the requested files on a substitute family or an alternate family.

Files for a user or group of users (private files) are usually grouped together and stored on one family. When the usercode is created, the family name is linked with the usercode for ease of the user.

At times it becomes necessary to access system files (compilers, utilities). Family substitution provides the mechanism so that files can be divided among disk families and still allow access to files the users need to perform their job.

Family substitution allows you to designate the family name where your private files are located and the family name where system files are located.

# A SERIES AND B 5000/B 6000/B 7000 CONCEPTS FAMILY SUBSTITUTION

## Family Substitution Parameters

### Target Family

System software and some application programs are written so that the system defaults to family **DISK** when searching for a file. The target family is the family name where the system will search for a file.

### Substitute Family

Substitute family is the place that you want the system to look for the file instead of the target family. It is usually the family name assigned to your usercode (location of private files).

### Alternate Family

If the files cannot be located on the substitute family, the system will also check the alternate family to see if the file can be located.

### ONLY

Only causes the system to search for files on the substitute family only.

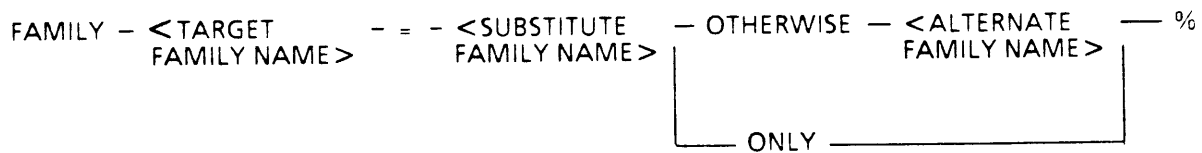


Figure 2-10 Family Substitution Syntax

Examples: FAMILY DISK = EDUCATION OTHERWISE DISK  
FAMILY DISK = EDUCATION ONLY  
FAMILY DISK = PAYPACK OTHERWISE PACK

**A SERIES AND B 5000/B 6000/B 7000 CONCEPTS  
FAMILY SUBSTITUTION**

*Family Substitution Chart*

<u>Function</u>	<u>Substitute Family</u>	<u>Alternate Family</u>
Read a File	Looks here first	Looks here second
Create a File	Always creates file here only	
CANDE & WFL: Remove or Change a File Name	Removes or Changes here	

Family Substitution is initially established when the usercode is created. It can also be updated when the usercode characteristics are updated.

A family substitution statement can be associated with a usercode, system job queues, Work Flow jobs (job attributes), Work Flow tasks (task attributes) and/or an editing MCS called CANDE. Family substitution can be changed as needed after creation.

**A SERIES AND B 5000/B 6000/B 7000 CONCEPTS  
FAMILY SUBSTITUTION**

**Practice**

Part A: Match the terms on the left with the descriptions on the right.

- |          |                         |    |                                                                         |
|----------|-------------------------|----|-------------------------------------------------------------------------|
| _____ 1. | PAST                    | a. | A file that contains file headers for each file on the family.          |
| _____ 2. | SYSTEM/ACCESS DIRECTORY | b. | The ability to easily specify where the system should search for files. |
| _____ 3. | FLAT DIRECTORY          | c. | A file used to update the FAST when a pack is brought on-line.          |
| _____ 4. | FAMILY SUBSTITUTION     | d. | A structure that contains an entry for each family on the system.       |
| _____ 5. | LAST                    | e. | An entry containing the row addresses and characteristics of the file.  |
| _____ 6. | FAST                    | f. | A structure containing pointers to file headers.                        |
| _____ 7. | FILE HEADER             | g. | A file comprised of the PAST and FAST.                                  |

Part B: See next page.



**A SERIES AND B 5000/B 6000/B 7000 CONCEPTS  
FAMILY SUBSTITUTION**

Part B: Write the Family Substitution statements necessary to accomplish the following:

1. The system should access files requested from the family EDUCATION. If a file is not located on this family, the system should not look on any other family.
2. You have been informed that all compilers will be available for your use on family TEST. Your usercode defaults to the family NEWAP, with no alternate family.
3. Your data files are located on family DBALL. All other files for your use are on family PRODUCTION.
4. All non-usercoded files are located on DISK. Your private files should be located on STUDENTPACK.

**A SERIES AND B 5000/B 6000/B 7000 CONCEPTS  
FAMILIES, FILES AND DISK DIRECTORIES**

**UNIT 5**

**FILE ATTRIBUTES**

**Objective**

Recognize some common file attributes.

**Purpose**

In order to access a disk file, you must be aware of the characteristics of the file.

**Resources**

A Series Systems An Introduction, Section 3 - A User's View of System Functions

A Series Disk Subsystem Software Overview, Section 2 - Disk Subsystem Concepts

A Series I/O Subsystem Reference Manual, Section 1 - Introduction  
Section 4 - File Attributes

# A SERIES AND B 5000/B 6000/B 7000 CONCEPTS FILE ATTRIBUTES

## What are File Attributes?

File attributes are specifications included with each file that define basic information about that file. These specifications are control parameters that contain all the information the I/O (Input/Output) subsystem needs to connect the correct physical file to a logical file and to process the file after the connection has been made.

Various file attributes are stored in the disk file header located in the Flat directory.

## Disk Storage

The basic **physical units** of disk storage are:

### BIT

the smallest unit of data. It has a value of on/off, 1/0, yes/no, or true/false.

### BYTE

1 character which contains 8 bits.

### WORD

6 bytes or 48 bits of data.

### SEGMENT/SECTOR

30 words or 180 bytes or 1440 bits.

The disk devices impose a structure restriction for reading or writing files. Each data transfer to or from a disk must be done in units of one or more segments or sectors.

The **logical units** of disk storage are:

### CHARACTER

a collection of 8 bits of data.

### FIELD

individual pieces of data which consist of a group of characters.

### RECORD

a collection of fields.

### FILE

a collection of records.

# A SERIES AND B 5000/B 6000/B 7000 CONCEPTS FILE ATTRIBUTES

## BLOCK

a whole number of records or segments.

## AREA/ROW

a whole number of blocks.

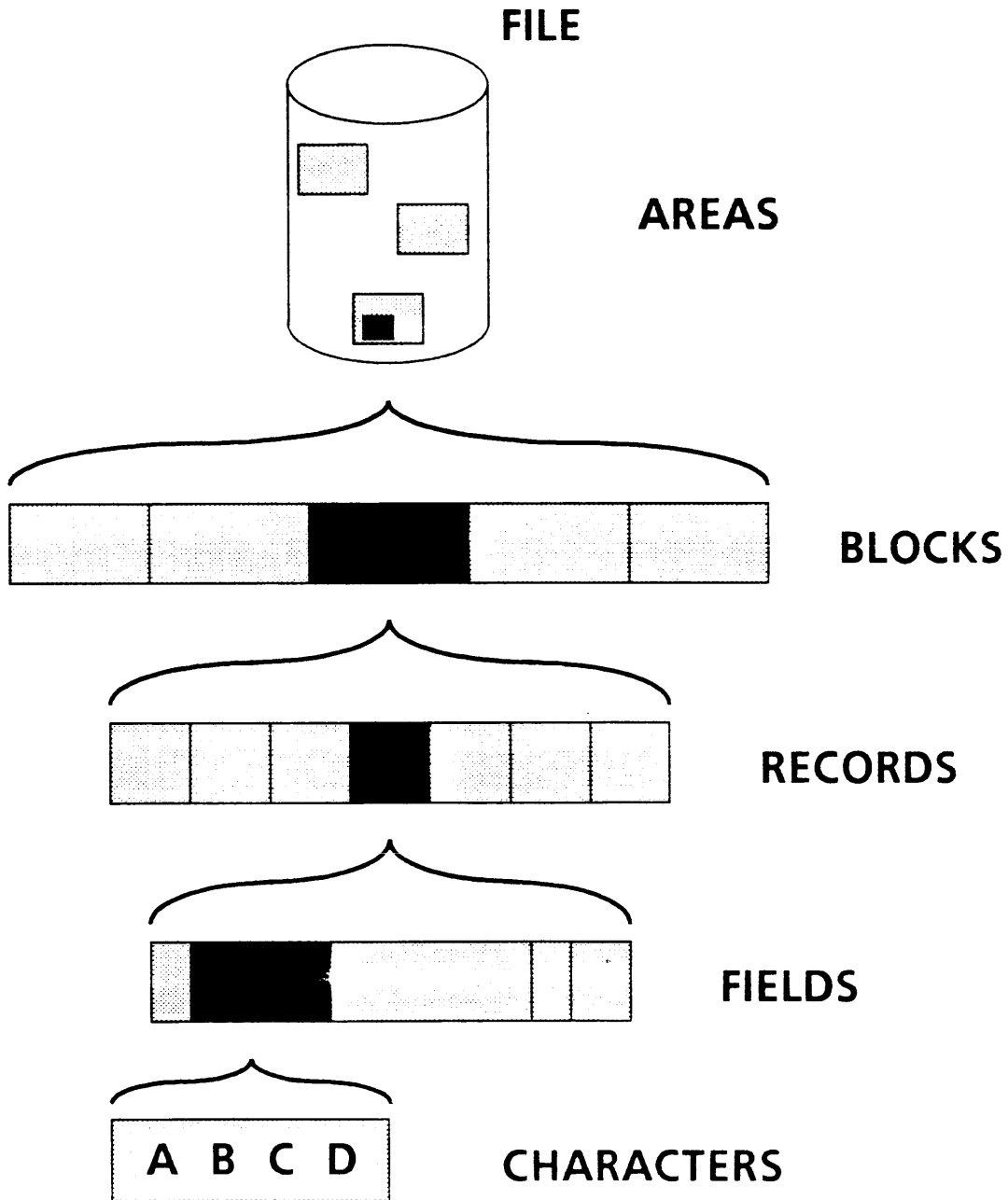


Figure 2-11 Disk Structures

## A SERIES AND B 5000/B 6000/B 7000 CONCEPTS FILE ATTRIBUTES

File attributes are used to:

Identify a File.

**FILENAME** - external file name.

**FAMILYNAME** - name of the family where file is stored.

**FAMILYINDEX** - relative number assigned to the member of family where file is stored (used if not spread among all members of the family).

**TITLE** - external file name and family name.

**INTNAME** - internal file name within a program.

**KIND** - the type of peripheral device where the file is located or to be created.

Indicate the File Structure.

**UNITS** - how the transfer of data is to be done (in words or characters).

**MAXRECSIZE** - maximum size of each record in the file in units.

**MINRECSIZE** - minimum size of a record in the file in units.

**BLOCKSIZE** - length of a block in units.

**AREAS** - number of rows a file can allocate.

**AREASIZE** - number of records in an area of a file.

**INTMODE** - internal character size of each record.

**FILEKIND** - internal structure and purpose of a file.

**BUFFERS** - number of buffers used in processing a file.

**FLEXIBLE** - file can be allocated more areas when file is full.

**DEPENDENTSPECS** - the logical file assumes the structure of the physical file when a file is opened.

## A SERIES AND B 5000/B 6000/B 7000 CONCEPTS FILE ATTRIBUTES

Control File Access (Security).

**MYUSE** - indicates how the file is to be used (input, output, I/O).

**NEWFILE** - determines if the system accesses an existing file or creates a new file.

Obtain Status of the File.

**CREATIONDATE** - returns the date when the file was first opened for creation.

**CREATIONTIME** - returns the time when the file was first opened during creation.

**USEDATE** - returns date file was last accessed.

**RESIDENT** - determines if a file exists.

Other file attributes can automatically translate the characters in a file, and return diagnostic information about attributes consistency and about physical I/O operations.

## A SERIES AND B 5000/B 6000/B 7000 CONCEPTS FILE ATTRIBUTES

The values of file attributes can be declared in the source program or set dynamically using Work Flow Language or ODT commands.

The following examples will declare a file with the following attributes.

The name of the file is EMPLOYEES/NAMES, and it is stored on a disk pack.

Each record is 14 words or 84 characters long.

Each block contains 30 records or 420 words.

Each area is 2100 records in size.

Total number of areas that can be used is 50.

The file is given 2 buffers.

### ALGOL Program Example

```
BEGIN
    FILE DATAFILE (KIND = DISK, MAXRECSIZE = 14, BLOCKSIZE = 420,
                    BUFFERS = 2, AREAS = 50, AREASIZE = 2100,
                    TITLE = "EMPLOYEES/NAMES.");
    •
    •
    •
END.
```

### COBOL74 Program Example

```
ENVIRONMENT DIVISION.
    •
    •
    •
FILE-CONTROL.
    SELECT DATAFILE ASSIGN TO 50 * 2100 DISK.

DATA DIVISION.
    FD DATAFILE    BLOCK CONTAINS 30 RECORDS;
                    RECORD CONTAINS 14 WORDS;
                    VALUE OF TITLE IS "EMPLOYEES/NAMES".
    •
    •
    •
STOP RUN.
```

**A SERIES AND B 5000/B 6000/B 7000 CONCEPTS  
FILE ATTRIBUTES**

**Practice**

Match the terms on the left with the descriptions on the right.

- |          |            |    |                                                                                                                       |
|----------|------------|----|-----------------------------------------------------------------------------------------------------------------------|
| _____ 1. | AREAS      | a. | This attribute specifies the number of logical records in an area of a disk file.                                     |
| _____ 2. | BLOCKSIZE  | b. | The value of this attribute is the number of areas (or rows) a disk file can allocate.                                |
| _____ 3. | AREASIZE   | c. | The value of this attribute is the length of a block.                                                                 |
| _____ 4. | MAXRECSIZE | d. | This attribute indicates whether or not a disk file can be allocated more areas.                                      |
| _____ 5. | UNITS      | e. | This attribute specifies the maximum size of records in the logical file.                                             |
| _____ 6. | FLEXIBLE   | f. | This attribute describes the peripheral unit associated with the logical file.                                        |
| _____ 7. | KIND       | g. | This attribute indicates whether the transfer of data in the file will be word or character oriented.                 |
| _____ 8. | TITLE      | h. | This attribute can be programmatically changed and is the internal file name.                                         |
| _____ 9. | INTNAME    | i. | This attribute is the external file name which is used to associate a logical file with a physical or permanent file. |



**SECTION 3**  
**INTERPRO OVERVIEW**

***INTRODUCTION***

**Section Objective**

Use MARC.

**Purpose**

To use MARC, you need an introduction to the InterPro products.

**Unit Objectives**

Identify InterPro and its features.

Identify InterPro products.

Use MARC screens to do inquiries.

# A SERIES AND B 5000/B 6000/B 7000 CONCEPTS INTERPRO OVERVIEW

## UNIT 1

### INTERPRO INTRODUCTION

#### Objective

Identify InterPro products.

#### Purpose

To determine if InterPro products can be used at your installation, an overview of InterPro is necessary.

#### Resources

A Series System An Introduction, Section 2 - Virtual Memory, Stacks, and Other System Concepts  
Section 3 - A User's View of System Functions

A Series Interactive Datacomm Configurator (IDC) User's Guide, Section 1 - Introduction  
Section 2 - General Concepts

A Series Communications Management System (COMS) Capabilities Manual

A Series Screen Design Facility (SDF) Capabilities Manual

A Series Extended Retrieval with Graphic Output (ERGO) User's Manual, Section 1 - Introduction  
Section 2 - Overview

# A SERIES AND B 5000/B 6000/B 7000 CONCEPTS INTERPRO INTRODUCTION

## What Is InterPro?

InterPro is **INTER**active software that addresses personnel **PRO**ductivity for A Series installations. InterPro is a group of products that was totally new to Burroughs software family when first released. The InterPro products are not extensions to Burroughs other software products. InterPro products are integrated products so that duplication of functions are eliminated, performance is improved and a higher level of system integrity can be reached.

## *Features of InterPro*

High Performance

Reduced Application Programming

Simplified Installation of System Software

Simplified Maintenance of System Software

Reduced Training

Utilization of the Major Architectural Advances Introduced in the A Series

## *Menus*

A key element of the InterPro products are its use of menus and menu-assisted support. All products guide the user from one operation to another. The user can request help at any point when questions arise and receive the help response on the terminal.

# A SERIES AND B 5000/B 6000/B 7000 CONCEPTS INTERPRO INTRODUCTION

## *InterPro Products*

The InterPro Products are divided into three categories: Communications, Data Management, and Operations Control.

The communications grouping includes three InterPro products. These are:

### **Interactive Datacomm Configurator (IDC)**

IDC is an interactive, menu-driven utility. IDC is designed to define the data communication network for your system. It eliminates the need to use the Network Definition Language (NDL II) to define the hardware setup of your data communications network.

IDC will also handle updates to the data communication network, which can be entered and implemented on-line. This update capability reduces the time required to make modification to the network.

IDC can be used interactively using IDC commands or can be used in a batch mode.

### **Communications Management System (COMS)**

COMS is a Message Control System (MCS). COMS will control the flow of data in the communications network. It has many features that are required by users to route messages from terminals to programs and back to terminals.

COMS provides high performance, is flexible, and is a reliable communications monitor.

It is designed to accommodate particular needs of an installation by allowing new functions to be specified, in addition to the inbuilt features. COMS Utility uses menus and provides on-line assistance when requested by the operator in order to define and maintain the system.

### **Screen Design Facility (SDF)**

SDF is an interactive screen form definition and screen painter program. SDF allows a screen designer to use a terminal to build screens instead of a programming language. SDF will perform some data validation during data entry, thus eliminating the need for the application programs to validate the data.

SDF improves screen development and brings on-line systems into production quickly.

It also allows the end user to become involved in the design of screens used in applications.

## **A SERIES AND B 5000/B 6000/B 7000 CONCEPTS INTERPRO INTRODUCTION**

The data management grouping includes two InterPro products. These are:

### **Advanced Data Dictionary Systems (ADDS)**

ADDS is an interactive, menu-driven program that provides for centralized storage and retrieval of data definitions. It allows the user to define the physical and logical databases for the system, conventional files, relationships among data elements, and screen formats.

ADDS makes it possible to eliminate inconsistencies and redundancies in data and to control additions, changes, and deletions. These controls and others ensure data integrity.

### **Extended Retrieval with Graphic Output (ERGO)**

ERGO is a query program that provides quick, on-line access to DMS II databases in tabular and graphic forms. ERGO produces reports using the input parameters specified by the user. The reports can contain graphic output in addition to normal data reporting. ERGO eliminates much of the programming time used to code report programs.

The operations control grouping includes only one InterPro product.

### **Menu Assisted Resource Control (MARC)**

MARC is a menu-guided and command-driven interface to the system. Marc provides screen after screen to the user, formats the user's responses so that the MCP will accept the input, passes input to the MCP and receives responses from the MCP which are displayed for the user.

MARC can be used by any type of user (ODT operator, programmer, etc.) to interface to the system. MARC can be used in different modes to allow for the experienced user or provide assistance for the beginning user.

# **A SERIES AND B 5000/B 6000/B 7000 CONCEPTS INTERPRO OVERVIEW**

## ***UNIT 2***

### ***MARC USAGE***

#### **Objective**

Use MARC screens to do inquiries.

#### **Purpose**

To make operations of the system easier, quicker and more user-friendly, you need to use MARC to communicate with the system.

#### **Resources**

A Series System An Introduction, Section 3 - A User's View of System Functions

A Series Menu-Assisted Resourced Control User's Guide, All Sections

# A SERIES AND B 5000/B 6000/B 7000 CONCEPTS MARC USAGE

## What Is Marc?

MARC is **Menu Assisted Resource Control**. MARC provides a menu-driven interface and a direct command entry interface to the system.

It was developed for systems operators, programmers, and end users. MARC menus and help facility make it possible to perform actions on the system without having any knowledge of system commands. MARC also allows the experienced operator to interface with the system without having to use all of the MARC menus.

MARC gathers input from users at any number of different stations, translates the input into specific commands, and passes these commands to COMS or other appropriate components of the system. MARC then gathers the command output and sends it to the appropriate stations.

## *Methods of Menu Operation*

Menu Directed Path (known as **Menu Mode**) steps users through a series of menus to the final response for the request. As each menu is displayed, a choice is made, another menu displayed, a choice is made or input fields filled in, until MARC has enough information to format a message to the system.

Direct Menu Selection (known as **Choice Field Typeahead**) allows users to apply the Menu Directed Path method but to bypass menus already mastered. It allows you to display only the menus you need. You supply information for the next logical menu on the current menu along with the current menu choice and then bypass the next menu. Typeahead allows only menu selection keys and/or form parameters to be entered in the choice field.

Menu Command Line (known as **Command Mode**) is used to enter a request in the command language of the MCP. It allows you to bypass the menus and enter commands directly. These commands must be entered in the correct syntax as defined by the MCP.





## A SERIES AND B 5000/B 6000/B 7000 CONCEPTS MARC USAGE

### *Actions*

When the MARC screen is displayed, the list of available actions will differ depending on the screen requested. Actions allowed at some time in MARC are:

<b>GO</b>	- must be followed by a screen name. It causes MARC to display the requested screen next.
<b>HOme</b>	- causes MARC to display the home menu next.
<b>PArent</b>	- causes MARC to display the screen that is the parent to the screen that you are viewing.
<b>PRev</b>	- causes MARC to display the last screen viewed.
<b>REturn</b>	- causes MARC to display the screen that generated the command. Usually you will be on a screen where the responses from the command are displayed for you.
<b>+</b>	- causes MARC to display the next screen for output or help text. This is used for scrolling forward process.
<b>-</b>	- causes MARC to display the last screen of output or help text. This is the backward scrolling process.
<b>COmnd</b>	- causes MARC to initiate the command mode function and receive input in the format required by the MCP.
<b>KEys</b>	- causes all of the help keywords that are available through MARC to be displayed.

### *Action Field Typeahead*

The action field typeahead feature allows you to make a sequence of entries at once in the action field rather than merely one entry. The entries are executed one at a time by MARC.

Action field typeahead rules:

1. First item in the sequence must be a valid action for the displayed screen. If GO is selected, a screen name must follow.
2. After the screen action, enter a menu selection or series of menu selections. The first selection must occur on the screen that the screen action leads to. Other selections must occur on the menu that the previous menu selection led to.
3. Following selections, form field values can be entered. Order of the parameters must match the order on the menu that the selections led to.

## **A SERIES AND B 5000/B 6000/B 7000 CONCEPTS MARC USAGE**

### *Help*

All of the InterPro products provide the user with on-line help. Help is available in 2 forms.

#### **ShortHelp Text**

ShortHelp displays a 2 line help message, located at the bottom of the screen.

To request ShortHelp, move the cursor to the item for which help is desired and press the SPCFY key once.

#### **Help Mode**

Help mode displays 1 or more screens containing information to assist you . The number of screens to be displayed depends on the item selected. This can be referred to as Long Help.

To request help mode, move the cursor to the item for which help is desired and press the SPCFY key once. Wait for the shorthelp text to be displayed. Then without moving the cursor, press SPCFY a second time. The first screen of help information will be displayed.

To view the other screens of help information, you use the scrolling capability (Action + or -).

The help information displayed for you may vary depending on the system on which you are working. MARC allows the user to customize the help information. The released MARC product has standard help screens which can be used or modified by the installation to customize help information.

# A SERIES AND B 5000/B 6000/B 7000 CONCEPTS MARC USAGE

## *MARC Logon/Logoff*

MARC will present a logon screen to any terminal that is attached to the COMS MCS. A logon screen may not be presented if the terminal cannot handle page entry, if the terminal is the ODT or if the COMS Utility has assigned that terminal to a different program other than MARC.

For a terminal assigned to a program other than MARC, enter **?ON MARC** to display the logon screen.

At the ODT, enter **??MARC** to receive the logon screen.

Terminals must enter a valid usercode and password.

The only exception is the ODT and a terminal defined as a superuser-capable. A superuser-capable terminal is a COMS status for a terminal to determine security clearance. The ODT and superuser-capable can enter an \* as a valid usercode. They are allowed access rights to the system without special identification.

```
LOGON - Menu-Assisted Resource Control          02:03 PM          MARC

                               Welcome.

Please enter your user code [          ]
...and your password [          ]

The USERCODE or PASSWORD you have entered is not valid.
Please reenter your USERCODE and PASSWORD.
```

Figure 3-2 MARC Logon Screen

**A SERIES AND B 5000/B 6000/B 7000 CONCEPTS  
MARC USAGE**

MARC provides three ways to sign off.

**BYE**

This will end the current MARC session, release print files for printing and log you off the system.

**SPLIT**

This will end the current MARC session, release any print files, and immediately start a new MARC session under the same usercode.

**HELLO <usercode>**

This will end the current MARC session, release any print files, and immediately start a new MARC session under the usercode entered.

# A SERIES AND B 5000/B 6000/B 7000 CONCEPTS MARC USAGE

## MARC Home Menu Example

Once you are logged on, the MARC Home Menu is displayed.

Each usercode can have a different Home Menu.

When a user logs on to MARC, the usercode entered is used to determine what Home menu should be displayed.

```
MARC - MENU-ASSISTED RESOURCE CONTROL          02:39 PM          MARC
Action: [                                         ]
          H0me PReV GO PAREnt COmnd              (Press SPCFY for Help)

INTRO Intro to MARC      SYS   System Control    NEWS  System News
JD     Job Display       CONFIG System Config  DATE  Date and Time
JC     Job Control
JQ     Job Queues        PK    Disk Pack          FILE  File Management
PS     Printing System   DK    Head-per-trk Disk  LIBS  System Libraries
          MT    Magnetic Tape
RUN    Run A Task        LP    Line Printer        DC    DataComm Control
START  Start WFL Job     IP    Image Printer       BNA   BNA Commands
UTIL   System Utilities  CR    Card Reader          COMS  COMS Displays
          CP    Card Punch          CC    COMS Control
MEM    Memory Management HC    Host Control          SEND  Send Messages
SP     Special Programs  PROC  Processors            SC    Session Control
DUMPS  Dumps             MM    Memory Modules      ON    Change Window
LOG     Logging          OTHER Other Devices     CANDE Cande Window
SWAP   Swapper

Choice: [                                         ]

Burroughs B6900:2372 MARC (version 36.130) SYSEDB6900.
User = USER1; Session = 1287.
```

Figure 3-3 MARC Home Menu Privileged and Systemuser

# A SERIES AND B 5000/B 6000/B 7000 CONCEPTS MARC USAGE

## Command Example

To demonstrate the use of MARC screens and command mode, we wish to determine if a file MARC/FILE/EXAMPLE is present on our system.

## Menu Mode

The following steps and screens demonstrate using the **Menu Mode** to determine if a file is present.

On the Home menu select the choice **FILE** (File Management).

```
MARC - MENU-ASSISTED RESOURCE CONTROL          02:04 PM          MARC
Action: [                                     ]
      Home PRev GO PArent COmnd          (Press SPCFY for Help)

INTRO Intro to MARC          START Start WFL Job          SPLIT Print Backup File
J      Active Entries          DIR  List My Files          BYE  Log Off
W      Waiting Entries        COPY Copy File          PASSWD Change Password
C      Completed Entries      CHANGE Change File Name  LANG  Change Language
SHOW  Show Print Queue        REMOVE Remove File          Other Menus:
Y      Task Status            SEC  Change Security      JDC  Job Disp/Control
TI     Task Times            INFO Show File Info      PS   Printing System
DS     Discontinue Task      FAMILY Show Family        UTIL System Utilities
ST     Suspend Task          CHFAM Change Family        SYSINF System Info
OK     Resume Task          WINDOW Current Windows    FILE File Management
SEND  Send Message          ON   Change Window        COMS COMS Displays
MSG   Read Message          CANDE Cande Window        SC   Session Control
RUN   Run a Program          NEWS Read News

Choice: [FILE                                     ]

Burroughs B6900:2372 MARC (version 36.130) SYSEDB6900.
User = CONCEPTS; Session = 1212.
```

Figure 3-4 MARC Home Menu Non-Privileged User

## A SERIES AND B 5000/B 6000/B 7000 CONCEPTS MARC USAGE

The FILE choice causes the File Management and Disk Directories Screen (File screen) to be displayed.

On the File Screen, select the choice FDIR (Show Files/Directories).

```
FILE - FILE MANAGEMENT AND DISK DIRECTORIES      02:05 PM      MARC
Action: [                                          ]
        H0me PReV GO PArent COmnd                (Press SPCFY for Help)

COPY  Copy Or Add A File                        FAMILY Display/Change Family Specific
CC    Copy & Compare A File
CHANGE Change Disk File Name
REMOVE Remove Disk File
SEC   Change Disk File Security
START Start A WFL Job

FILES Show My File Directory
FDIR  Show Files/Directories

INFO  Show Key Info About A File
DETAIL Show Details About Files
REPORT Print FILEDATA Disk Report

Choice: [FDIR                                          ]
```

Figure 3-5 File Management & Disk Directories Screen



## A SERIES AND B 5000/B 6000/B 7000 CONCEPTS MARC USAGE

The FDIR selection causes the File Titles and Directories Screen (FDIR screen) to be displayed.

On this screen you are to fill in parameters instead of filling in the choice line, as in the 2 previous screens.

Enter the file name MARC/FILE/EXAMPLE.

Your usercode will be added by MARC, and family substitution will provide the family name.

```

          FDIR - Display File Titles & Directories          02:06 PM      MARC
Action: [
          Home PRev GO PAREnt COmnd                        (Press SPCFY for Help)
]

You may display a list of files and directories in ONE of two ways:

  1 By USERCODE (or *). . . . . [
      FILE TITLE or DIRECTORY. . . [MARC/FILE/EXAMPLE
]

                                OR

  2 Under all USERCODEs (type an X) . . . . . [ ]

-----

On the FAMILY (optional) . . . . . [
Number of directory LEVELs (optional). . . . . [ ]
```

Figure 3-6 File Titles and Directories Screen

**A SERIES AND B 5000/B 6000/B 7000 CONCEPTS  
MARC USAGE**

The output from our request will be displayed:

On the bottom 2 lines of the screen where we entered the parameters.

OR

On the Command Output Screen if the output would not fit on 2 lines.

The Command Output Screen will display the first page of output. Additional pages can be viewed by using the scrolling action.

```
          OUTPUT - MARC COMMAND OUTPUT                02:08 PM      MARC
Action: [Return                                     ]
        Home GO Return Comnd + -                    (Press SPCFY for Help)

Response returned at 02:08 PM:

FILE (CONCEPTS)MARC/FILE/EXAMPLE ON SYSTEMSED (COBOL74SYMBOL)
DATE AND TIME OF CREATION: MONDAY SEP 23,1985 (85266) AT 16:53:08
          LAST ACCESS: FRIDAY JAN 31,1986 (86031) AT 19:56:36
          LAST ALTER: MONDAY SEP 23,1985 (85266) AT 16:53:08
SIZE IN SEGMENTS: 14
SECURITY: PRIVATE - USAGE: READ/WRITE

FILE MARC/FILE/EXAMPLE
```

Figure 3-7 MARC Output Command Screen

# A SERIES AND B 5000/B 6000/B 7000 CONCEPTS MARC USAGE

## Choice Field Typeahead

The following screens and steps demonstrate using the **Choice Field Typeahead** to determine if the file is present.

On the Home menu, select the choice FILE. Follow that by the selection that you want on the File Management and Disk Directories screen, which would be FDIR. This would cause the system to bypass the File Management and Disk Directories screen and display the File Titles and Directories screen.

```

MARC - MENU-ASSISTED RESOURCE CONTROL                02:08 PM      MARC
Action: [                                             ]
          H0me PReV GO PAREnt COmnd                    (Press SPCFY for Help)

INTRO Intro to MARC      START Start WFL Job          SPLIT Print Backup File
J      Active Entries    INFO Show File Info          BYE   Log Off
W      Waiting Entries   DIR  List My Files          PASSWD Change Password
C      Completed Entries COPY Copy File             LANG  Change Language
SHOW  Show Print Queue  CHANGE Change File Name
                                REMOVE Remove File
                                SEC  Change Security    JDC  Job Disp/Control
Y      Task Status      INFO Show File Info    PS   Printing System
TI     Task Times       FAMILY Show Family     UTIL System Utilities
DS     Discontinue Task CHFAM Change Family     SYSINF System Info
ST     Suspend Task     WINDOW Current Windows
OK     Resume Task      ON   Change Window     FILE File Management
SEND  Send Message     CANDE Cande Window     COMS COMS Displays
MSG   Read Message     NEWS  Read News        SC   Session Control
RUN   Run a Program

Choice: [FILE FDIR ]

```

Figure 3-8 MARC Home Menu Non-Privileged User

# A SERIES AND B 5000/B 6000/B 7000 CONCEPTS MARC USAGE

The FILE FDIR selection causes the File Titles and Directories Screen to be displayed.

On this screen, choices are not allowed. You are to fill in parameters instead.

If you knew the order of the parameters to enter, the parameters could have been entered following FILE FDIR on the Home menu.

The output from our request will be displayed on the bottom 2 lines of the screen where we entered the parameters or on the Command Output Screen if the output requires more than 2 lines.

```
          FDIR - Display File Titles & Directories          02:11 PM          MARC
Action: [                                                    ]
          H0me PRev GO PAREnt COmnd                          (Press SPCFY for Help)

You may display a list of files and directories in ONE of two ways:

  1 By USERCODE (or *). . . . . [                            ]
    FILE TITLE or DIRECTORY. . . [MARC/FILE/EXAMPLE1
                                                    ]

                                OR

  2 Under all USERCODEs (type an X) . . . . . [ ]

-----

On the FAMILY (optional) . . . . . [                            ]
Number of directory LEVELs (optional). . . . . [                ]

No file(s) were found on SYSTEMSED
```

Figure 3-9 File Titles and Directories Screen

# A SERIES AND B 5000/B 6000/B 7000 CONCEPTS MARC USAGE

## Command Mode

The following steps and screens demonstrate using the **Command Mode** to determine if the file is present.

On the Home menu select the Action CO (command). CO is then followed by the ODT command to determine if a file resides on disk. This command is PD (Print Directory).

You must use the syntax of PD as defined in the ODT manual.

Enter CO PD <file name>, for example CO PD MARC/FILE/EXAMPLE.

The output appears on either the Command Output Screen or the bottom 2 lines of the parameter screen (refer to Figure 3-7 for the PD output).

```

          MARC - MENU-ASSISTED RESOURCE CONTROL                02:12 PM      MARC
Action: [CO PD MARC/FILE/EXAMPLE                               ]
          Home PRev GO PArent COmnd                          (Press SPCFY for Help)

INTRO  Intro to MARC      START  Start WFL Job      SPLIT  Print Backup File
J      Active Entries
W      Waiting Entries    DIR    List My Files      BYE    Log Off
C      Completed Entries  COPY  Copy File        PASSWD Change Password
SHOW   Show Print Queue  CHANGE Change File Name  LANG   Change Language
                                     REMOVE Remove File
                                     SEC    Change Security    JDC    Job Disp/Control
Y      Task Status        INFO  Show File Info    PS     Printing System
TI     Task Times        FAMILY Show Family    UTIL  System Utilities
DS     Discontinue Task  CHFAM Change Family  SYSINF System Info
ST     Suspend Task
OK     Resume Task
                                     WINDOW Current Windows  FILE   File Management
SEND   Send Message      ON    Change Window     COMS   COMS Displays
MSG    Read Message      CANDE Cande Window   SC     Session Control
RUN    Run a Program     NEWS  Read News

Choice: [                                                       ]
```

Figure 3-10 MARC Home Menu Non-Privileged User

# A SERIES AND B 5000/B 6000/B 7000 CONCEPTS MARC USAGE

## Tasking Mode

In addition to Menu Mode and Command Mode, MARC also has a Tasking Mode, which allows programs to be run through MARC.

Tasking Mode is initiated from Menu Mode by the COPY or RUN menu selections, or by any of the system utility menu selections, or from Command Menu when any Work Flow command is entered.

In tasking mode, MARC will display screens for the user to enter all information needed to execute a program. This may require entries in multiple screens.

When the program is executed, the task status screen will be displayed. This screen displays the mix number of the program, name of program and the status information about the program. The screen will be updated as the program executes to keep you informed of the program status.

```
1250 - (CONCEPTS)"MARC WFL".                                02:17 PM      MARC
Action: [REturn                                             ]
        H0me REturn MSg ON SPlit

Parameter= COPY MARC/FILE/EXAMPLE FROM SYSTEMSED (PACK) TO SYSTEMSED (PACK)
Task status= TERMINATED

Elapsed= 26.913      Processor= 0.487      I/O= 0.403

14:16 1243\1251 BOT (CONCEPTS)WFLCODE ON SYSTEMSED
14:16 1243\1252 BOT *LIBRARY/MAINTENANCE
14:17 1252 PK51 (CONCEPTS)MARC/FILE/EXAMPLE REMOVED ON SYSTEMSED
14:17 1252 (CONCEPTS)MARC/FILE/EXAMPLE COPIED FROM SYSTEMSED TO SYSTEMSED
14:17 1243\1252 EOT *LIBRARY/MAINTENANCE
14:17 1243\1251 EOT (CONCEPTS)WFLCODE ON SYSTEMSED
14:17 1243\1250 EOT (CONCEPTS)MARC WFL

EOJ
```

Figure 3-11 Sample Tasking Screen

## A SERIES AND B 5000/B 6000/B 7000 CONCEPTS MARC USAGE

### *Security*

A usercode can have a status of Commandcapable, Systemuser, Privileged, or Non-privileged.

Commandcapable limits the COMS commands that the user can enter. (Allowable commands WRU, PASS, ON, PURGE, WINDOWS, RESUME, SUSPEND, CLOSE, END, and BYE.)

Systemuser allows access to Controller functions that affect the operations of the system. Systemusers are given a system-wide view through MARC at a terminal, as if they were at the ODT.

A terminal can have a status of Superuser.

Superuser allows entry of \* as a usercode and gives the user systemuser status but is still non-privileged. Superuser status has the advantage of allowing users to log on to the system when no usercodes have been declared in the SYSTEM/USERDATAFILE. The ODT is automatically considered a superuser.

MARC can be modified to display unique help information and unique menus.

The standard MARC release provides standard menus and help information. Menus can be modified to users' specifications. Menus can also be changed to reflect the differences between usercodes. Modification of menus also includes addition of menus.

The ODT can run MARC.

To receive the logon screen, you enter ??MARC. This will cause the system to treat the ODT as if it were a terminal. The ODT can now operate as any other terminal attached to the system.

To return the ODT back to ODT mode, sign off from MARC or enter ??ODT.

**A SERIES AND B 5000/B 6000/B 7000 CONCEPTS  
MARC USAGE**

This page left blank for formatting.



**SECTION 4**  
**SYSTEM INITIALIZATION CONCEPTS**

# A SERIES AND B 5000/B 6000/B 7000 CONCEPTS SYSTEM INITIALIZATION CONCEPTS

## *INTRODUCTION*

### **Section Objective**

Identify the different levels of system initialization.

### **Purpose**

In order to control the system, you must be able to determine when a system requires initialization and/or re-initialization. You must also determine what level of initialization is to be performed.

### **Unit Objectives**

Identify the different levels of system initialization.

Identify how to change the MCP file that the system operates under.

# A SERIES AND B 5000/B 6000/B 7000 CONCEPTS SYSTEM INITIALIZATION CONCEPTS

## UNIT 1

### SYSTEM INITIALIZATION

#### Objective

Identify the different levels of system initialization.

#### Purpose

In order to control the system, you must be able to determine when a system requires initialization and/or re-initialization. You must also determine what level of initialization is to be performed.

#### Resources

A Series Systems An Introduction, Section 4 - Planning for Effective Operations

A Series A 3 System Software Installation Guide, Section 6 - Simple Halt/Loading of Your System  
Section 7 - Halt/Loading Using UTILOADER and  
LOADER

A 9 System Software Installation Guide, Section 6 - Simple Halt/Loading of Your System  
Section 7 - Halt/Loading Using UTILOADER and LOADER  
Section 8 - Creating a Halt/Load Pack on a Non-Running  
Machine

# A SERIES AND B 5000/B 6000/B 7000 CONCEPTS SYSTEM INITIALIZATION

## What Is System Initialization?

System initialization is the process used to bring a system to a normal operating condition (available for intended use). Initialization of the system will cause a new copy of the MCP (Master Control Program) to be applied to the system from either a system tape or a disk on the system.

Initialization of the system is required:

- When the system is first powered up.

- When the system develops a fatal error.

- When the MCP becomes corrupted.

- To install a different level of the MCP.

There are three methods used to initialize a system: Halt/Load, Cold Start, and Cool Start.

## A SERIES AND B 5000/B 6000/B 7000 CONCEPTS SYSTEM INITIALIZATION

### *Halt/Load Initialization*

Halt/Load is the least severe form of system initialization.

Halt/Load initialization is the process of bringing all processing in the system to a stop (halt) and then loading the MCP into memory from the Halt/Load Pack which is mounted on the Halt/Load Unit and having the MCP executed.

A Halt/Load Pack (H/L Pack) is a pack which contains an MCP that is the current operative MCP.

A Halt/Load Unit (H/L Unit) is the disk device that the H/L pack is mounted on.

The terms H/L Unit and H/L Pack are used interchangeably.

The H/L Unit is usually family "DISK" It is the location of the SYSTEM/ACCESS directory and the MIT (Mirror Information Table).

The unit number of the H/L Unit is contained in the bootstrap, which is a collection of data and machine instructions capable of locating, loading, and executing the MCP file .

A Halt/Load may be performed:

To compensate for memory parity errors.

To handle a system thrashing condition.

To compensate for software bugs.

To correct a hung system condition.

At the discretion of the operator.

To reconfigure the system.

## A SERIES AND B 5000/B 6000/B 7000 CONCEPTS SYSTEM INITIALIZATION

To Halt/Load a system, you enter **??PHL** (Programmatic Halt/Load) at the ODT.

This is a primitive command which bypasses the ordinary handling of ODT commands and causes an immediate programmatic Halt/Load of the system. When the system has been reset, the operator is requested to verify or enter the date and time.

On some of the systems, a Halt/Load can be or must be done by depressing the Enable , Halt and Load buttons on the main cabinet.

The Enable and Halt buttons are depressed simultaneously. You then press the Load Mode button to illuminate it, followed by the Load button. The system will then request verification or entry of the date and time.

Files are retained by the system and the MCP option Autorecovery will control the recovery process of programs that were executing. If necessary, recovery of databases will be performed.

### *Cold Start Initialization*

Cold start is the most severe form of initialization. All files are removed from the H/L unit. A new copy of the MCP is loaded onto the H/L unit from a system tape.

A Cold Start:

Is done to first bring up the system.

Is done to clean up a corrupted H/L unit.

May be done to install a new software release.

May be done periodically to clean off the H/L unit.

## A SERIES AND B 5000/B 6000/B 7000 CONCEPTS SYSTEM INITIALIZATION

### Cold Start Procedure

The UTILOADER and LOADER programs must be used to perform a Cold Start.

Detailed directions for this procedure should be supplied for each site by the field engineer or the site's system administrator. Following are some general guidelines for performing a Cold Start.

The UTILOADER program must be loaded into the system. The load procedure for this differs depending on which system you are using. See your system reference manual for detailed directions on loading UTILOADER.

Once UTILOADER is running, select the TAPELOAD UTILOADER command. This will inform UTILOADER that you wish to load a program from a tape.

UTILOADER will now request the location of the tape drive where the tape is mounted. Enter the name of the loader program, name of the tape, and the mnemonic and unit number of the tape drive. Example: SYSTEM/LOADER FROM SYSTEMAS ON MT14

The LOADER program is loaded into memory, executes and displays ODT INPUT REQUIRED. This message will also be displayed upon the completion of a command. You enter commands to LOADER in the correct order to complete the Cold Start.

On some systems it is necessary to load the Controlware for controlling the disk devices and possibly other controllers. If this is necessary for your system, load the controlware.

Identify the H/L unit. Example: HALTLOADUNIT 44.

Enter the largest overlayable array size. Example: OLAYROW 1200. This step distinguishes a Cold Start from a Cool Start.

Load the MCP file. Example: LOAD MCP SYSTEM/MCP FROM TAPE  
SYSTEMAS TO DISKPACK 44 NAME = DISK SERIAL = 190373

LOADER will display COLD START IS REQUESTED PK 44 DISK ENTER OK TO CONTINUE. You enter OK to continue with the Cold Start. When the OK is received, the disk is purged and all files on the H/L unit are lost.

Enter the date. Example: 01/01/86

Terminate the LOADER program by entering STOP.

## A SERIES AND B 5000/B 6000/B 7000 CONCEPTS SYSTEM INITIALIZATION

The Cold Start should now be complete. To bring the system to an operational state, a Halt/Load must now be performed. You can use the HALTLOAD UTILoader command to do this.

The Cold Start will load only the MCP file onto the H/L unit. All other files must be copied to disk from your backup media following the Halt/Load by using the LIBRARY/MAINTENANCE program.

No recovery will be performed after a Cold Start. All recovery data located on the H/L unit is lost during the Cold Start.

### *Cool Start Initialization*

Cool Start is an initialization that falls in severity between a Halt/Load and a Cold Start. It is more severe than a Halt/Load but less severe than a Cold Start.

Cool Start causes an MCP file to be loaded to the Halt/Load Unit from the system tape. The MCP on the H/L unit is replaced and the other files on the H/L unit are retained.

A Cool Start is performed when the integrity of the MCP file on the H/L unit is in question.

### *Cool Start Procedure*

The Cool Start procedure is similar to the Cold Start procedure. The procedure is the same except for the OLAYROW entry, which is omitted for a Cool Start.

Once the MCP is loaded to the H/L unit, a Halt/Load must be performed to load and execute the MCP and to start the recovery procedure.

### *Power Failure*

A Halt/Load is usually the initialization procedure that is needed following a power failure.

The power failure causes the Controllers to lose the Controlware.

On most systems you are required to load the Controlware for the Controllers. This requires the UTILoader program to be used. The Controlware must be loaded before the Halt Load is attempted.



# A SERIES AND B 5000/B 6000/B 7000 CONCEPTS SYSTEM INITIALIZATION CONCEPTS

## UNIT 2

### CHANGING MCP FILES

#### **Objective**

Identify how to change the MCP file that the system operates under.

#### **Purpose**

In order to have the system use a different operating system, you must know the what commands to use.

#### **Resources**

A Series A 3 System Software Installation Guide, Section 6 - Simple Halt/Loading of Your System

A Series A 9 System Software Installation Guide, Section 6 - Simple Halt/Loading of Your System

## A SERIES AND B 5000/B 6000/B 7000 CONCEPTS CHANGING MCP FILES

### Causing A Halt/Load to Change the MCP

Under certain conditions, you will want to load a new MCP code file and designate it as the new MCP code file to be used by the system. A second copy of the MCP file located on the system can help to avoid the need to perform a Cold Start.

A Halt/Load is necessary to change the operating system that is to be used to control the system. You must inform the system which MCP file is to be used before the Halt/Load is performed.

### *New MCP on H/L Unit*

The CM (Change MCP) ODT command is used to direct the system as to which MCP file to load into memory. The CM command is entered before the Halt/Load is performed. When the Halt/Load is requested, the system will use the MCP file named in the CM command.

Example: CM SYSTEM/MCP36140

For a new release of the MCP, you will want to test that MCP before permanently updating the system to use the new MCP file. The CM ODT command is still used.

Example: CM # SYSTEM/MCP36310

For the next Halt/Load only, the new MCP file will be used. A second Halt/Load will cause the system to revert back to the original MCP file.

### *New MCP Not on H/L Unit*

If the new MCP file is located on a family other than the H/L unit, you must use the ODT command HLUNIT (Specify Halt Load Unit) or BOOTUNIT for B 5900/B 6900, to indicate a change of H/L units. The CM command is entered first to identify the new MCP file (include family name) and the HLUNIT command follows. When a Halt/Load is performed, the values take effect.

Example: CM SYSTEM/MCP35220 ON TESTPACK

HLUNIT PK 46

## A SERIES AND B 5000/B 6000/B 7000 CONCEPTS CHANGING MCP FILES

### *Primitive Command*

A primitive command is a command that is processed directly by the MCP, bypassing the ordinary handling of ODT commands.

A primitive CM command is available. All processing will be halted immediately and the system loaded with the new MCP file.

Example: ??CM SYSTEM/MCP36140

## A SERIES AND B 5000/B 6000/B 7000 CONCEPTS CHANGING MCP FILES

### Practice

Match the terms on the left with the descriptions on the right.

- |          |                 |    |                                                                                      |
|----------|-----------------|----|--------------------------------------------------------------------------------------|
| _____ 1. | Halt/Load       | a. | This is the most severe form of system initialization that an operator can invoke.   |
| _____ 2. | ??PHL           | b. | This action loads a new MCP to the H/L unit from a system tape.                      |
| _____ 3. | OLAYROW message | c. | Command used to install a new MCP file.                                              |
| _____ 4. | Cold Start      | d. | Command used to invoke a Halt/Load.                                                  |
| _____ 5. | CM              | e. | This action is the least severe form of system initialization.                       |
| _____ 6. | Cool Start      | f. | A Cold Start is indicated by the operator by entering this message to SYSTEM/LOADER. |

**SECTION 5**

**HARDWARE AND I/O OVERVIEW**

# A SERIES AND B 5000/B 6000/B 7000 CONCEPTS HARDWARE AND I/O OVERVIEW

## *INTRODUCTION*

### **Section Objective**

Identify the major hardware components of the A Series and B 5900, B 6900, and B 7900 systems.

### **Purpose**

Familiarity with the hardware components is essential when performing operations, reading reference manuals, and making decisions about hardware configurations.

### **Unit Objectives**

Identify the major hardware components of the A Series and B 5900/B 6900/B 7900 systems

Identify the major hardware components of a specific system, as described in the hardware overview sections.

# **A SERIES AND B 5000/B 6000/B 7000 CONCEPTS HARDWARE AND I/O OVERVIEW**

## ***UNIT 1***

### ***HARDWARE AND I/O OVERVIEW***

#### **Objective**

Identify the major hardware components of the A Series and B 5900, B 6900, B 7900 systems.

#### **Purpose**

Familiarity with the hardware components is essential when performing operations, reading reference manuals, and making decisions about hardware configurations.

#### **Resources**

A Series Systems An Introduction, Section 1 - What's in an A Series System

# A SERIES AND B 5000/B 6000/B 7000 CONCEPTS HARDWARE AND I/O OVERVIEW

## Hardware Components

All A Series and B 5000/B 6000/B 7000 systems include similar hardware components, although they may be called by different names and have different limits on the various models. This unit describes the common hardware components and functions. Details about specific models are provided in Figures 5-2, and in the hardware overview sections.

### Processor

The processor performs all the arithmetic and logic in the system. It includes a series of registers, which are storage locations with designated functions required for executing programs.

**Multiprogramming** allows many programs to share a single processor. A processor may execute only one program at any point in time, but when that program is interrupted (for example, to read from a disk), the processor can execute a portion of another program until it is interrupted again. Multiprogramming improves throughput by keeping the processor busy whenever there are programs ready to execute, as shown in Figure 5-1.

The term **mix** refers to all of the programs that are currently sharing the processor.

Some models may have multiple processors, as described later in this unit.

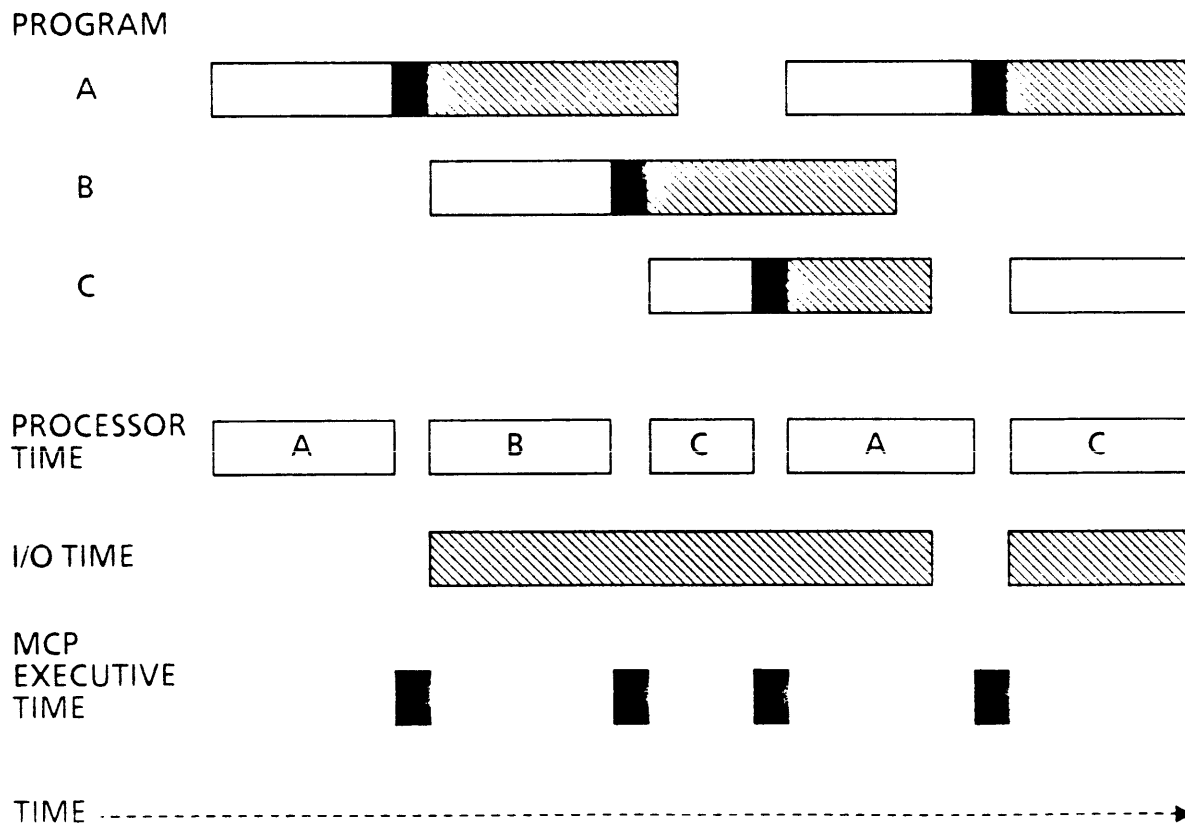


Figure 5-1 Multiprogramming



## A SERIES AND B 5000/B 6000/B 7000 CONCEPTS HARDWARE AND I/O OVERVIEW

### *Maintenance Processor*

The Maintenance Processor is a separate processor required to initialize the system, and to conduct maintenance tests. The Maintenance Processor has different names on different models of systems.

The A Series, B 6900, and B 7900 Maintenance Processors allow for Remote Diagnostics, so that diagnostic programs can be executed against the system by Burroughs personnel located at remote sites. Remote Diagnostics are not available for B 5900 systems.

### *Operator Display Terminal*

The Operator Display Terminal (ODT) is the computer operator's interface to the system. The ODT receives the operator's commands, and displays the system status.

Most A Series models have 2 ODTs to accommodate 2 operators, and to display 2 screens of system status information at once.

Additional ODTs can be configured into the system if necessary.

The term **SPO** (supervisory printed output) is sometimes used to refer to an operator's terminal, although ODT is the preferred term.

The ODT is also used in conjunction with the Maintenance Processor to initialize the system and to perform maintenance functions on most A Series models.

### *Memory*

Memory is temporary storage, which contains data and object programs while they are in use.

Memory is measured in words.

A word is 6 bytes or characters of data.

The maximum amount of memory allowed on specific models is shown in Figures 5-2, and discussed further in the MCP and MCP/AS Overview in Section 7.

**A SERIES AND B 5000/B 6000/B 7000 CONCEPTS  
HARDWARE AND I/O OVERVIEW**

	<b>B 5900</b>	<b>B 6800</b>	<b>B 6900</b>	<b>B 7800</b>	<b>B 7900</b>
<b>MEMORY</b>	Max. 1 M Words	Max. 1 M Words	Max. 1 M Words	Max. 1 M Words  (1M Words per partition)	Max. 16 M Words
<b>INPUT/OUTPUT</b>	UIO B 5920: 2-4 IOBMs B 5930: 2-5 IOBMs	MPX 1 per DP 20 Channels	UIO 2-8 IOBMs	IOM 1-7 depending on the number of CPMs	UIO 4-10 IOBMs
<b>PERIPHERALS</b>	Max. 4095	Max. 255	Max. 4095	Max. 255	Max. 4095
<b>PROCESSORS</b>	CPU 2-4 with Global	DP 2-4 with Global	DP 2-4 with Global	CPM 1-7 depending on the number of CPMs	CPM 1-3
<b>MAINTENANCE PROCESSORS</b>	MP MIP MTS-2	MDP (B 80)	BDU (B 81)	B 7800: MDP B 7700: MDU	AP/AMP (B 5900)
<b>DATA COMM</b>	UIO 2 NSPs	DCP 4 per DP	UIO 4 NSPs	DCP Max. 8  (4 per IOM)	UIO 4 NSPs
<b>ODT'S</b>	MTS-2 B 5920: 1 B 5930: 2	TD830s 2	MTs 2  (Soft Display) (Direct Spo)	TD830s 2	MTS-2 2  (SYCON)
<b>OTHER</b>					HDPs PAC Kit

Figure 5-2a HARDWARE SUMMARY Part 1

**A SERIES AND B 5000/B 6000/B 7000 CONCEPTS  
HARDWARE AND I/O OVERVIEW**

	<b>A 3</b>	<b>A 9</b>	<b>A 10</b>	<b>A 12</b>	<b>A 15</b>
<b>MEMORY</b>	Max. Words D: 2 M E: 2 M F: 4 M K: 8 M	Max. Words B: 1 M D: 2 M F: 4 M	Max. Words D: 4 M F: 8 M H: 16 M	Max. Words 16 M	Max. Words F: 8 M H,I,J: 16 M K,L,M,N: 32 M
<b>INPUT/OUTPUT</b>	UIO D: 1 IOBM E: 2 IOBMs F: 2-4 IOBMs K: 4-6 IOBMs	UIO B: 2 IOBMs D: 4 IOBMs F: 4-6 IOBMs	UIO D: 2-8 IOBMs F: 2-8 IOBMs H: 4-16 IOBMs	UIO 4-16 IOBMs	UIO 4-32 IOBMs
<b>PERIPHERALS</b>	Max. 4095	Max. 4095	Max. 4095	Max. 4095	Max. 4095
<b>PROCESSORS</b>	CPU D: 1 F: 1 K: 2	CPU B: 1 D: 1 F: 1	CPU D: 1 F: 1 H: 2	CPM: 1	CPM F: 1 H,I,J: 2 K,L: 3 M,N: 4
<b>MAINTENANCE PROCESSORS</b>	UIP	MP/MIP ET2000 5 1/4 Floppy 5 1/4 Winch. (2)	MIP ET2000 5 1/4 Floppy 5 1/4 Winch. (2)	SMS II MIP ET2000 5 1/4 Floppy 5 1/4 Winch. (2)	SMP  (Local and Remote Diagnostics)
<b>DATA COMM</b>	UIO D: 4 DC DLPs E: 4 DC DLPs, 1 NSP F: 4 DC DLPs, 1 NSP K: 4 DC DLPs, 2 NSP's	UIO Combination of 1-4 NSPs and/or DC- DLPs	UIO Combination of 1-4 NSPs and/or DC- DLPs	UIO Combination of 1-4 NSPs and/or DC- DLPs	UIO Combination of 1-4 NSPs and/or DC- DLPs
<b>ODT'S</b>	ET2000 1-2	ET2000 2  (Softcon)	ET2000 2	ET2000 ET1100	ET2000 2

Figure 5-2b HARDWARE SUMMARY Part 2

# A SERIES AND B 5000/B 6000/B 7000 CONCEPTS HARDWARE AND I/O OVERVIEW

## *Input/Output Subsystem*

The I/O subsystem controls and communicates with all of the devices attached to the system. All A Series, B 5900, B 6900, and B 7900 systems use Burroughs **Universal I/O (UIO)**.

This form of I/O is called **Universal** because it is also used on V Series and B 2000/B 3000/B 4000 and systems.

Universal I/O also applies to both peripheral and data communications management.

Each peripheral is connected to a **Data Link Processor (DLP)** which is designed to control that specific type of device, as shown in Figure 5-3. DLPs are an example of Function Processors, or microprocessors with specialized functions.

Some peripherals (for example, image printers) can have only 1 device attached to each DLP.

Other peripherals, such as disk pack and tape drives, can have multiple devices attached to a single DLP through a controller.

A series of packs may be connected to multiple controllers and DLPs, to provide multiple **paths** for the system to use in communicating with the packs.

The DLP's are housed in groups called **I/O Base Modules (IOBM)**, or I/O-Data Comm (IODC) base modules.

Each I/O base module can hold a maximum of 8 DLPs, depending on the number of circuit boards in the various DLPs used.

Different hardware models allow different numbers of I/O base modules, as in Figure 5-3.

**Line or Logic Expansion Modules (LEMs)** can be added to increase the maximum number of I/O base modules on a system.

The I/O base modules are connected by **Message Level Interface (MLI)** cables to an I/O processor, which interfaces to main memory and the central system. The I/O processor has different names on different models.

Systems designed before the 900 family and A Series (B 6800s, for example) used **Multiplexors (MPX)** instead of Universal I/O. Some of the manuals make distinctions in operations between UIO and MPX systems.

**A SERIES AND B 5000/B 6000/B 7000 CONCEPTS  
HARDWARE AND I/O OVERVIEW**

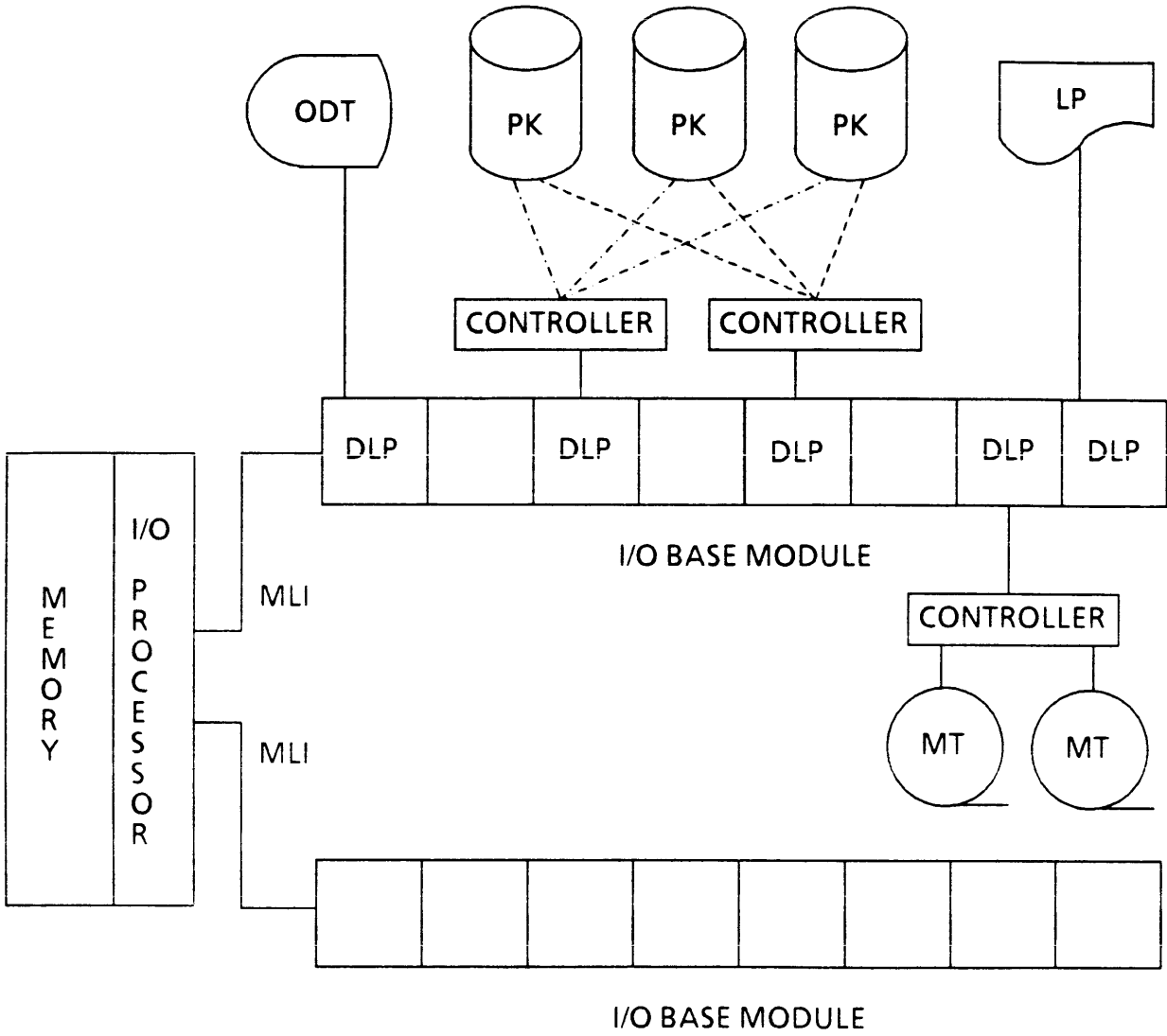


Figure 5-3 Universal I/O

## A SERIES AND B 5000/B 6000/B 7000 CONCEPTS HARDWARE AND I/O OVERVIEW

### *Peripherals*

Many types of peripheral devices can be connected to a system. Each type of device is assigned a mnemonic for use in ODT commands. Some of the more common mnemonics and device types are:

PK	Disk Pack Drive
LP	Line Printer
IP	Image Printer
MT	Magnetic Tape Drive
MTP	Phase-encoded (PE) Tape
SC	System Console (ODT)
CR	Card Reader
CP	Card Punch

The PER ODT command displays information about all devices of a certain type. For example, PER MT inquires into the status of all the magnetic tape drives.

When a device is installed, it is assigned a unique number. Many ODT commands reference peripherals by their device type mnemonics and numbers, such as PK 44 or MT 14.

# A SERIES AND B 5000/B 6000/B 7000 CONCEPTS HARDWARE AND I/O OVERVIEW

## Multiprocessor Systems

Several models of A Series and B 5900/B 6900/B 7900 systems can have multiple Central Processing Units (CPUs). Different models use these multiple processors in different ways, as described later in this unit.

### Multiprocessing

**Multiprocessing** occurs when 2 or more CPUs are under the control of a single MCP. It allows multiple programs to execute at the same time, by executing 1 program in each processor. Multiprocessing also involves multiprogramming in each processor, as shown in Figure 5-4.

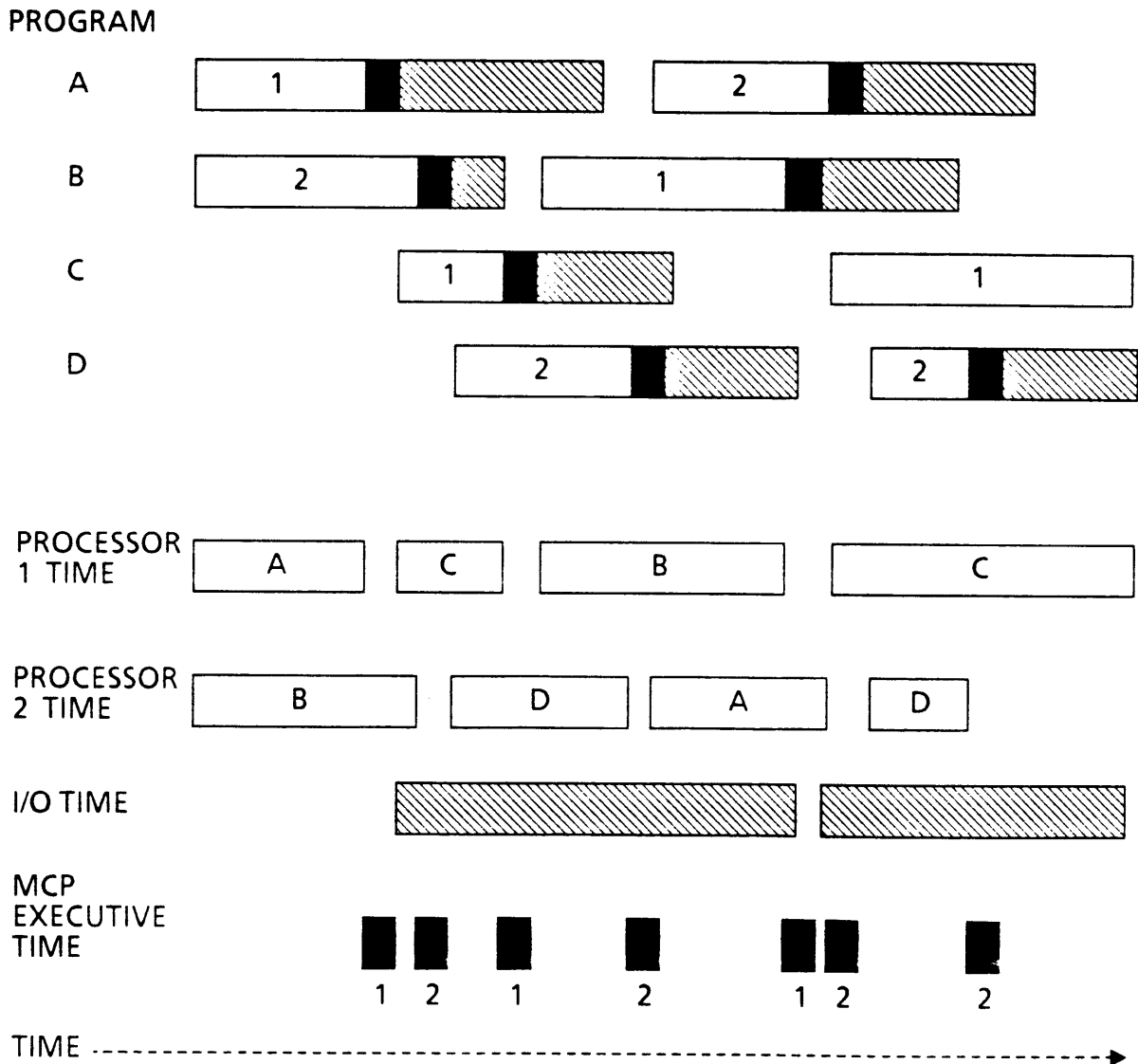


Figure 5-4 Multiprocessing

## A SERIES AND B 5000/B 6000/B 7000 CONCEPTS HARDWARE AND I/O OVERVIEW

### *A Series Multiprocessor Systems*

The A 3 model K includes 2 processors. The system can multiprocess, with both processors controlled by the same MCP, as shown in Figure 5-4. The A 3 K can also be used as a single processor system, but cannot be split into 2 independent single processor systems.

The A 10 model H is a dual processor system, which can be partitioned into 2 single processor systems under 2 separate MCPs. The A 10 can also be a **monolithic** system, a large multiprocessor system with all resources controlled by a single MCP, as in Figure 5-4.

The A 15 can have a maximum of 4 processors, as shown in Figure 5-2. All of the processors can be controlled by 1 MCP, to create a large multiprocessing system. Models J, L, and N can be partitioned into 2 independent systems under 2 separate MCPs, with 1 or 2 processors in each partition.

### *B 5000/B 6000/B 7000 Multiprocessor Systems*

On B 5900, B 6900, and B 6800 systems, 2 to 4 processors can be connected through a special type of memory called Global Memory. These systems can be dynamically configured in 3 ways :

Independent Mode allows each processor to run independently, under a separate copy of the MCP. The Global Memory can be used as an extension of main memory.

In Multiprocessor Mode, the system multiprocesses as in Figure 5-4. This is sometimes called **tightly coupled**, because all the processors and other resources are controlled by 1 MCP, which resides in Global Memory.

Shared Resources Mode allows each processor to execute its own MCP and control its own peripherals, but the processors can communicate with each other through shared portions of Global Memory. This configuration is also called **loosely coupled**.

Multiple Processor B 7900s may be configured as a large multiprocessor system under a single MCP, or partitioned into 2 independent systems under 2 MCPs.



## A SERIES AND B 5000/B 6000/B 7000 CONCEPTS HARDWARE AND I/O OVERVIEW

### *Configuration File*

Configuration files apply to multiprocessor systems that can be split into independent systems under separate MCPs.

A configuration file is a disk file that names and defines the different configurations, or combinations of processors, memory, and peripherals, that are required on the various systems.

The CF (Configuration File) ODT command is used to specify the name of the configuration file.

The RECONFIGURE ODT command allows the operator to specify the name of the configuration that will take effect on the next Halt/Load.

**A SERIES AND B 5000/B 6000/B 7000 CONCEPTS  
HARDWARE AND I/O OVERVIEW**

This page left blank for formatting.

**SECTION 6**

**DATA COMMUNICATIONS CONCEPTS**

## ***INTRODUCTION***

### **Section Objective**

Identify the hardware and software that is the Data Communications Subsystem.

### **Purpose**

Familiarity with the data communications hardware and software components is essential when reading reference manuals, making decisions about the communications network and controlling the network communications environment.

### **Unit Objectives**

Identify the Data Communications Hardware.

Identify the Data Communications Software.

**A SERIES AND B 5000/B 6000/B 7000 CONCEPTS  
DATA COMMUNICATIONS CONCEPTS**

**UNIT 1**

**DATA COMMUNICATIONS HARDWARE**

**Objective**

Identify the Data Communications Hardware.

**Purpose**

You must be familiar with the data communications hardware components in order to read reference manuals, make decisions about the communications network and control the communications environment.

**Resources**

A Series Systems An Introduction, Section 3 - A User's View of System Functions

# A SERIES AND B 5000/B 6000/B 7000 CONCEPTS DATA COMMUNICATIONS CONCEPTS

## What is the Data Communications Subsystem?

Data Communications is the transmission, reception and validation of data.

The Data Communications Subsystem is the communications link between remote devices, such as other systems (hosts) and terminals, and the system. It removes communications functions from the central processor and places these functions in processors specifically designed for data communications.

The Data Communications Subsystem is modular in design and distributes communication functions among a series of functional processors which include the Network Support Processor (NSP), Line Support Processor (LSP), Quad Line Adaptors (QLA), or the Data Communications Data Link Processors (DCDLP).

### *Network Support Processor (NSP)*

The NSP is a data communications subsystem network controller device. It is located in an IODC (I/O-Data Comm) Base Module by itself to allow it to be the only device accessing the system from that IODC base.

The NSP has its own memory, is programmable using the Network Definition Language II (NDL II) software, and is responsible for:

- Controlling the line disciplines for the lines in the network.
- Off-loading the majority of detail data link and line discipline control to the LSP.
- Selecting the proper line discipline to be off-loaded to the LSP.
- Assembling and delivering messages to and from the CPU.
- Informing the LSP of any dynamic network reconfigurations.

### *Line Support Processor (LSP)*

The LSP is a processor that controls the lines attached to it. It provides the connection between the QLA and the NSP.

The LSP has its own memory, can be programmed using NDL II software and is responsible for:

- Controlling each line that transmits or receives data.
- Assembling characters into messages.
- Delivering messages to and from the NSP.

## **A SERIES AND B 5000/B 6000/B 7000 CONCEPTS DATA COMMUNICATIONS CONCEPTS**

### ***Quad Line Adaptor (QLA)***

The QLA is a set of four Line Adaptors controlled by the LSP.

### ***Line Adaptor (LA)***

The LA controls the line that is attached.

The LA has local memory which is used to store translation tables, message buffers, line and station parameters, the line's polling sequence, and the code required to control the communication line and the line discipline.

It is responsible for accumulating bits into characters for the LSP and translating the characters from or to EBCDIC into the format declared in the NDL II software for that line.

### ***Electrical Interface (EI)***

The EI converts the signal voltage level from the system to the voltage level required by the data communications cable. Different types of EIs are available for different types of communications cables, such as TDI (Two-Wire Direct Interface) or RS232.

### ***Data Communications Data Link Processor (DCDLP)***

The DCDLP is a special DLP which contains a special micro-coded control program for handling data communications.

DCDLP is an alternate approach to the NSP/LSP combination.

It is used in place of the NSP, LSP and QLA. The DCDLP has standard protocols preprogrammed into it to handle the functions of the NSP, LSP, and QLA. This provides for speed and lower cost but lacks the flexibility and programmability of the NSP/LSP combination.

The system will identify and treat the DCDLP as an NSP.

A SERIES AND B 5000/B 6000/B 7000 CONCEPTS  
 DATA COMMUNICATIONS CONCEPTS

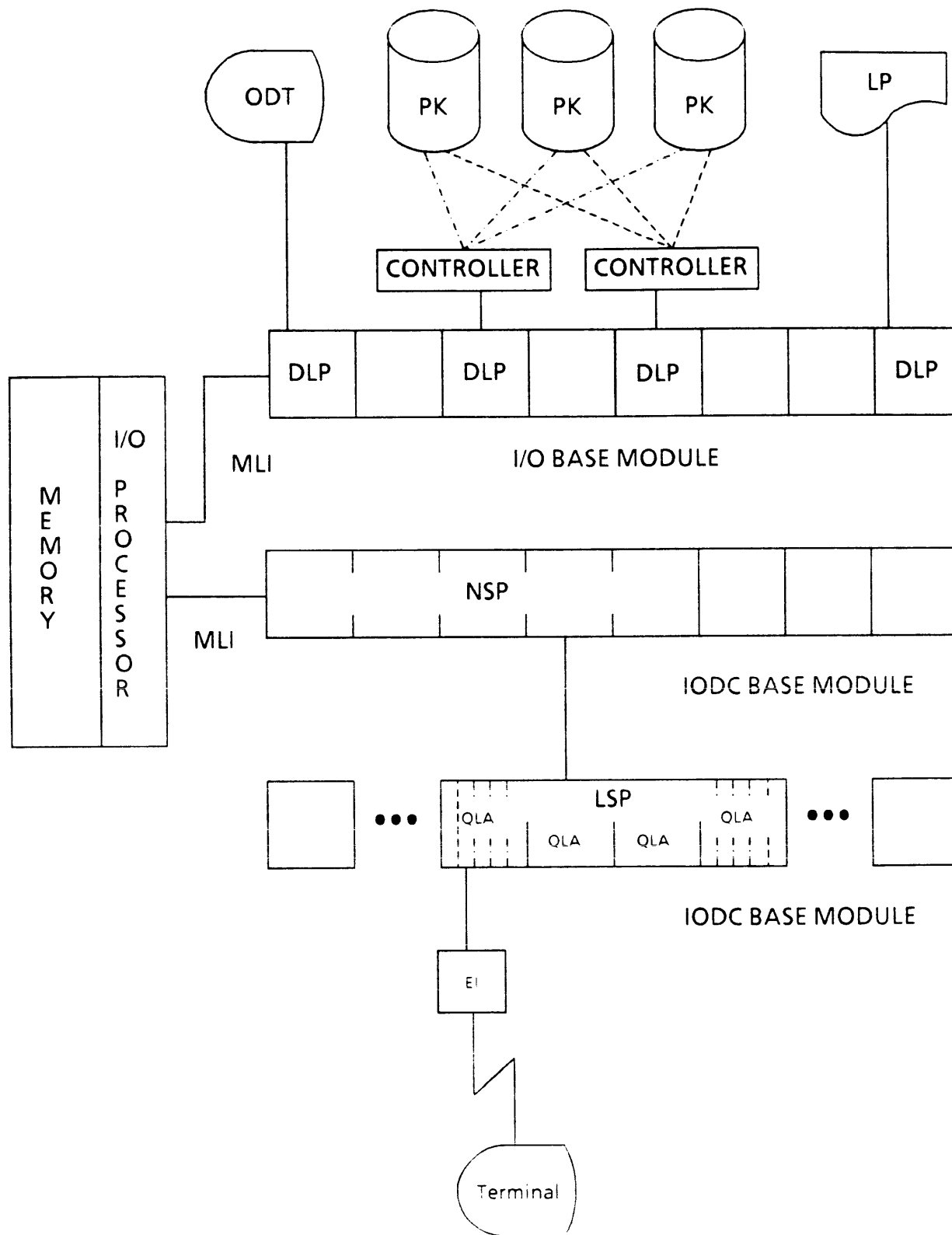


Figure 6-1 Universal I/O Including Data Communications NSP/LSP



## A SERIES AND B 5000/B 6000/B 7000 CONCEPTS DATA COMMUNICATIONS CONCEPTS

### Data Flow Using the NSP/LSP Approach

The following steps transfer a message from the terminal to the host and returns the response back to the terminal. The data flow is to demonstrate the use of the data communications hardware and does not include the data communications software necessary. For details on the data communications software, see unit 2.

1. A message, entered by the operator, is translated by the terminal into the correct bit patterns and sent out across the communications lines.
2. The bits pass through the EI and are received by the QLA.
3. The QLA assembles the bits into a character. Translation is done here on each character into EBCDIC format. Once a character is assembled it is passed on to the LSP and the QLA moves on to the next group of bits.
4. The LSP receives characters from the QLA one at a time. The LSP assembles those characters into a message which is then passed to the NSP.
5. The NSP receives the message from the LSP and performs any editing necessary (as declared in the NDL) and passes the message to the Host.
6. The Host will now process the message, formulate a response and pass that response to the NSP.
7. The NSP will edit the response and pass the message to the LSP.
8. The LSP will disassemble the message into characters and send 1 character at a time to the QLA.
9. The QLA will translate and disassemble each character into bits and send the bits out to the terminal.
10. The bits will pass through the EI and be received by the terminal, translated and displayed for the operator.

**A SERIES AND B 5000/B 6000/B 7000 CONCEPTS  
DATA COMMUNICATIONS CONCEPTS**

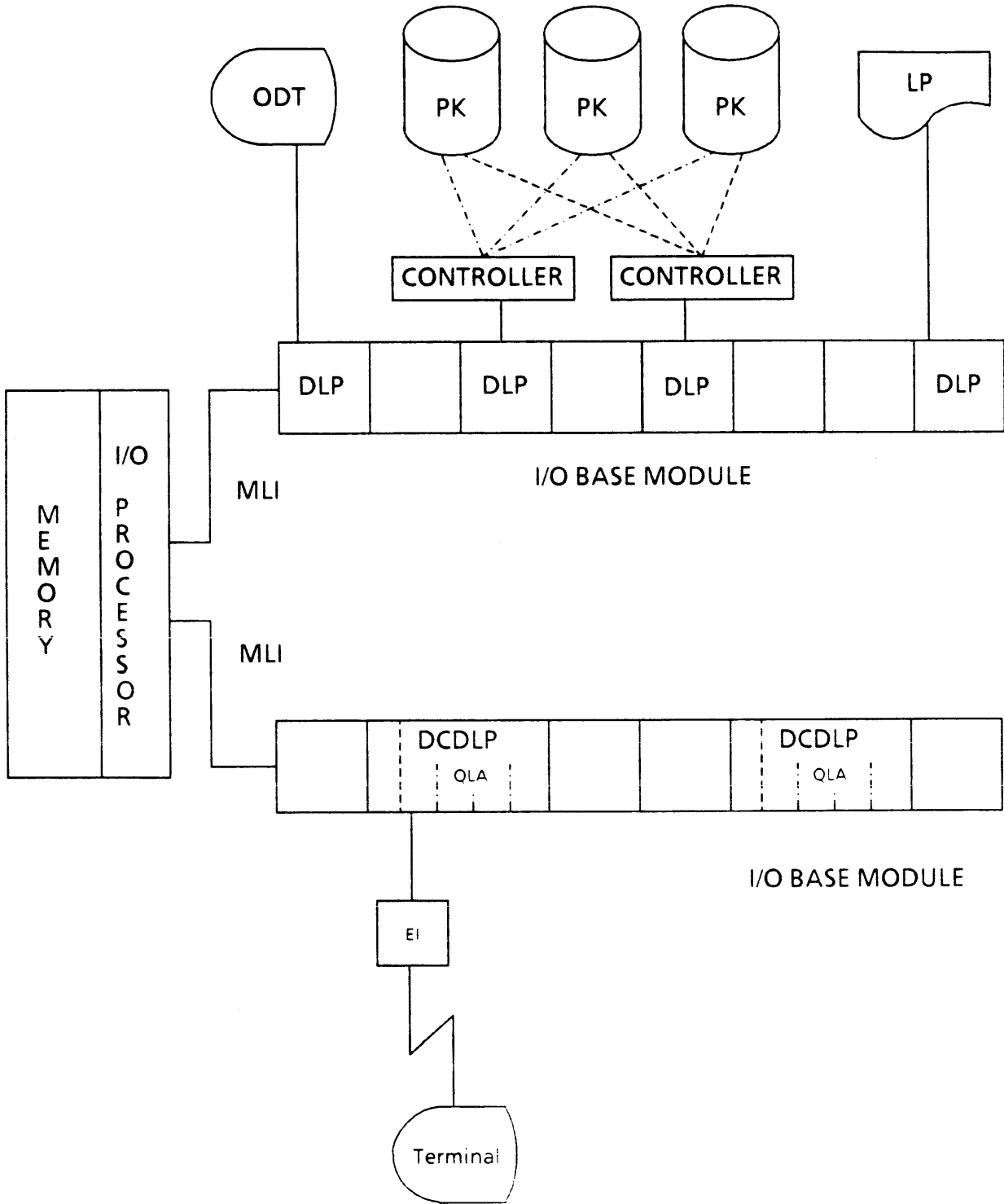


Figure 6-2 Universal I/O Including Data Communications DCDLP

# A SERIES AND B 5000/B 6000/B 7000 CONCEPTS

## DATA COMMUNICATIONS CONCEPTS

### Data Flow Using the DCDLP Approach

The following steps transfer a message from the terminal to the host and returns the response back to the terminal. The data flow is to demonstrate the use of the data communications hardware and does not include the data communications software necessary. For details on the data communications software, see unit 2.

1. A message, entered by the operator, is translated by the terminal into the correct bit patterns and sent out across the communications lines.
2. The bits pass through the EI and are received by the QLA.
3. The QLA assembles the bits into a character. Translation is done here on each character into EBCDIC format. Once a character is assembled it is passed on to the DCDLP.
4. The DCDLP receives characters from the QLA one at a time. The characters are assembled into a message, edited if needed and passed to the Host.
5. The Host will now process the message, formulate a response and pass that response to the DCDLP.
6. The DCDLP will edit and disassemble the message into characters and send 1 character at a time to the QLA.
7. The QLA will translate and disassemble each character into bits and send the bits out to the terminal.
8. The bits will pass through the EI and be received by the terminal, translated and displayed for the operator.

**A SERIES AND B 5000/B 6000/B 7000 CONCEPTS  
DATA COMMUNICATIONS CONCEPTS**

**UNIT 2**

**DATA COMMUNICATIONS SOFTWARE**

**Objective**

Identify the Data Communications Software.

**Purpose**

You must be familiar with the data communications software components in order to read reference manuals, make decisions about the communications network and control the communications environment.

**Resources**

A Series Systems An Introduction Section 3 - A User's View of System Functions

A Series Communications Management System Capabilities Manual

A Series Interactive Datacomm Configurator User's Guide

# A SERIES AND B 5000/B 6000/B 7000 CONCEPTS DATA COMMUNICATIONS SOFTWARE

## Software Components

The data communications system software components consist of the following:

### Network Information File (NIF)

A description of the data communications network is written in the Network Definition Language (NDL II). The NDL II source is compiled using the NDL II Compiler. The NDL II Compiler produces the NIF. The NIF contains tables and code generated from the NDL II source.

### DATAKOMINFO File

The NIF is used to generate the DATAKOMINFO file which is used by the data communication subsystem to define the network and how the network is to be controlled.

### Data Communications Controller (DCC)

The DCC is part of the Master Control Program (MCP) and executes in the CPU. It provides the MCS and Application Program interface, and initiates and performs network control functions.

### Message Control System (MCS)

An MCS is a program written in DCALGOL that controls the flow of messages between terminals, application programs, and the operating system.

The MCS is responsible for controlling security (logging the user on), routing of messages to the proper location, handling recovery of the network, performing control commands, and performing other functions required or desired by the MCS designers. The MCS is often referred to as the traffic cop for the Data Communications Subsystem.

### Application Program

This is the program that is written by the user to fulfill a particular data processing need. It is responsible for editing data, building or updating databases and data files, and screen formatting.

## **A SERIES AND B 5000/B 6000/B 7000 CONCEPTS DATA COMMUNICATIONS SOFTWARE**

### ***Software Data Flow***

The following steps take a message from the terminal to the application program and brings the response back to the terminal from a software viewpoint only.

1. A message, entered by the operator is translated by the terminal into the correct bit patterns and sent out across the communications line.
2. The message is received by the system and is passed to the NSP or DCDLP. For detail steps on how the message reaches the NSP or DCDLP, see unit 1 under Data Flow Using the NSP/LSP Approach or Data Flow Using the DCDLP Approach.
3. The NSP or DCDLP passes the message to the Host. The DCC, which is part of the MCP receives the message from the NSP or DCDLP, processes it and passes the message to the MCS.
4. The MCS receives the message, processes it and passes it on to the application program.
5. The Application Program receives the message, updates databases and performs any other processing required, formats a response, and passes it to the MCS.
6. The MCS receives the response from the Application Program and passes it on to the DCC.
7. The DCC receives the response and passes it to the NSP or DCDLP.
8. The NSP/LSP or DCDLP follows the steps as defined in unit 1 and the response is sent to the terminal.
9. The terminal receives the response and displays it for the operator.

**A SERIES AND B 5000/B 6000/B 7000 CONCEPTS  
DATA COMMUNICATIONS SOFTWARE**

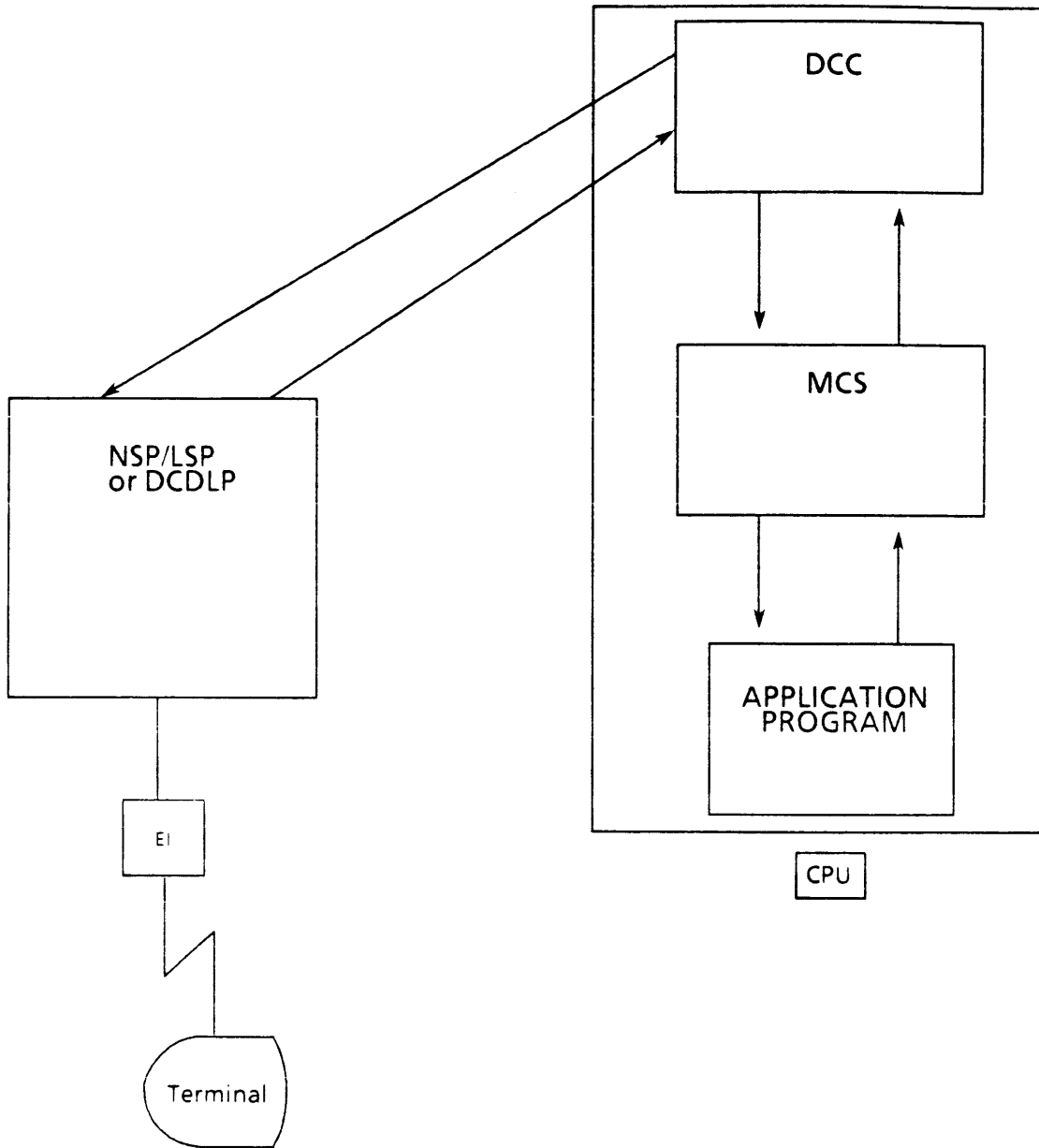


Figure 6-3 Software Data Flow

## A SERIES AND B 5000/B 6000/B 7000 CONCEPTS DATA COMMUNICATIONS SOFTWARE

### *MCS Products*

#### Communications Management System (COMS)

COMS is an MCS and one of the InterPro products. It is a high performance transaction processing program.

COMS is modular in design. It allows you to select the processing features you need and to use your own internally developed processing features. COMS is available in 3 versions: COMS Kernel, COMS (ENTRY) and COMS (Full).

A portion of COMS is integrated with the MCP for performance and handles the log-on control, station-to-station routing, non-participation, error handling, network control and multiple views.

COMS provides extended transaction processing by offering extensive standard functions, using libraries for efficiency, and accommodating user-written modules.

COMS can route transactions to different application programs, depending on a transaction code (transaction based routing).

COMS provides a window feature that enables you to operate a number of program environments independently and simultaneously at a station. COMS allows for transaction routing windows, remote file windows, MCS windows, and command windows. For more details on windows, see Section 8 Unit 5 - COMS Windows and Dialogs.

COMS supports a simplified header for interfacing with COBOL 74 application programs.

COMS is designed to provide a continuous operating environment. It has a provision to perform testing on-line and dynamic installation of new or updated modules. Synchronized recovery for DMS II databases is also available.

COMS has a special support library interface to support current GEMCOS users.

Diagnostic aids are available in COMS.

#### Generalized Message Control System (GEMCOS)

GEMCOS is an MCS that provides the standard features that all MCSs must provide. It has the ability to take options provided by the user and customize the standard MCS features, providing a tailored MCS which meets the specific transaction processing requirements for the installation.

GEMCOS has a synchronized recovery feature which interfaces with the DMS II (Database Management System) software. GEMCOS has its own screen formatter which allows the user to design screens on the terminal.



## A SERIES AND B 5000/B 6000/B 7000 CONCEPTS DATA COMMUNICATIONS SOFTWARE

### Remote Job Entry (RJE)

RJE provides users with simple access to distributed resources, including remote database access, file transfer, remote task initiation and control. A network consisting of a variety of Burroughs computers may be connected. The users of this network can access any system in the network and be either a Host or User to another member of the network.

RJE has three major functions.

#### Remote Job Entry

This feature allows a user who is part of the network to enter an input request at the user's location and have the input request processed by the host system. The output from the request is directed back to the user's system. This allows the user of a system to use the resources of another system in the network to do some of the workload.

#### File Transfer

Files can be transferred to/from disk and magnetic tape devices. This allows the user to transfer a file located on a local system to the host system or to have a file on the host system transferred to the user system.

#### Virtual Terminals

Virtual terminals is a logical concept based on the actual message path established between the terminal, ODT, or application program in the user system. The virtual terminal can control jobs in the host system through the capabilities of the host system MCS. The host system treats this terminal as if it were one of its own terminals instead of another system in the network. The virtual terminal has access to all user and host facilities.

There are 2 different products that correspond to this on CMS and B 1000 systems: RJE provides remote job entry and control, and SYCOM provides the other capabilities. The A Series has 1 product, usually called just RJE, which communicates with RJE and SYCOM.

## A SERIES AND B 5000/B 6000/B 7000 CONCEPTS DATA COMMUNICATIONS SOFTWARE

### *Interactive Datacomm Configurator (IDC)*

IDC is one of the InterPro Products. IDC is an interactive, menu-driven utility that allows you to easily create, interrogate, and modify datacomm network configurations from the ODT or from a datacomm terminal. IDC allows the logical representation of a configuration to be created quickly and efficiently, with a minimum amount of specialized knowledge. The configuration file is known as the **DATACOMINFO** file.

IDC uses the existing configuration file or configuration file sent on the release tape as a beginning for your network. Changes are then made to this configuration file. You can build your network by example. Examples and help information are available on-line. The need for specialized training in the network area is greatly reduced.

IDC will create and maintain your network definition as well as recovery of itself if a failure occurs.

IDC can improve the network uptime because modifications can be made on-line. Input is verified when it is entered, thus identifying errors immediately and allowing for immediate correction. IDC will implement modifications when you indicate that all changes have been entered.

### *Data Communications Subsystem Initialization/Termination*

The Data Communications Subsystem is initialized and terminated by the ID (Initialize Datacomm) ODT command.

The NIF and DATACOMINFO files are used to initialize the NSP/LSP or DCDLP. Once the Data Communications Subsystem is initialized, the MCSs and Application Programs will be executed as needed.

Example:

ID 108	%Initialize Data Comm
ID:QUIT	%Terminate Data Comm

**A SERIES AND B 5000/B 6000/B 7000 CONCEPTS  
DATA COMMUNICATIONS SOFTWARE**

**Practice**

Match the terms on the left with the descriptions on the right.

- |           |                   |    |                                                                                                                                                                             |
|-----------|-------------------|----|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| _____ 1.  | QLA               | a. | A part of the MCP that controls the physical movement of information between the central system and the Data Communications Subsystem.                                      |
| _____ 2.  | NSP               | b. | Accepts keystrokes and converts them into bits and transmits them to the system.                                                                                            |
| _____ 3.  | LSP               | c. | The source program that describes the Data Comm network in terms of station names, terminal types, line speeds and line procedures.                                         |
| _____ 4.  | MT/TD/ET          | d. | On input, assembles bits into bytes; on output, disassembles bytes into bits.                                                                                               |
| _____ 5.  | DCC               | e. | Transmits whole messages between itself and the CPU.                                                                                                                        |
| _____ 6.  | NDL II            | f. | This program is an MCS which provides such features as extensive transaction processing, windows, and a continuous operating environment.                                   |
| _____ 7.  | MCSs              | g. | Receives characters from the QLA and sends messages to the NSP.                                                                                                             |
| _____ 8.  | NDL II Compiler   | h. | The DCALGOL programs that exist for the purpose of controlling stations (traffic cop), and provide an interface between the Data Comm network and the application programs. |
| _____ 9.  | DATA COMINFO FILE | i. | This program compiles the NDL II source and produces the code for the NSP as well as tables used by the central system and the Data Communications Subsystem.               |
| _____ 10. | DCDLP             | j. | The Data Communications Subsystem is initialized by this command.                                                                                                           |

Continued on next page.

**A SERIES AND B 5000/B 6000/B 7000 CONCEPTS  
DATA COMMUNICATIONS SOFTWARE**

- \_\_\_\_\_ 11. COMS
  - k. The special DLP used for data communications instead of an NSP/LSP.
  
- \_\_\_\_\_ 12. ID command
  - l. The Data Communications Configuration file.

**SECTION 7**

**MCP AND MCPIAS OVERVIEW**

# A SERIES AND B 5000/B 6000/B 7000 CONCEPTS MCP AND MCP/AS OVERVIEW

## ***INTRODUCTION***

### **Section Objective**

Identify the MCP and MCP/AS and their functions.

### **Purpose**

You should be aware of the functions of the MCP and MCP/AS before deciding which operating system to use on your system.

### **Unit Objectives**

Identify the MCP and MCP/AS.

Explain the ASN and ASD addressing techniques.

Identify the functional areas of the MCP and MCP/AS.

**A SERIES AND B 5000/B 6000/B 7000 CONCEPTS  
MCP AND MCP/AS OVERVIEW**

**UNIT 1**

***MCP AND MCPIAS DIFFERENCES***

**Objective**

Identify the MCP and MCP/AS.

Explain the ASN and ASD addressing techniques.

**Purpose**

You should be aware of the differences between MCP and MCP/AS before deciding which operating system to use for your system.

**Resources**

# A SERIES AND B 5000/B 6000/B 7000 CONCEPTS MCP AND MCP/AS DIFFERENCES

## MCP

The Master Control Program (MCP), which is the standard A Series (A/B 5000/B 6000/B 7000) operating system, provides complete management of all system resources and tasks automatically. Hardware specifications indicate a system's power potential, whereas system software determines how much of that power is usable.

The MCP has ultimate responsibility for:

- Assigning memory.
- Managing input/output functions.
- Communicating with the operator.
- Logging system use.
- Automatically loading programs.
- Maintaining a library of all files.
- Supervising numerous other functions.

The MCP makes optimum use of all system resources.

It automatically allocates system resources to meet the needs of executing programs.

It continually and automatically assigns resources, initiates jobs, and monitors their performance in a multiprogramming environment.



## A SERIES AND B 5000/B 6000/B 7000 CONCEPTS MCP AND MCP/AS DIFFERENCES

### *MCP ASN Memory Management Addressing*

The basic concept of the MCP Address Space Number (ASN) memory management is the Address Space (AS). An **address space** is the memory that a program may access at any one time. The system's memory is divided into some number of address spaces, each of which contains a maximum of 1 million words (6MB) of memory and is assigned a unique number (ASN). The B 7000/A 15 systems have a maximum of 2 million words (12MB) of memory per address space.

The MCP is responsible for controlling the entire system's memory and must be available for all ASNs on the system. To provide for this, a portion of memory is assigned as being common to all address spaces so that programs in different address spaces can share code and data. The common portion is known as the **shared component** of the address space or memory environment. The non-shared portion is known as the **local component** of the address space.

A system can have as many local components as needed or as the hardware permits, but only one shared component per system is permitted.

The combination of the shared component plus the local components make up 1 address space of 1 million words maximum, (see Figure 7-1). On the B 7000/A 15, an address space of 2 million words is permitted, and is composed of the shared component, 1 local component for program code only and 1 local component for program data only, (see Figure 7-2).

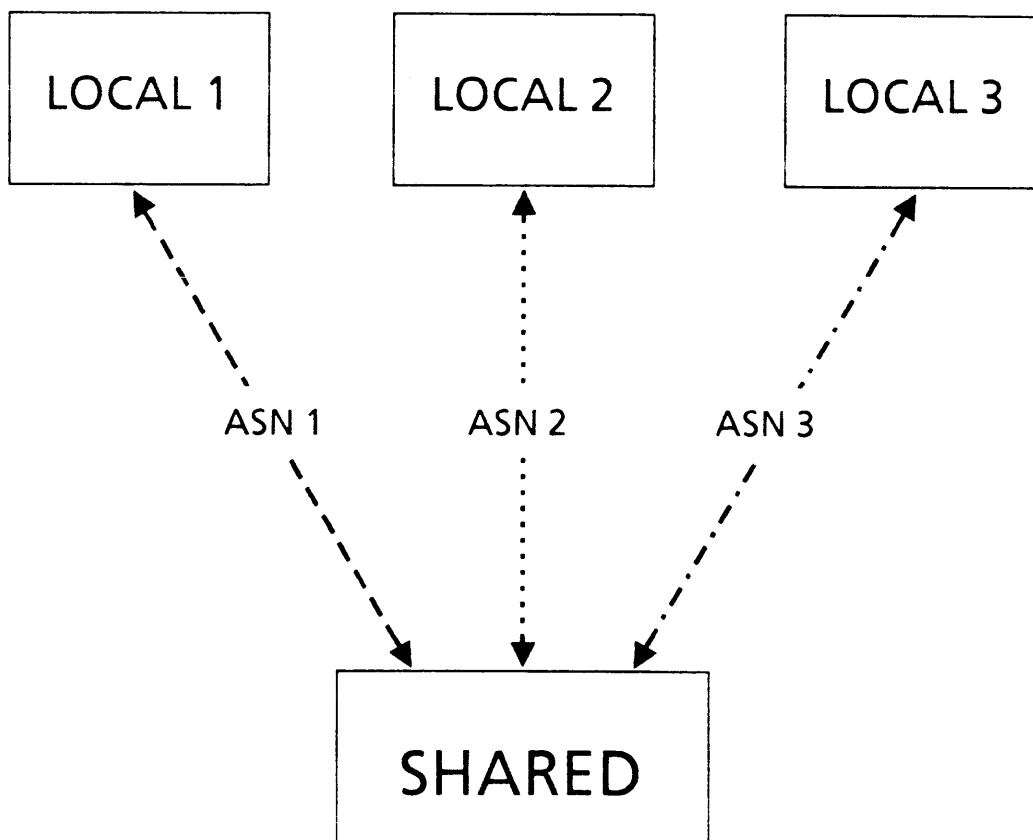


Figure 7-1 ASN Memory Structure - 1 MWs

**A SERIES AND B 5000/B 6000/B 7000 CONCEPTS  
MCP AND MCP/AS DIFFERENCES**

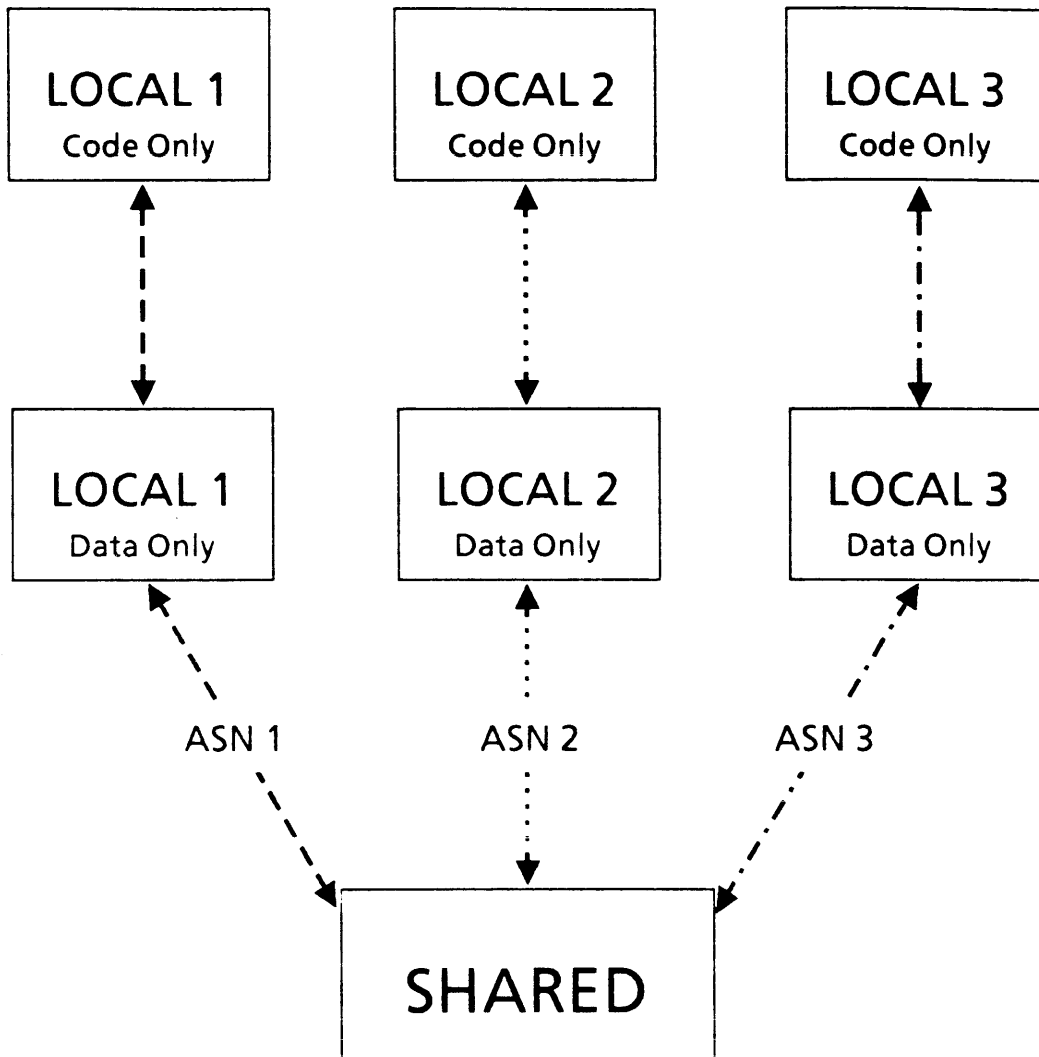


Figure 7-2 ASN Memory Structure - 2MWs

One or more ASNs form a **Subsystem**, which is a logical division of physical memory. A user can specify a subsystem in which a task is to execute.

ASN uses 20 bit words for addressing the memory space.

The ASs or subsystems are created and maintained by the ODT commands, MS (Make Subsystem) and EC (Environment Component).

The MS command establishes what subsystems are to exist on the system.

The EC command sets the sizes of each component in the different subsystems

## **A SERIES AND B 5000/B 6000/B 7000 CONCEPTS MCP AND MCP/AS DIFFERENCES**

### **ASN Advantages:**

Large amounts of memory can be configured on the system and all programs are visible to any processor or I/O processor.

Additional memory and program visibility significantly improves system performance.

Component sizes can be reconfigured to tailor the system to the needs of the user.

A memory environment can be dedicated to a specified collection of tasks.

### **ASN Disadvantages:**

Workload is generally required to be balanced among the local components.

Programs are not visible to each other unless they are in the same ASN, or in the shared component.

Sharing of data is limited to the ASN.

Space contention can occur in the shared component.

## **MCP/AS**

Master Control Program/Advanced Systems (MCP/AS) is a new generation of operating system for the A Series Machines only (it does not apply to B 5000/B 6000/B 7000s). MCP/AS was specifically designed to complement the innovative architecture and expanded memory capability of the A series machines.

The standard MCP is still released and supported with all systems, since MCP/AS is a chargeable product, and on the A 3 and A 15, special hardware is required.

MCP/AS offers all of the essential features of the standard MCP plus the expanded memory addressing feature. The standard MCP is available for the A machines but MCP/AS will be the operating system of choice for most A Series machines with more than 1 million words of memory.

## **A SERIES AND B 5000/B 6000/B 7000 CONCEPTS MCP AND MCP/AS DIFFERENCES**

MCP/AS looks at all memory as one monolithic area, all of which can be directly addressed. This provides a number of important benefits:

The memory addressing capacity is expanded to 96MB.

MCP/AS supports programs of any size, limited only by availability of hardware resources.

MCP/AS is totally transparent to the operator/programmer.

Special configuring of the system is not required.

Existing application programs can be used without changes or recompiling.

All programs are visible to each other.

System performance is improved by a reduction in "Stack Searching".

### ***MCP/AS ASD Memory Management Addressing***

MCP/AS features a new form of Memory Management called Actual Segment Descriptor (ASD), which expands the capacity, performance and memory addressability of the A machines. MCP/AS ASD removes the memory limitations that exist using ASNs, by changing the way that the operating system views physical memory and the way the processor executes certain instructions.

ASD uses a structure maintained by the MCP/AS called the **ASD Table**, which handles all allocated memory areas that have been accessed. The ASD table contains ASDs for each piece of data or code that has been in memory during the life of a program. An ASD is an actual segment descriptor which is used to point to the location of a data or code item in memory or on disk. An ASD Number is used as an index into the ASD Table.

All code and data segments are accessed through the ASD table to find the memory address or the disk address of the segment, (see Figure 7-3). Memory searching to update memory or disk addresses of information for all affected programs is reduced by using the ASD table.

ASD uses 32 bit words for addressing memory.

The ASD Table is initially allocated a size proportional to the amount of memory on the system at Halt/Load time. The ASD (Actual Segment Descriptor) ODT command is used to display the size of the table and the maximum number of ASD entries used. The ASD command can also be used to change the size of the ASD Table to tailor it to the installation's needs. The new size takes effect on the next Halt/Load.

A SERIES AND B 5000/B 6000/B 7000 CONCEPTS  
MCP AND MCP/AS DIFFERENCES

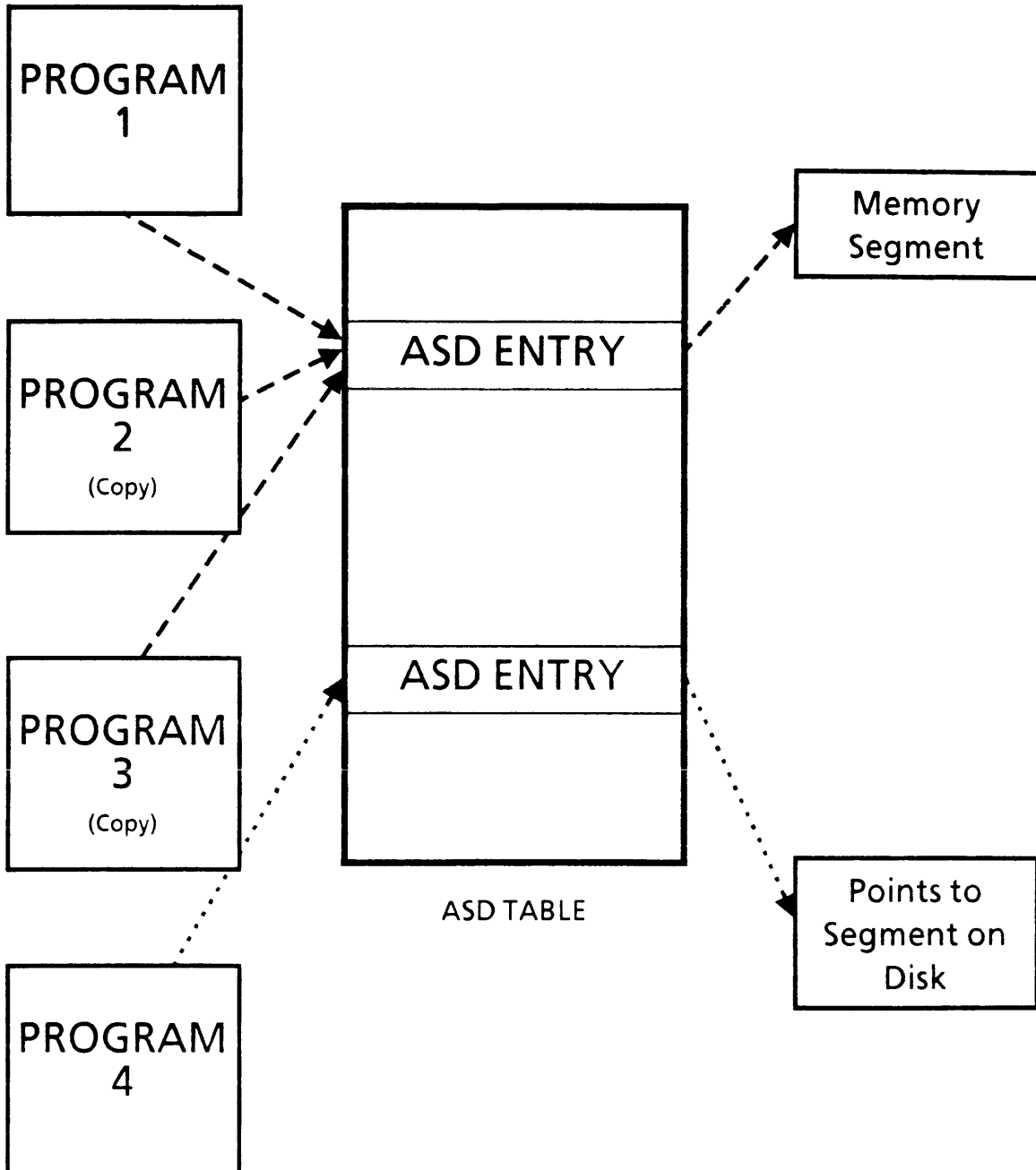


Figure 7-3 ASD TABLE

**A SERIES AND B 5000/B 6000/B 7000 CONCEPTS  
MCP AND MCP/AS OVERVIEW**

**UNIT 2**

***MCP MCPIAS FUNCTIONAL AREAS***

**Objective**

Identify the functional areas of the MCP and MCP/AS.

**Purpose**

In order to understand the A Series systems, you should be aware of what functions the MCP and MCP/AS are providing.

**Resources**

## A SERIES AND B 5000/B 6000/B 7000 CONCEPTS MCP MCP/AS FUNCTIONAL AREAS

### Functional Areas

The MCP (MCP and MCP/AS) can be divided into functional areas. By looking at each functional area, you can determine what the operating system is doing for you.

### *Independent Runners*

**Independent Runners** are procedures or modules of the MCP that execute at the same time with other procedures or modules. Independent Runners are classified as either visible or invisible independent runners. **Visible** independent runners will appear in the mix. **Invisible** independent runners will not appear in the mix when they are executing.

There are 3 invisible independent runners that must always run for the system to operate. They are:

#### ETERNALIR

This independent runner is responsible for looking for tasks to fire up and for checking the peripherals for changes in their ready status.

#### ANABOLISM

This independent runner is responsible for firing up independent runners when needed.

#### CONTROLLER

This independent runner is responsible for handling the operator communications. It is the main driver for system communications.

### *Work Flow*

This area controls the Work Flow Language, which is a language used to control and monitor the execution and flow of jobs and tasks in the system.

A **Task** is a single, complete unit of work performed by the system. An execution of a single program is a task.

A **Job** is one or more related tasks that are grouped together.

The Work Flow Compiler, Controller and Jobformatter are involved in the Work Flow functional area. More details will be covered in Section 13 - Work Flow Language.

# A SERIES AND B 5000/B 6000/B 7000 CONCEPTS

## MCP MCP/AS FUNCTIONAL AREAS

### *Memory Management*

The memory management function of the MCP is responsible for allocation and deallocation of memory. The system supports three methods of memory management, which are different implementations of using disk pack as a backup storage device for memory. The different methods of memory management will be discussed in Section 12 - Memory Management Overview.

### *Operator Interface*

This functional area is responsible for accepting input messages and sending output messages to and from the operator via the system's ODTs. Some operator input messages will be passed on to the Work Flow area and some output messages are sent to the printer.

The independent runner CONTROLLER is responsible for handling messages to and from the ODTs.

### *Basic Utilities*

A utility is a program used for general support of the processes of the system. This area handles the basic utilities such as LIBRARY/MAINTENANCE, which allows the user to copy files between media. LIBRARY/MAINTENANCE is a visible independent runner.

### *Process Control*

This area controls the start-up and shut-down of jobs and tasks. A determination is made of which programs can run and when they can run.

This area is also responsible for making system log entries, which maintains the history and status of the system. Recognizing changes in peripheral device status, existence of new files, new jobs, and new tasks is part of this process control area.



## **A SERIES AND B 5000/B 6000/B 7000 CONCEPTS MCP MCP/AS FUNCTIONAL AREAS**

### ***Peripheral Control***

This area controls all non-random type devices. The responsibilities include:

Locating input data files.

Assigning output devices to programs.

The program needs only to request the type of output device needed, and an available unit will be assigned. When a device is no longer required by the program, it becomes available for use by another program. This allows for dynamic resource management and true device independence.

Reading labels.

Automatic Volume Recognition (AVR). When a device is placed on-line, the system will attempt to recognize the device by reading the device label prior to any operator intervention.

Maintaining a table of available units.

A table of devices that are available is maintained so that the devices can be assigned to programs when requested. This table is used to provide the response to the PER ODT command.

Handling the MCP logs, retries, and errors for each unit.

### ***Disk Management***

The disk management area is responsible for:

Disk initialization.

Verifying the integrity of files on disk, establishing the disk available tables, and reconstructing the disk directories are handled under disk initialization.

Disk allocation.

Available space is assigned to new output files and directory entries are made by the disk allocation process. Included in this area is the controlling of family names (creation and changes) and the updating of the available table.

File security.

This area handles the security restrictions designed by the user to control which tasks will be allowed read and /or write access to data files on disk.

## **A SERIES AND B 5000/B 6000/B 7000 CONCEPTS MCP MCP/AS FUNCTIONAL AREAS**

### ***Data Management***

Data management performs the interface between the users' programs and the users' databases. The special I/O requirements of DMS II (Data Management System II) are handled in the data management functional area.

### ***Data Communications***

This function of the MCP provides the interface between user programs, MCSs and the Data Communications Subsystem. The allocation and deallocation of data communications queues and the data communications tables is also controlled by the DCC.

### ***MCP I/O***

The MCP I/O area controls I/O functions such as:

- Initiating physical I/O for the user.
- Performing I/O functions for the MCP itself.
- Building control words needed to do the physical I/O.
- Handling I/O finishes.
- Handling physical I/O errors.

These I/O functions are reentrant and used by all tasks. MCP I/O is often referred to as Physical I/O.

### ***User I/O***

This functional area handles the user-defined aspects of data handling. This includes opening and closing of files and the checking or modifying of file attributes. Special character translation is controlled here. User I/O is sometimes known as Logical I/O.

## A SERIES AND B 5000/B 6000/B 7000 CONCEPTS MCP MCP/AS FUNCTIONAL AREAS

### *Diagnostics*

The diagnostics area provides the ability to generate dumps.

Memory dumps of programs can be obtained at the request of the operator or by the program itself.

Full system memory dumps can be generated.

### *Peripheral Test Driver (PTD)*

The PTD area is used by field engineering to check out any peripheral device on the system. It involves MTRs (Maintenance Test Routines) to handle maintenance and confidence testing.

### *Sort*

The sort verb in compilers calls the sort MCP routine. The compilers generate the code to allow the user to sort files using a combination of disk, magnetic tape and memory to accomplish the sort. The combination of resources used by any given sorting task is totally under the control of the programmer.

## A SERIES AND B 5000/B 6000/B 7000 CONCEPTS MCP MCP/AS FUNCTIONAL AREAS

### *Interrupt Handling*

This area handles the interrupts that are generated by the system. The A Series systems are interrupt driven systems. The system performs a function until an interrupt is detected, at which time the interrupt is handled, and the function request by the interrupt is done until another interrupt is detected.

The interrupt handling controls are used to interface with the resources handling controls of the MCP.

There are many interrupts used by the system to make it function. There are 3 types of interrupts of which you should be.

#### Data Processor caused:

1. Interval Timer, may cause the MCP to switch to a different task.
2. Presence Bit, causes a request to memory management modules to load data into memory from disk.
3. Invalid Operand, usually due to bad object code.
4. Invalid Index, due to user program indexing error.
5. Stack Overflow, due to user program needing more working area.

#### Input/Output caused:

1. Status Change, when a peripheral unit changes status.
2. I/O Finished, a previously requested I/O is completed.

#### Software caused:

1. Allows one task to interrupt another task of the same job.

**SECTION 8**  
**BASIC CANDE COMMANDS**

# A SERIES AND B 5000/B 6000/B 7000 CONCEPTS BASIC CANDE COMMANDS

## *INTRODUCTION*

### **Section Objective**

Identify CANDE commands and files.

### **Purpose**

In order to use the CANDE MCS, you must be familiar with CANDE commands and CANDE files.

### **Unit Objectives**

Identify CANDE files.

Explain the basic CANDE editing commands.

Explain basic CANDE control commands.

Explain additional CANDE commands.

Explain windows and dialogs.

**A SERIES AND B 5000/B 6000/B 7000 CONCEPTS  
BASIC CANDE COMMANDS**

**UNIT 1**

**CANDE OVERVIEW**

**Objective**

Identify CANDE files.

**Purpose**

In order to use CANDE efficiently, you should be aware of the files that CANDE uses.

**Resources**

A Series CANDE Operations, Section 2 - General Information

A Series CANDE Reference, Section 2 - General Information

# A SERIES AND B 5000/B 6000/B 7000 CONCEPTS CANDE OVERVIEW

## CANDE

The Command AND Edit (CANDE) language is an MCS, written in DCALGOL, that provides normal MCS capabilities such as message routing, security, inquiries and program execution capabilities and text editor capabilities such as generalized file preparation and updating capabilities. Programs are executed in an interactive, terminal-oriented environment.

CANDE allows the users to:

- Create and prepare data and program files.

- Compile and execute programs.

- Edit and maintain files.

- Control access to the system and to files.

- Obtain information about jobs, the network, and other terminals.

- Dynamically alter the system to meet new requirements.

All of these functions are accessed through CANDE commands, which are discussed in the following units.

### *CANDE File Structure*

CANDE will create a workfile that is a copy of the source file you are going to edit. The workfile is entirely separate from the original source file. This allows you to update and test the changes before changing the original file.

CANDE maintains a tankfile called TANKFILE/SYSTEM/CANDE for Halt/Load recovery purposes. The tankfile contains CANDE option settings, configuration information, workfile recovery information, and commands entered by all users to edit their workfiles.

Certain CANDE commands cause the changes in the tankfile to update the workfile, which is located in a file called CANDE/TEXT <recovery number>. The Update command forces the system to update the workfile immediately. Other commands such as Save, Run, Reseq, and Write cause the Update command to be implicitly invoked. The object code associated with the updated workfile is located in the updated codefile called CANDE/CODE <recovery number>.

When the Save command is entered the CANDE/CODE <recovery number> and CANDE/TEXT <recovery number> files become permanent files.



## A SERIES AND B 5000/B 6000/B 7000 CONCEPTS CANDE OVERVIEW

The tankfile contains any changes to the active workfile since the last update occurred. The information about the workfile located in the tankfile is written to the recovery file for retention following a failure. The recovery files that CANDE creates are named CANDE/RECV < recovery number >. The < recovery number > consists of the logical station number (LSN) followed by 1 digit to distinguish multiple recovery files for the same station.

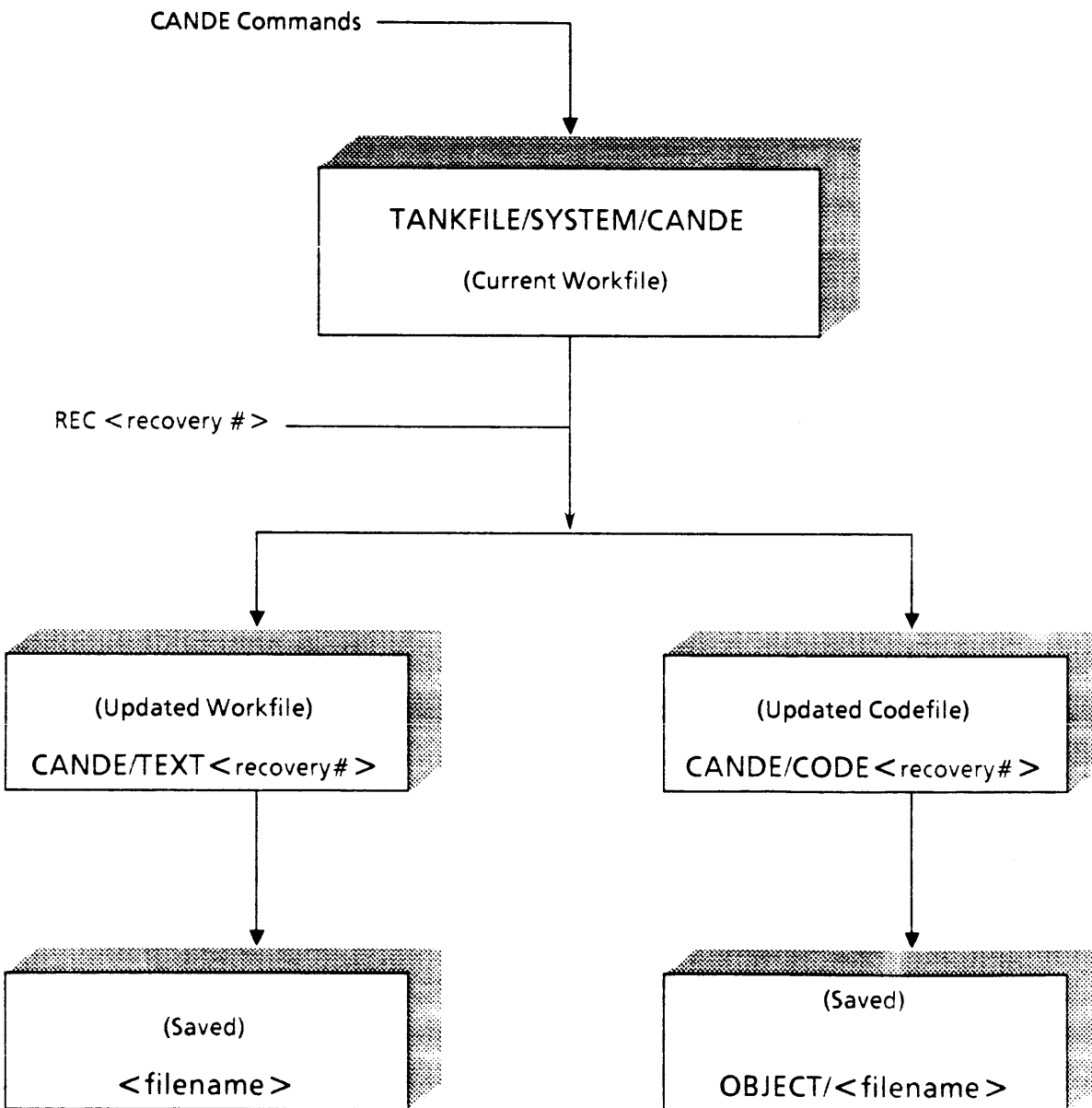


Figure 8-1 CANDE Files

# A SERIES AND B 5000/B 6000/B 7000 CONCEPTS BASIC CANDE COMMANDS

## *UNIT 2*

### ***CANDE EDITING COMMANDS***

#### **Objective**

Explain the basic CANDE editing commands.

#### **Purpose**

In order to create, compile and execute programs, you must know what editing commands CANDE provides for accomplishing these functions.

#### **Resources**

A Series CANDE Reference, Section 2 - General Information  
Section 4 - CANDE Commands

# A SERIES AND B 5000/B 6000/B 7000 CONCEPTS CANDE EDITING COMMANDS

## Editing Commands

CANDE contains a set of commands used for editing of text records in a file. A workfile, which is a temporary storage space, is always used for updating purposes instead of the original library file.

CANDE displays a # to the operator to signify the completion of a command.

## Log On/Off

CANDE requires that all of its users enter a usercode and password for security purposes. This is done by using the HELLO command.

```
HELLO <usercode> <password>
```

If a HELLO is entered to a currently running CANDE session, that session is terminated, and a new session is initiated.

When you are done using CANDE, you should log off. CANDE will hold all print files until you log off, so that all of your output can be printed together.

```
BYE
```

This command ends the current CANDE session and releases all print files for printing.

```
SPLIT
```

This command ends the current CANDE session, releases the print files and immediately starts a new CANDE session with the same usercode.

## A SERIES AND B 5000/B 6000/B 7000 CONCEPTS CANDE EDITING COMMANDS

### *Establishing A Workfile*

All editing must be applied to a workfile.

**MAKE <workfile name> <file type> <file attributes>**

The Make command creates a new workfile where the changes entered will be applied.

The file type should be supplied to ensure that sequence numbers will be located in the correct positions in the permanent file and that the type is consistent with the file's intended use. The default type is SEQ.

Certain file attributes can be specified by you at the workfile creation time, or the file attributes can be left to default.

Example:           **MAKE SAMPLE/WORKFILE C74**

**GET <file name>**

The Get command creates a workfile from a currently existing file. The workfile inherits the type and file attributes of the currently existing file.

Example:           **GET SOURCE/MYFILE**

**GET <file name> AS <workfile name>**

At times you will want to create a workfile with a name that is different from the original file name. This is accomplished by specifying the AS clause.

Example:           **GET SOURCE/MYFILE AS TEST/SOURCE/MYFILE**

## A SERIES AND B 5000/B 6000/B 7000 CONCEPTS CANDE EDITING COMMANDS

### *Single-Line/Page Entry*

Once a workfile is present, editing can be performed by using the single-line and/or page entry modes.

Single-line entry mode enters, replaces or deletes a single line in the workfile. The update is entered on the first line of the terminal starting with a sequence number and is followed by text not exceeding 80 characters in total length.

Page entry mode allows a full page of text to be entered and edited at one time. Movement back and forth through the workfile is easily performed in page mode.

Some terminals are restricted to Single-Line entry because of their physical characteristics.

### *Sequence Numbers*

All workfiles require sequence numbers for controlling the data.

#### SEQ

This command will establish sequence numbers for the workfile. The location of the sequence numbers on the screen will always be on the left but will be stored in the file in the correct columns depending on the workfile type.

CANDE provides the sequence numbers for each line to be entered in the workfile in sequencing mode. In single-line entry mode, one line number is presented at a time. In page entry mode, a whole page of numbered lines is presented. The default numbering scheme starts at 100 and increments each number by 100.

Examples:           SEQ  
                          SEQ 200 + 10

Sequence numbers are displayed, line after line or page after page, until a CANDE command other than SEQ is entered.

## A SERIES AND B 5000/B 6000/B 7000 CONCEPTS CANDE EDITING COMMANDS

### RESEQ

This command allows you to have CANDE renumber the lines in your workfile, thus allowing room in the numbering scheme for data insertion. New sequence numbers are assigned to each line in the workfile without changing the order of the lines.

Examples:           RESEQ  
                      RESEQ 200 + 10

### *Display or Edit Workfile*

CANDE provides commands to display the contents and the characteristics of the workfile.

### WHAT

This command provides information about the workfile such as workfile name, type, number of records, sequence number assigned to the last record, and the status of the workfile.

### LIST

The List command displays the contents of the workfile, or of other files.

CANDE displays the contents of the file one line after another or one page after another, until the whole workfile or the requested portion of the workfile is displayed. A page or line is displayed, and the operator inputs at least 1 blank to signify to CANDE to continue the display.

Examples:           LIST  
                      LIST 1000 - 2500  
                      LIST ACCOUNTS/PAYABLE/TEST/1

When CANDE is in list mode, the list must be completed before another editing command will be processed. To terminate the list before completion, enter ?BRK or ?DS. These special Control Commands inform CANDE to discontinue the list.

## A SERIES AND B 5000/B 6000/B 7000 CONCEPTS CANDE EDITING COMMANDS

### PAGE

This command lists the contents of the workfile and allows the operator to update the displayed information.

CANDE displays a full page of information, and at the top of the page, the token NEXT+ is displayed. To continue on to the next page of the display, transmit NEXT+. To move backwards in the display one page, transmit NEXT-. The token SAME can be used to refresh the currently displayed page of information.

When in page mode, changes can be made on the pages of information displayed. If you have updated any of the information on the page, make sure you transmit from some point on the page after the last change.

In both page mode and single line entry mode, you can key in a sequence number followed by the text, and CANDE will update the workfile by using the sequence number.

The Page command by default starts at the beginning of the file. Page followed by a sequence number starts at that number and goes forward.

Examples:           PAGE  
                      PAGE 1000

### DELETE <sequence number(s)>

At times it is necessary to eliminate a line or group of lines from the workfile. The sequence number and text is removed by using the delete command.

Examples:           DELETE 3000  
                      DELETE 100 - 250

## A SERIES AND B 5000/B 6000/B 7000 CONCEPTS CANDE EDITING COMMANDS

### *Compiling Source Language Files*

#### COMPILE

This command invokes a compiler. The simplest form of the command uses the type attribute of the workfile to know which compiler to invoke, and the workfile as the source to be compiled.

The object code produced will be named OBJECT/<workfile name>. For example, if the workfile is named SOURCE/EMP/UPDATE/PROG, the compiled object will be named OBJECT/SOURCE/EMP/UPDATE/PROG.

#### COMPILE <source file name >

This form of the command allows you to compile a source file that is not the workfile.

#### COMPILE AS \$ <file name >

This format allows you to name the object file that is produced. The \$ informs CANDE that you do not want the OBJECT/ directory to precede the file name.

### *Library Files*

#### SAVE

The save command takes the current workfile and object code file if present, and adds these files to your library. These files are non-permanent files until they are saved.

Changes made to the workfile and the compiled object code are not retained at the termination of a CANDE session unless a save command is entered.

You have the option of saving both the source and object code or just one of the files, depending on what your needs are.

Examples:           SAVE  
                      SAVE SOURCE  
                      SAVE AS NEW/SOURCE TEST/PAYROLL



## A SERIES AND B 5000/B 6000/B 7000 CONCEPTS CANDE EDITING COMMANDS

### *Program Execution*

RUN or  
EXECUTE

The Run or Execute commands cause the execution of a program. The object code associated with the current workfile is executed unless another name is specified.

Examples:            RUN  
  
                      RUN MYPROG/TEST

CANDE will look for an object code file name of OBJECT/<workfile name> to execute. For RUN MYPROG/TEST, CANDE searches for a code file named OBJECT/MYPROG/TEST. If the object has a name other than this format, the \$ must be specified so that CANDE does not use the OBJECT directory before the file name.

Example:            RUN \$MYPROG/PAYROLL/TEST

All line printer output from the program execution is held by CANDE until you log off. At log off time, the output will be printed.

### *CANDE Printing*

WRITE

The write command produces a line printer listing of all or part of the workfile, or of other files. The listing is not printed until you log off.

Examples:            WRITE  
  
                      WRITE 105 - 1501  
  
                      WRITE ACCOUNTS/PAYABLE/TEST/1

## A SERIES AND B 5000/B 6000/B 7000 CONCEPTS CANDE EDITING COMMANDS

### *Discarding a Workfile*

#### REMOVE

The remove command erases the workfile and object code file from disk. You can remove the workfile only, object code only or other files in your library using different formats of the remove command.

Examples:            REMOVE  
                      REMOVE SOURCE  
                      REMOVE ACCOUNTS/PAYABLE/TEST/1

### *Workfile Recovery*

After a system failure or CANDE failure, CANDE creates recovery files for users logged on at the time of the failure. The user must log on when CANDE is again operational and CANDE will display a list of all recovery files created. It is extremely important that you log on with the same usercode/password combination before requesting recovery of your workfile.

RECOVER <recovery number >

This command recovers your workfile to the time of the failure.

Example:            RECOVER 100

#### DISCARD

This command allows you to remove your unwanted recovery files.

Example:            DISCARD 150

#### SAVE RECOVERY

The Save command can be followed by the word recovery to have CANDE save the workfile in the form of a recovery file. This workfile must be retrieved at a later time by using the Recover command.

Save recovery can be very important when disk space problems occur. CANDE does not update the file during the save recovery process. CANDE builds a recovery file to retain the updates and this allows large files to be stored more quickly than using other forms of the save command.

## A SERIES AND B 5000/B 6000/B 7000 CONCEPTS CANDE EDITING COMMANDS

### *Location of Workfile*

CANDE uses the family specification associated with your usercode or established during this CANDE session to determine the disk family on which to store your files.

#### FAMILY

The Family command allows you to interrogate or alter your family specification.

#### FAMILY \*

This format of the Family command sets your family specification to the original value associated with your usercode, as declared in the SYSTEM/USERDATAFILE.

### *Workfile Names and File Attributes*

CANDE allows you to inquire into the files stored in your library. CANDE executes the utility SYSTEM/FILEDATA to acquire the requested information about your files.

#### FILES

CANDE will list all file names in your library when Files is entered

FILES < file name > or < file directory >

CANDE will list the file name or group of file names on your screen. You can determine the existence of a file or group of files using this command.

Example:           FILES TEST/PROG/SOURCE

#### LFILES

LFILES < file name > or < file directory >

This command lists the names and the attributes of one or more files.

Example:           LFILES TEST/PROG/SOURCE

When using FILES andLFILES to list groups of files, do not include the / = following the directory name.

**A SERIES AND B 5000/B 6000/B 7000 CONCEPTS  
BASIC CANDE COMMANDS**

**UNIT 3**

**CANDE CONTROL COMMANDS**

**Objective**

Explain basic CANDE Control Commands.

**Purpose**

In order to control and interrogate the CANDE network, you must be able to use CANDE Control Commands.

**Resources**

A Series CANDE Reference, Section 5 - Control Commands

A Series CANDE Operations, Section 2 - General Information  
Section 3 - Network Control Commands

# A SERIES AND B 5000/B 6000/B 7000 CONCEPTS CANDE CONTROL COMMANDS

## What Are MCS Control Commands?

A special group of commands, known as Control Commands, provides the ability to control and interrogate the operating environment.

Control Commands must begin with the defined control character followed by the command. These commands can be entered from any CANDE station.

The CANDE control command character by default is an ?.

## *Task Inquiry/Controlling Commands*

A group of control commands are used to inquire into the status of a task. Another group of control commands are used to control the task itself.

?WHY/Y

Displays information about the current task or specified task.

?CS

Reports the status of a compile. It displays the sequence number currently being processed and the number of syntax errors encountered up to this point.

?TI

Displays the process time, elapsed time and I/O time of the currently executing task.

?C

Lists the recently completed programs.

?CU

Displays the core utilization for a task.

?JA/MXA

Displays the status of jobs running under the usercode of your station.

## A SERIES AND B 5000/B 6000/B 7000 CONCEPTS CANDE CONTROL COMMANDS

### ?STATUS

Displays the status of the currently running tasks, currently executing command, such as Find or Replace, or any station on the network.

### ?MSG

Displays the most recent messages from a program.

### ?AX

Passes text from the operator to the executing program. This command is in response to a program's Accept statement.

### ?OK

Resumes processing for a suspended task.

### ?DS

Discontinues the current task.

### ?DUMP

Causes a program dump to be taken for a task.

### ?ST

Suspends the currently executing task or specified task.

### ?GO

Causes the CANDE session to be resumed with the first entry in the queue.

### ?FA

Alters the file attributes for a task currently waiting on a no file condition.

### ?RM

Allows a duplicate library condition to be resolved by removing the old file and retaining the new file.

## A SERIES AND B 5000/B 6000/B 7000 CONCEPTS CANDE CONTROL COMMANDS

?OF

Causes an End-Of-File condition for a task waiting on a no file condition where that file has the attribute of Optional set to true.

?HI

Causes the Exceptionevent interrupt for the task to be set. What happens to the program when this interrupt is set depends on the logic of the task.

### *System Inquiries Control Commands*

Control Commands (? <command >) are passed to the controlling MCS for your station. A controlling MCS such as COMS may have a second MCS running under it such as CANDE. If CANDE is not the controlling MCS, a CANDE control command can be passed to CANDE by entering ?? <command >.

Control commands that are used to inquire into the system are:

?WRU

Displays the identification of the station and the version of CANDE that is running.

?WHERE

Displays the station names and LSNs of all stations where the specified usercode is logged on.

?WM

Displays information about the currently running MCP.

?WT

Displays the current date and time.

?TD

Displays current date and time.

?TIME

Displays current time and date.

## A SERIES AND B 5000/B 6000/B 7000 CONCEPTS CANDE CONTROL COMMANDS

?SC

Displays the current system configuration.

?COUNTS

Displays information about the current activity of CANDE. This information includes the number of active tasks, number of active compiles and the number of active stations.

?SCHEDULE

Lists each schedule session currently active or scheduled with the same usercode.

### *Additional Control Commands*

?MCS

Causes control of the station to be transferred to another MCS.

?SS

Sends a message to another CANDE station.

?TO

Sends a message to all logged on stations under a specified usercode.



## A SERIES AND B 5000/B 6000/B 7000 CONCEPTS CANDE CONTROL COMMANDS

### *Network Control Commands*

A specialized group of Control Commands are Network Control Commands, which allow you to alter or determine the status of the CANDE network.

Network Control Commands are restricted commands and must be entered at an authorized station which is known as a control station.

### *Some Network Control Commands*

#### ?ABORT

Causes the CANDE MCS to terminate.

#### ?INFO

Displays the current settings of the CANDE operations and parameters.

#### ?OP

Allows interrogation, setting or resetting of CANDE options

#### ?READY

Allows you to ready a station.

#### ?WHERE

Displays information about all stations currently logged on to CANDE.

# A SERIES AND B 5000/B 6000/B 7000 CONCEPTS CANDE CONTROL COMMANDS

## Practice

Match the CANDE commands on the left with the descriptions on the right.

- |           |         |    |                                                            |
|-----------|---------|----|------------------------------------------------------------|
| _____ 1.  | DELETE  | a. | Determine the status of a task, station, or CANDE command. |
| _____ 2.  | HELLO   | b. | Discard the workfile or some other file.                   |
| _____ 3.  | LIST    | c. | Display and update the contents of the workfile.           |
| _____ 4.  | REMOVE  | d. | Display the contents of the workfile.                      |
| _____ 5.  | ?AX     | e. | Log on to CANDE.                                           |
| _____ 6.  | ?CS     | f. | Reply to an accept message from a program.                 |
| _____ 7.  | ?DS     | g. | Display status of a compile.                               |
| _____ 8.  | ?STATUS | h. | Remove lines from a workfile.                              |
| _____ 9.  | ?Y      | i. | Terminate an executing task.                               |
| _____ 10. | PAGE    | j. | Display information about the current task.                |

**A SERIES AND B 5000/B 6000/B 7000 CONCEPTS  
CANDE**

***UNIT 4***

***COMS WINDOWS AND DIALOGS***

**Objective**

Explain windows and dialogs.

**Purpose**

In order to establish multiple CANDE sessions and use the CANDE window through MARC, you should be aware of COMS windows and dialogs and their functions.

**Resources**

A Series Menu-Assisted Resource Control User's Guide, Section 8 - COMS Windows and Dialogs

A Series CANDE Operations, Section 2 - General Information

# A SERIES AND B 5000/B 6000/B 7000 CONCEPTS COMS WINDOWS AND DIALOGS

## Window/Dialog

COMS has a distinctive feature of supporting multiple windows and dialogs. A **Window** is an interface to a program environment (connection between terminal and the program that the terminal is communicating with). A **Dialog** is a unique view into a window (each access to a program environment).

COMS supports one to eight dialogs per program environment or window. To identify which window and dialog your terminal is currently assigned to, the system will display on the status line of your terminal `<window name>/<dialog number>`.

Examples:           MARC/1  
                      MARC/3  
                      CANDE/5

Certain windows in COMS are considered standard windows, and you will probably find these windows on your system. These windows are CANDE, MARC and GEMCOS. Other windows can be added to your system by defining the windows to COMS using the COMS Utility. The commands below can be entered to display the current windows on your system.

?WINDOWS (COMS)

WINDOW (MARC Choice)

The commands below can be entered to determine which window and dialog you are currently communicating with, if this information is not displayed on your terminal.

?STATUS (COMS) or

?WRU (MARC)           .

## A SERIES AND B 5000/B 6000/B 7000 CONCEPTS COMS WINDOWS AND DIALOGS

### *Changing Windows/Dialogs*

?ON <window name>/<dialog number>

The ON command is used to move from one Window/Dialog to another. This is a COMS command.

If the dialog number is not specified, the default is dialog 1 of the requested window. If an \* is specified as the window name, you are moved to the default window for your station, which is usually MARC/1.

Examples:       ?ON MARC/2  
                  ?ON CANDE/1  
                  ?ON MARC

ON (MARC Action)

When on the MARC window, you can use the ?ON command to move from one window to another, or you can use the MARC Action ON.

If ON is entered without specifying a window, a window screen will be displayed for you to fill in. You can bypass this window screen by specifying the window name and dialog number following ON in the action field.

Example:        ON CANDE/1

If you are currently in MARC on the Tasking mode screen, you must use the ON action to switch to another window or another dialog of MARC to do any other type of work until the task is completed.

**A SERIES AND B 5000/B 6000/B 7000 CONCEPTS  
BASIC CANDE COMMANDS**

**UNIT 5**

**ADDITIONAL CANDE COMMANDS**

**Objective**

Explain additional CANDE commands.

**Purpose**

In order to perform more complex editing on files, you need to be aware of the additional commands available in CANDE.

**Resources**

A Series CANDE Reference, Section 4 - CANDE Commands

A Series Printing Utilities User's Guide, Section 2 - BACKUP PROCESSOR Utility

## A SERIES AND B 5000/B 6000/B 7000 CONCEPTS ADDITIONAL CANDE COMMANDS

### More Commands

In unit 2 of this section, some basic file manipulation commands were discussed. This unit will cover some additional commands you may need to control your workfiles.

### *Text/File Movement Commands*

CANDE has some commands that are used to move text within a workfile or among workfiles. The text movement commands are:

#### INSERT

This command copies lines from one location to another location in the same workfile and assigns new sequence numbers to the copied lines. Another format of this command will copy text from a file that is not the workfile into the workfile.

Examples:           INSERT 2000 - 2900 AT END + 10  
                      INSERT COBOL/COPY/SOURCE AT 350  
                      INSERT COBOL/COPY/SOURCE 110 - 300 AT 2010 + 100

#### MOVE

This command moves lines from one location to another in the workfile and assigns new sequence numbers to the relocated lines. Lines are moved, not copied as in the Insert command.

Examples:           MOVE 1100 - 2100 TO 3501 + 5  
                      MOVE TO END + 100 4100 - 5000

#### FIND

This command searches the workfile for the appearances of the specified target text. The sequence number or sequence number and text are displayed on the screen.

Examples:           FIND ;EMPNAME; ;T  
                      FIND ADDRESS1/ 5000 - 20000

## A SERIES AND B 5000/B 6000/B 7000 CONCEPTS ADDITIONAL CANDE COMMANDS

### REPLACE

This command searches the workfile for appearances of the specified target text and then replaces the target text with the new text.

Examples:           REPLACE /EMPNAME//EMPLOYEE/ :T  
                      REPLACE ;ADDRESS;;EMPADD; 500 - 2450  
                      REPLACE LIT .EMP..EMPLOYEE. :T

The File movement commands are:

### MERGE

This command copies the contents of a file into the workfile using the sequence numbers to order the lines. When a sequence number matches, the record of the workfile is retained instead of the record in the file.

Examples:           MERGE FILE/SOURCE/1  
                      MERGE FILE/NEWSOURCE 10 - 240

### REMERGE

This command copies the contents of a file into the workfile using the sequence numbers to order the lines. When matching sequence numbers are found, the record from the file replaces the record in the workfile.

Examples:           REMERGE FILE/SOURCE/10A 41000 - 51000  
                      REMERGE NEWSOURCE/FILEA



## A SERIES AND B 5000/B 6000/B 7000 CONCEPTS ADDITIONAL CANDE COMMANDS

### *Miscellaneous Commands*

CANDE contains some additional commands that can be very useful for text editing, program execution, and communications among users.

#### NEWS

This command displays the contents of the CANDE news file. The news file usually contains information that the system administrator wishes to pass on to the users.

#### DO

The Do command reads a file containing CANDE commands and executes each command in order. This file is often referred to as a Do File. Do files are very useful for applying the same patches to multiple workfiles.

Examples:           DO DO/FILE/1  
                      DO PRACTICE/DO/FILE

#### SCHEDULE

This command invokes a file containing CANDE commands as a separate CANDE session. This session can begin immediately or at a later time.

The input commands are merged with the output from each command and a schedule output file, which is an image of the information that would have appeared on the terminal, is produced. This file can be examined at a later time to verify the completion of each command.

Examples:           SCHEDULE SCH/FILE : AFTER 2200  
                      SCHEDULE SCH1/FILE TO SCH1/OUTPUT/FILE

#### START

This command causes the Work Flow Language compiler to be invoked and a Work Flow job to be started. The file name or workfile specified must contain Work Flow Language statements and the file type must be Job.

Example:           START WFL/FILE 1

## A SERIES AND B 5000/B 6000/B 7000 CONCEPTS ADDITIONAL CANDE COMMANDS

### BACKUPPROCESS

This command invokes the Backup Processor Utility which allows you to copy to disk, print, list on a terminal or remove printer backup files.

This command is very useful to view printer backup files at a terminal before ending the CANDE session.

The Backup Processor Utility has its own set of commands to control the printer backup files. The interactive commands include:

COPY	Copies a backup file
DIRECTORY	Lists the files in the backup file directory
HELP	Displays help information
LIST	Displays the contents of a backup file
OPTION	Specifies the Backup Processor utility options
QUERY or WHAT	Displays information about a backup file
QUIT	Terminates the Backup Processor utility
PRINT	Sends a print request for a backup file
REMOVE	Removes a backup file
SELECT or NEXT	Specifies a particular directory or file
FIRST	Displays the first page of text in the backup file
LAST	Displays the last page of text in the backup file
SAME	Redisplays the current page of text
-	Displays the previous page of text
+	Displays the next page of text

**SECTION 9**

**STACK ARCHITECTURE CONCEPTS**

# A SERIES AND B 5000/B 6000/B 7000 CONCEPTS STACK ARCHITECTURE CONCEPTS

## **INTRODUCTION**

### **Section Objective**

Recognize the major characteristics of the A Series stack architecture.

### **Purpose**

A knowledge of stack architecture concepts is necessary when reading reference manuals, compile listings, and program dumps.

### **Unit Objectives**

Identify the various elements of an object program as it exists on disk.

Identify the memory structures used in program execution.

Identify the word formats used internally on the A Series systems.

Identify the most common control word formats used internally on the A Series systems.

Recognize the functions of the display registers.

Recognize the functions of the top-of-stack registers.

Identify the operators used in basic calculations.

# **A SERIES AND B 5000/B 6000/B 7000 CONCEPTS STACK ARCHITECTURE CONCEPTS**

## **UNIT 1**

### **OBJECT CODE FILE LAYOUT**

#### **Objective**

Identify the various elements of an object program as it exists on disk.

#### **Purpose**

Familiarity with the structure of an object program is necessary when reading compiler listings and dumps.

#### **Resources**

A Series System Architecture , Volume 2, Section 1 - Data Structures  
Section 2 - Stack Concepts and Processor State

# A SERIES AND B 5000/B 6000/B 7000 CONCEPTS OBJECT CODE FILE LAYOUT

## Object Code File Layout on a Pack

The object code files that are created by compilers consist of the following elements, which are also illustrated in Figure 9-1.

### *Object Code Segments*

The object code instructions generated by compilers are grouped into variable-length object code segments, depending on the size and structure of the source program. This saves memory space when the program executes by allowing individual segments, rather than the whole program, to be brought into memory.

#### COBOL program segmentation

The programmer may use **SECTIONs** to divide the program into segments.

#### ALGOL Program segmentation

Each procedure or block corresponds to an object code segment.

The programmer may also use dollar options to specify segment boundaries .

#### Automatic Program Segmentation

The compilers allow a maximum of 1500 words of object code per segment. A new segment is created automatically when this limit is exceeded.

### *Segment Dictionary*

The segment dictionary contains a segment descriptor for each object code segment, to store the address of the object code segment within the object code file.

**A SERIES AND B 5000/B 6000/B 7000 CONCEPTS  
OBJECT CODE FILE LAYOUT**

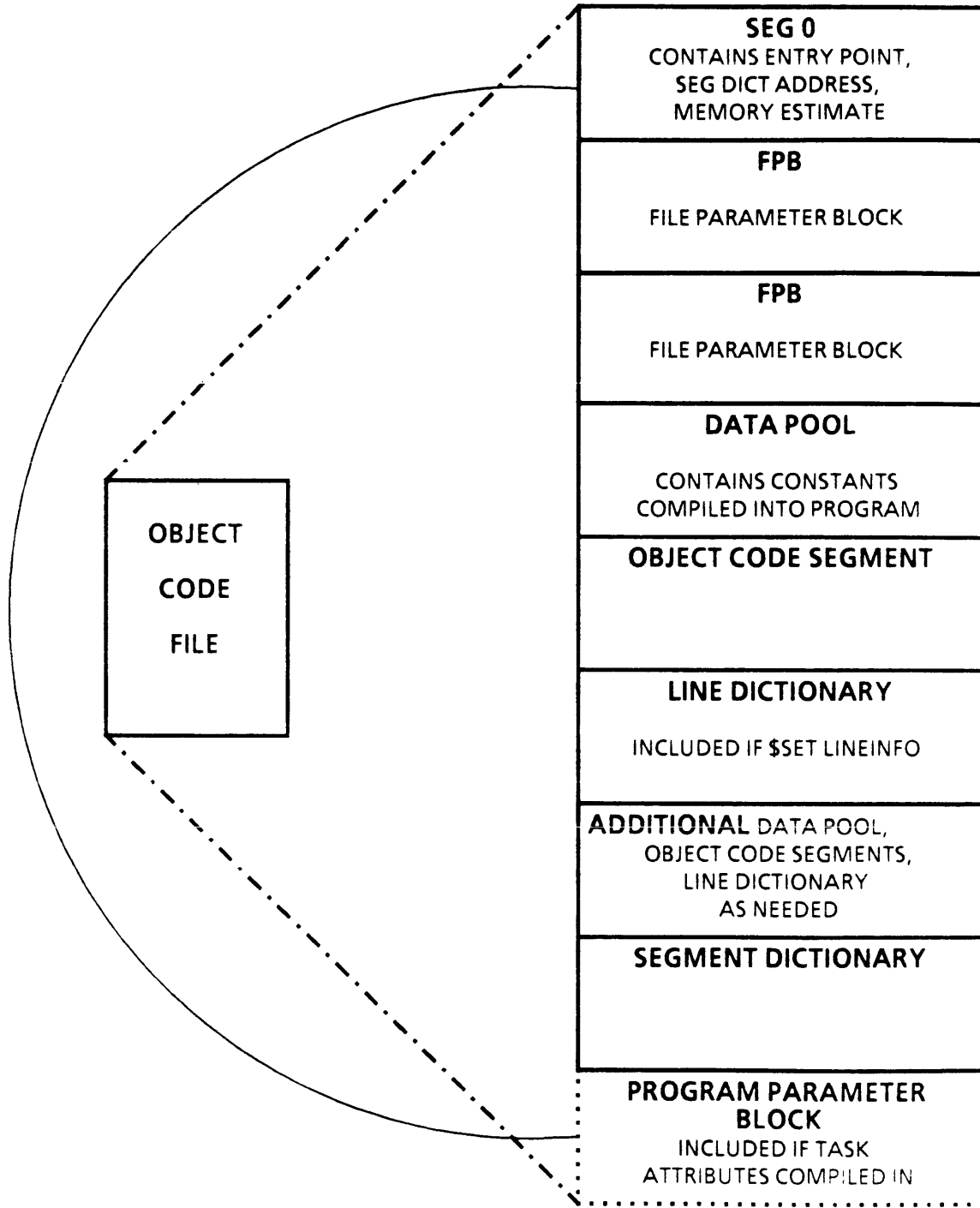


Figure 9-1 Object Code File Layout on a Pack

# A SERIES AND B 5000/B 6000/B 7000 CONCEPTS OBJECT CODE FILE LAYOUT

## *Segment 0*

Segment 0 contains information used by the MCP when the program is executed.

The **memory estimate** specifies the amount of memory required to begin executing this program. This estimate is updated each time the program is executed, so that it will become more accurate over time.

The **address of the segment dictionary** allows the MCP to locate the segment dictionary within the object code file.

The **entry point** indicates which segment, word, and byte contains the first instruction to be executed.

## *File Parameter Block(s)*

Each file declared in the program will have a File Parameter Block (FPB), containing the file attributes (for example, BLOCKSIZE, AREASIZE) that were declared in the source program.

## *Data Pool*

The Data Pool consists of literals, such as page headings, that were compiled into the program.

## *Line Dictionary*

If the program is compiled with the dollar option LINEINFO, a line dictionary will be created to relate object code instructions to sequence numbers in the source file. If the program faults, the sequence number of the source statement where the failure occurred will be displayed to aid in debugging.

## *Program Parameter Block*

If any task attributes (for example, family substitution statement) are compiled into the program, a program parameter block (PPB) will be created to store them.



**A SERIES AND B 5000/B 6000/B 7000 CONCEPTS  
STACK ARCHITECTURE CONCEPTS**

**UNIT 2**

***MEMORY STRUCTURES FOR PROGRAM EXECUTION***

**Objective**

Identify the memory structures used in program execution.

**Purpose**

Familiarity with memory structures is necessary when reading dumps and reference manuals.

**Resources**

A Series System Software Support Reference Manual, Section 8 - Memory Management

## **Memory Structures for Program Execution**

Every executing program has at least three structures in memory, as shown in Figure 9-2.

The **Process Stack** (often called "the stack") is a series of words in memory that grows and shrinks as the program executes, to store the program's working environment. The term **process** refers to a single execution of a program. The stack contains:

Temporary data (for example, Cobol 77-level items, Algol real items)

Addresses of data (for example, data in a file)

Addresses of object code

Program history (for example, a procedure invoked another procedure)

The MCP accounting area (a series of words built by the MCP at the bottom, or base, of each process stack)

The **Segment Dictionary** contains the addresses of the object code segments (segment descriptors). This is copied into memory from the object code file on disk.

The **Process Information Block** holds the task attributes for the program.

Attributes are copied from the Program Parameter Block in the object code file.

Additional attributes are developed and accumulated as the program executes (for example, ELAPSEDTIME).

There may also be one or more object code segments in memory for the program, as well as file buffers and other data structures.

# A SERIES AND B 5000/B 6000/B 7000 CONCEPTS MEMORY STRUCTURES

## MEMORY

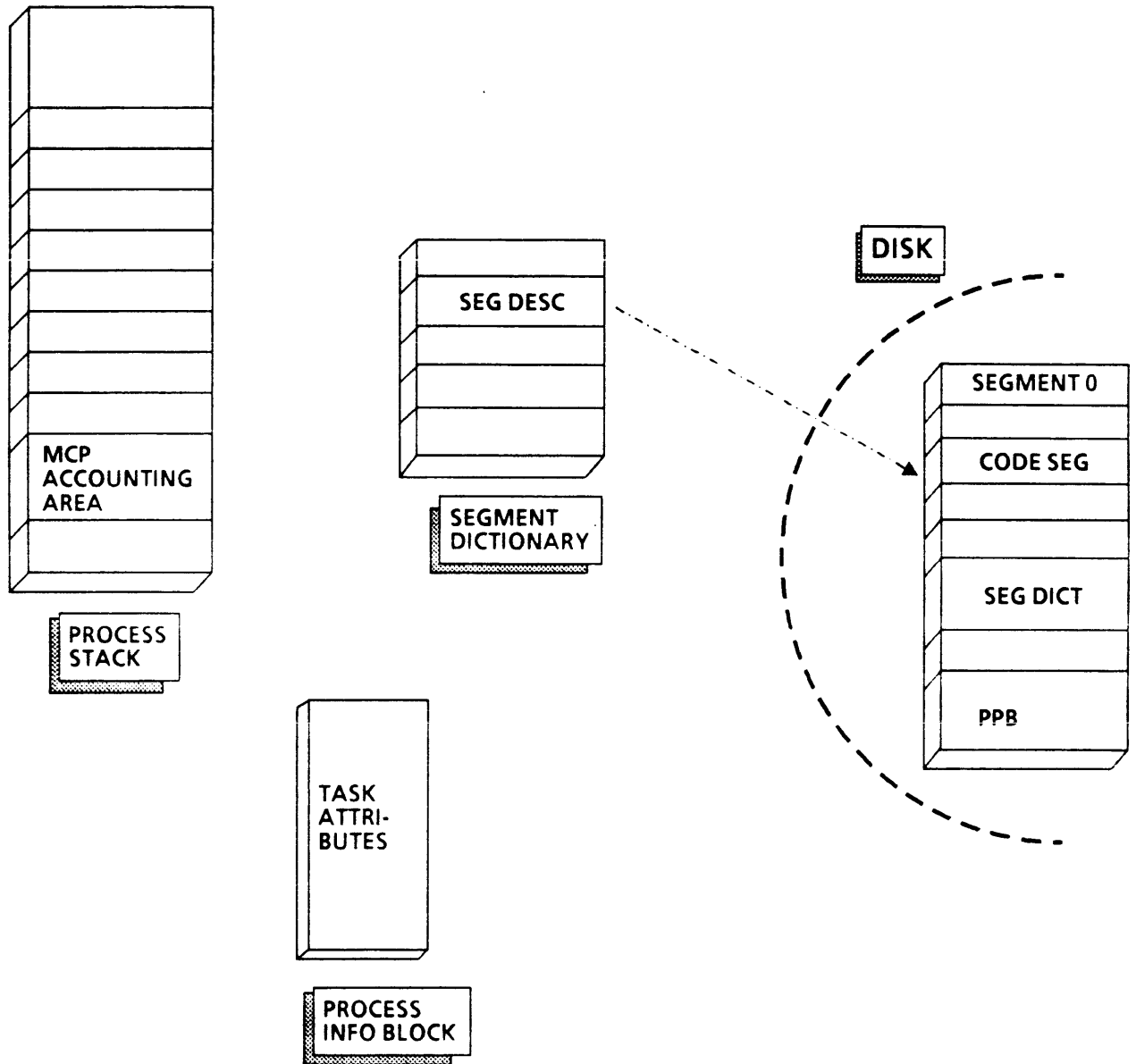


Figure 9-2 Memory Structures for Program Execution

## A SERIES AND B 5000/B 6000/B 7000 CONCEPTS MEMORY STRUCTURES

### *Steps in Program Execution*

The MCP performs the following steps when a program is executed:

1. Locates the object code file using the directory structures.
2. Checks the memory estimate in Segment 0 to determine if there is enough memory available to execute the program at this time. If not, the program will be scheduled until there is enough memory available.
3. Builds the Process Information Block (PIB) in memory.
4. Uses the address of the segment dictionary in Segment 0 to locate the segment dictionary in the object file on disk.
5. Copies the segment dictionary into memory.
6. Allocates space for the Process Stack in memory, and sets up the MCP accounting area in the stack.
7. Obtains the entry point (address of first executable instruction) from Segment 0.
8. Finds the Segment Descriptor for the required object code segment in the Segment Dictionary.
9. Causes a Presence Bit Interrupt, to bring the required segment into memory.
10. Updates the Segment Descriptor to point to the object code segment in memory, instead of on disk, as shown in Figure 9-3.
11. Executes the object code instructions, beginning at the entry point.
12. Brings additional object code segments into memory as needed, and updates their Segment Descriptors to point to memory locations.
13. Updates the memory estimate in Segment 0 at end of task.
14. Frees the memory space occupied by the Process Stack, Segment Dictionary, and object code segments at end of task. Frees the Process Information Block at end of job.

**A SERIES AND B 5000/B 6000/B 7000 CONCEPTS  
MEMORY STRUCTURES**

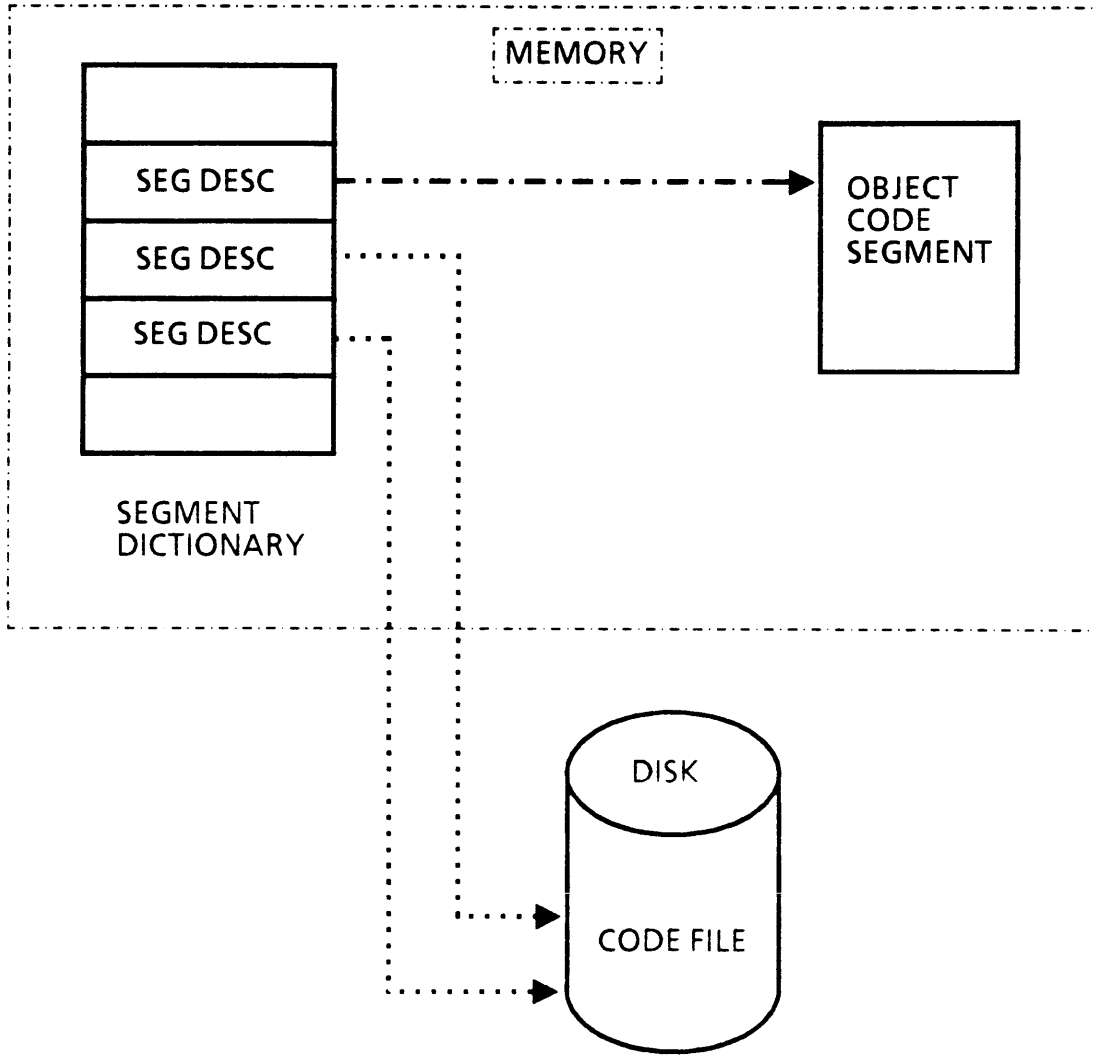


Figure 9-3 Segment Descriptors

## A SERIES AND B 5000/B 6000/B 7000 CONCEPTS MEMORY STRUCTURES

### Reentrant Code

Reentrant code allows multiple executions of the same program to share some of the memory structures, conserving memory and pack accesses. Figure 9-4 illustrates reentrant code.

The Segment Dictionary and object code segments in memory are shared.

Each execution of the program has its own Process Stack and Process Information Block.

The Segment Dictionary will not be removed from memory until all of the executions of the program have ended.

A SERIES AND B 5000/B 6000/B 7000 CONCEPTS  
MEMORY STRUCTURES

MEMORY

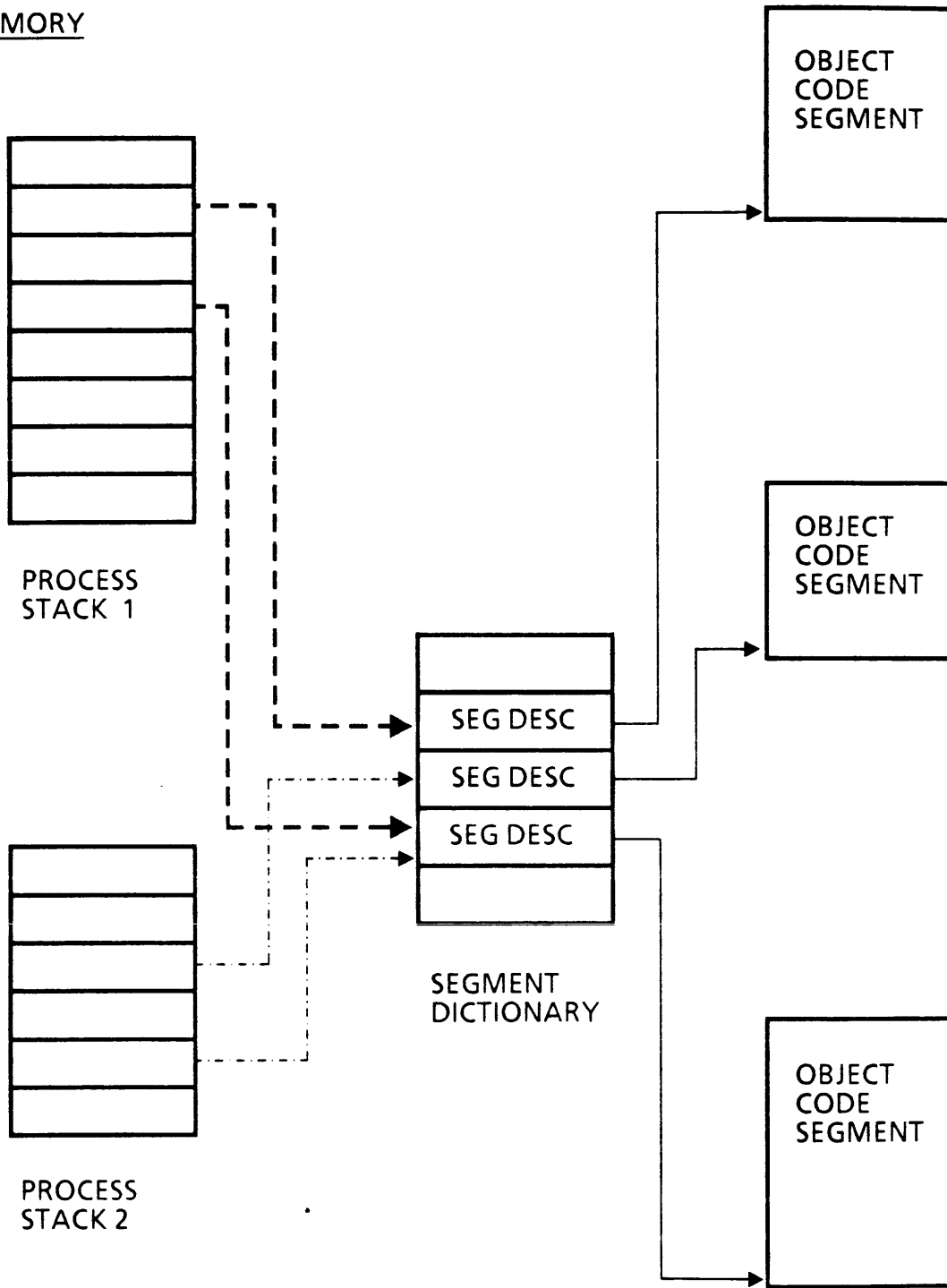


Figure 9-4 Reentrant Code

## A SERIES AND B 5000/B 6000/B 7000 CONCEPTS MEMORY STRUCTURES

### Practice

Match the terms on the left with the descriptions on the right.

- |          |                           |                                                                                                           |
|----------|---------------------------|-----------------------------------------------------------------------------------------------------------|
| _____ 1. | Process Information Block | a. This action occurs at program execution time if there is insufficient memory available.                |
| _____ 2. | Process Stack             | b. This action occurs when a segment descriptor is accessed for the first time.                           |
| _____ 3. | File Parameter Blocks     | c. The MCP builds this structure in memory and adds information from the program's PPB if it is present.  |
| _____ 4. | Scheduled                 | d. The memory estimate is located in this part of the object file.                                        |
| _____ 5. | Presence Bit Interrupt    | e. These parts of the object code file contain the file attributes declared in the source program.        |
| _____ 6. | Segment Dictionary        | f. This structure stores the program's working environment.                                               |
| _____ 7. | Segment 0                 | g. These vary in size and number depending on the structure of the source program.                        |
| _____ 8. | Object Code Segments      | h. This memory structure contains the segment descriptors for object code segments on disk and in memory. |



**A SERIES AND B 5000/B 6000/B 7000 CONCEPTS  
STACK ARCHITECTURE CONCEPTS**

***UNIT 3***

***WORD FORMATS AND DATA REPRESENTATION***

**Objective**

Identify the word formats used internally on the A Series systems.

**Purpose**

Familiarity with word formats is necessary when reading reference manuals and program dumps.

**Resources**

A Series System Architecture, Volume 2, Section 1 - Data Structures

# A SERIES AND B 5000/B 6000/B 7000 CONCEPTS WORD FORMATS AND DATA REPRESENTATION

## Word Formats

A word on an A Series system consists of 6 bytes, or 48 bits, of data. When the word is in memory or in the processor, 4 additional bits are associated with the word. The parts of a word are described below.

The bits are numbered 0 through 51, as shown in Figure 9-5.

The **information field** contains the 48 bits of data, in bits 0 through 47 of the word. Bit 0 is the least significant bit, and bit 47 is the most significant bit. The 48 bits will be redefined to have various meanings, depending on the type of data in the word.

The **tag field** indicates the type of information stored in the word (for example, numeric data or object code), in bits 48 through 50.

The **parity bit** is bit 51. This bit is used by the system in checking for errors in reading and writing the word in memory or in the processor.

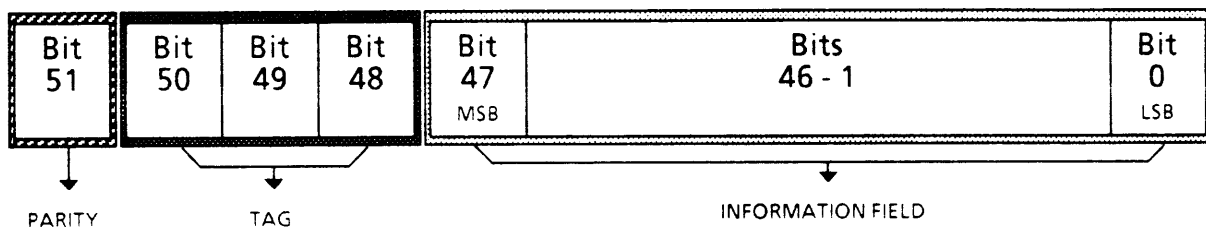


Figure 9-5 Word Layout

### Partial Word Notation

Partial word notation is a convention used in the reference manuals, and in ALGOL or COBOL74 programs, to refer to bits or fields within a word. The syntax of partial word notation is [ mm : nn ].

The most significant (left-most) bit of the field is specified before the colon (mm).

The length of the field in bits is specified after the colon (nn).

Examples:

[ 47 : 48 ] refers to the information field.

[ 50 : 3 ] denotes the tag field.

[ 51 : 1 ] is the parity bit.

[ 38 : 6 ] refers to 6 bits within the information field.

# A SERIES AND B 5000/B 6000/B 7000 CONCEPTS WORD FORMATS AND DATA REPRESENTATION

## Number Bases

Several number bases are used in the A Series systems.

**Decimal (Base 10)** is the most common format for numeric data on reports and displays, because it is the format that humans use daily. Base 10 uses the digits 0 through 9, with each position representing a power of 10.

**Binary (Base 2)** is used in memory, in the processor, and on disk to represent all data as a series of bits (binary digits). Binary uses the digits 0 and 1, with each position representing a power of 2.

**Octal (Base 8)** is used to store numeric data within the processor. Octal uses the digits 0 through 7, with each position representing a power of 8. Each group of 3 bits corresponds to 1 octal digit.

**Hexadecimal (Base 16)** is used in program dumps and SYSTEM/DUMPALL listings to represent binary data in a more readable format, since each group of 4 bits corresponds to 1 hexadecimal digit. Base 16 uses the digits 0 through 9, and A through F, with each position representing a power of 16.

The A Series word format is often illustrated using columns of 4 bits each, as shown in Figure 9-6.

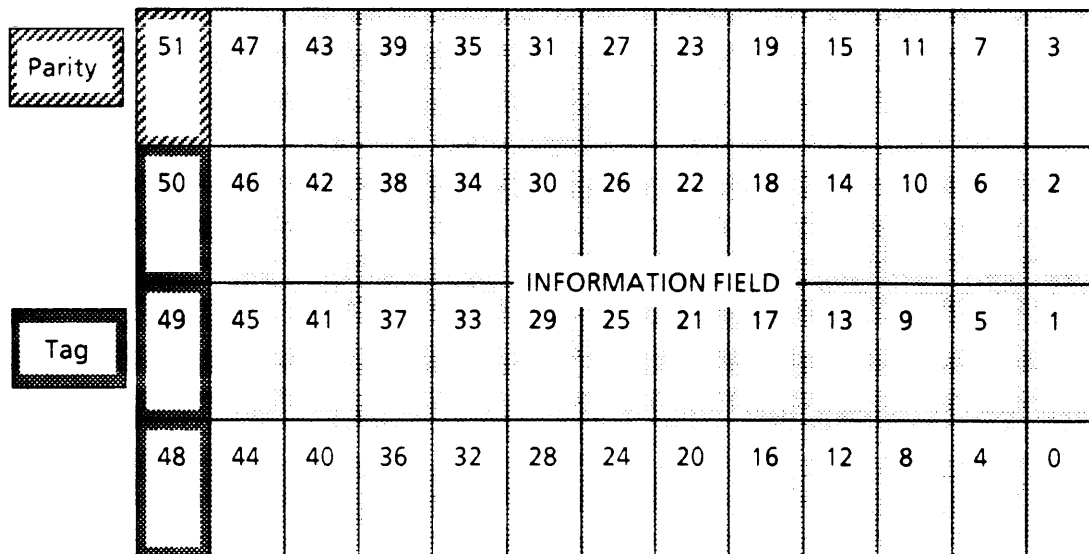


Figure 9-6 Word Format

## A SERIES AND B 5000/B 6000/B 7000 CONCEPTS WORD FORMATS AND DATA REPRESENTATION

In Figure 9-6, each column of 4 bits corresponds to 1 hexadecimal digit. The 12 columns in the information field represent the 12 hexadecimal digits that will be printed for each word in a program dump.

A bit may contain only 0 or 1.

The values of the bits in each column are 1 (bottom bit), 2, 4, and 8 (top bit).

The values of the bits in a column add to give a hexadecimal digit.

Example: A word in a program dump containing the hexadecimal digits 2594000000 actually has the bit pattern shown in Figure 9-7 (only the 1-bits, not the 0-bits, are shown).

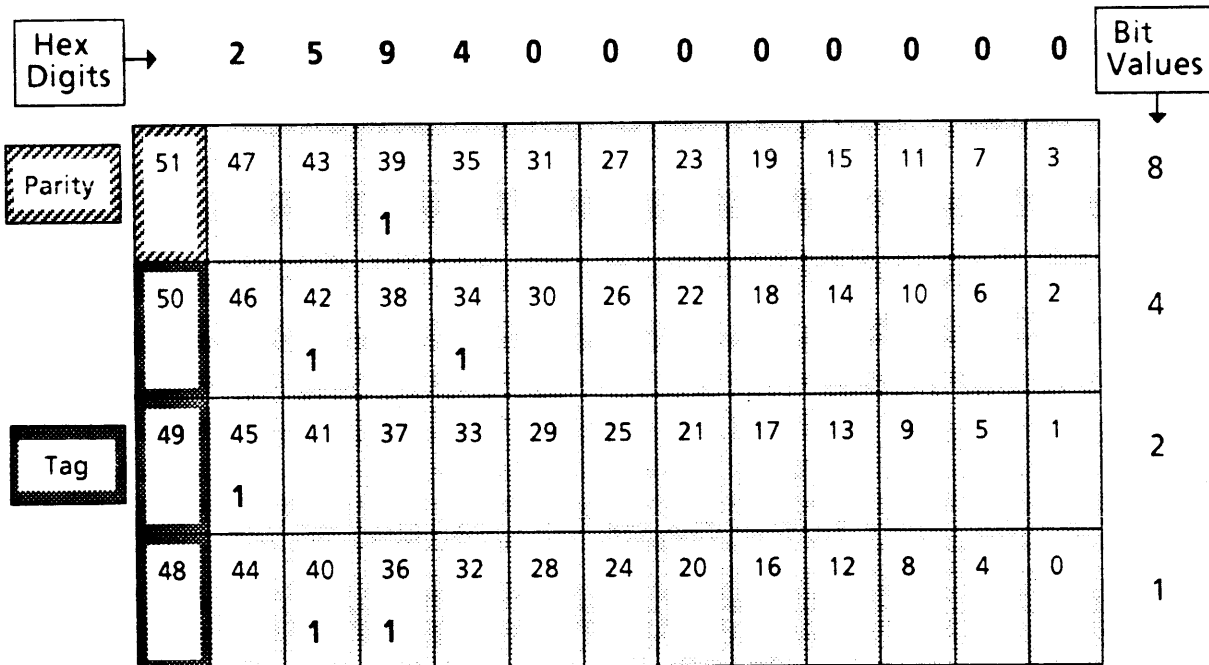


Figure 9-7 Word Containing the Hexadecimal Value 25940000000

# A SERIES AND B 5000/B 6000/B 7000 CONCEPTS WORD FORMATS AND DATA REPRESENTATION

## Sample Control Word Formats

Control Words are specially formatted words placed on the stack during program execution. Two types of control words are illustrated below. Additional control words will be defined in Unit 4.

### Single Precision Operand

A Single Precision Operand is a word that is placed on the stack to store the value of a numeric data item declared in the program (for example, an ALGOL real variable, or a COBOL 77-level item with PIC 9(5)V99). A tag field of 0 identifies a word as a Single Precision Operand.

Single Precision Operands use octal (base 8) to represent the value of the variable. The number is automatically converted from and to decimal for input and output. The value is stored as :

$$\text{NUMBER} * 8^{\text{POWER}}, \text{ or } \text{MANTISSA} * 8^{\text{EXPONENT}}$$

The format of Single Precision Operands is described below and illustrated in Figure 9-8.

- [ 46 : 1 ] Sign of Mantissa 0 = Positive, 1 = Negative
- [ 45 : 1 ] Sign of Exponent 0 = Positive, 1 = Negative
- [ 44 : 6 ] Exponent
- [ 38 : 39 ] Mantissa

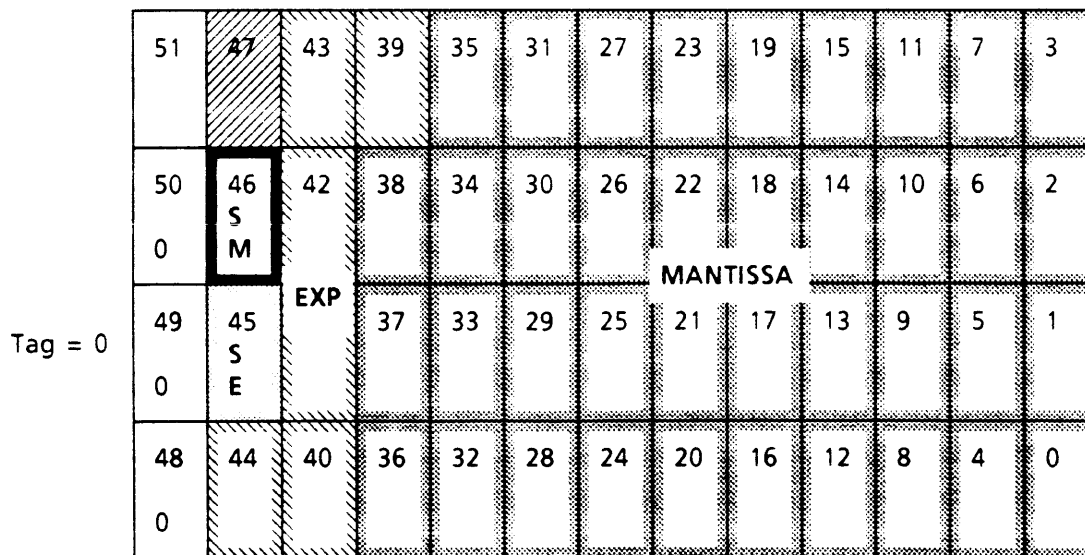


Figure 9-8 Single Precision Operand Format

Usually, program dumps print Single Precision Operands both as 12 hexadecimal digits and as a decimal number. Operands within arrays or tables print only the 12 hexadecimal digits.

The word in Figure 9-7 is actually a Single Precision Operand with the value of 10.



# A SERIES AND B 5000/B 6000/B 7000 CONCEPTS WORD FORMATS AND DATA REPRESENTATION

## Data Representation

Two main character sets are used on the A Series system.

**EBCDIC** (Extended Binary Coded Decimal Interchange Code) is used to represent character data on disk and tape, in memory, and in the processor.

Each EBCDIC character contains 8 bits, or 2 hexadecimal digits. Program dumps and **SYSTEM/DUMPALL** will print 2 hexadecimal digits per character.

A word can contain 6 characters (6 bytes) of alphabetic data in the 48 information bits.

**ASCII** (American Standard Code for Information Interchange) is used in the data comm subsystem, because it is the industry standard for communications.

Data is translated between ASCII and EBCDIC by the line adaptor.

Each ASCII character contains 7 bits, plus a parity bit.

**A SERIES AND B 5000/B 6000/B 7000 CONCEPTS  
STACK ARCHITECTURE CONCEPTS**

***UNIT 4***

***CONTROL WORDS***

**Objective**

Identify the most common control word formats used internally on the A Series systems.

**Purpose**

Familiarity with control words is necessary when reading reference manuals and program dumps.

**Resources**

A Series System Architecture, Volume 2, Section 1 - Data Structures



## A SERIES AND B 5000/B 6000/B 7000 CONCEPTS CONTROL WORDS

### Control Words

As defined in unit 3, control words are specially formatted words placed on the stack during program execution. Two types of control words, Single Precision Operands and Segment Descriptors, were illustrated in Unit 3. Additional control words will be defined in this unit.

### *Sample ALGOL Program*

The sample ALGOL program in Figures 9-10 will be used in Units 4, 5, and 6 to illustrate various stack concepts. Selected portions of the program will be discussed in Units 4 and 5, and then the entire program will be summarized in Unit 6.

This program reads a disk file of employees, calculates the gross pay, taxes, and net pay for an hourly employee, and prints a paycheck.

Employees are paid an hourly rate for all hours worked. If an employee works more than 40 hours, overtime pay is calculated at an additional one-half the hourly rate for the extra hours (this is equivalent to time-and-a-half).

Some portions of the program have been shown as comments (following the % symbol), because the purpose of the program is to present stack topics, not to teach ALGOL.

A compile listing and program dumps for this program are included in Section C.



**A SERIES AND B 5000/B 6000/B 7000 CONCEPTS  
CONTROL WORDS**

```
2100  PROCEDURE CALC_CHECK;
2200      BEGIN
2300          REAL REG_PAY, OT_PAY;

2400          PROCEDURE CALC_OT;
2500              BEGIN
2600                  OT_PAY := HOURLY_RATE * .5 * (HOURS_WORK - 40);
2700              END OF CALC_OT;

2800          REG_PAY := HOURS_WORK * HOURLY_RATE;
2900          IF HOURS_WORK GTR 40 THEN CALC_OT;
3000          GROSS_PAY := REG_PAY + OT_PAY;
3100          CALC_TAXES;
3200          NET_PAY := GROSS_PAY - TOTAL_TAXES;
3300      END OF CALC_CHECK;

3400  % % % MAIN LOGIC % % %

3500  % READ EMPLOYEE, MOVE VALUES INTO VARIABLES
3600  CALC_CHECK
3700  % PRINT CHECK;
3800  END OF PROGRAM.
```




Figure 9-10b Sample ALGOL Program Part 2

## A SERIES AND B 5000/B 6000/B 7000 CONCEPTS CONTROL WORDS

### *Mark Stack Control Word*

A Mark Stack Control Word (MSCW) is placed on the stack to record that a new procedure has been executed. The Mark Stack Control Words keep information about the history of the program's execution, and about the addressing environment (this will be discussed in Unit 5).

ALGOL puts a MSCW on top of the stack when a procedure is invoked by name.

COBOL places a MSCW on top of the stack when the Procedure Division begins executing.

Since the MCP treats each program as a procedure, an MSCW is placed on the stack when the program begins executing.

### *Return Control Word*

A Return Control Word (RCW) always appears above a MSCW on the stack, to point to the object code instruction that the program should return to after finishing the procedure. Figure 9-11 illustrates the MSCW and RCW that are built on the stack when the sample ALGOL program in Figures 9-10 begins executing.

### *Single Precision Operand*

As described in Unit 3, a Single Precision Operand is placed on the stack to store the value of a numeric data item declared in the program. Each real variable declared in lines 1100-1200 of the sample ALGOL program corresponds to a Single Precision Operand which is initialized to 0, as shown in Figure 9-11.

The maximum integer value of a Single Precision Operand is 549,755,813,887. Depending on the programming language used, a Single Precision Operand can be 12 digits long, including decimal places. ALGOL also allows scientific notation (powers of 10), and larger maximum number values.

### *Double Precision Operand*

Double Precision Operands are used to store the values of very large numbers, or numbers that require many decimal positions. Depending on the programming language, a Double Precision Operand may store values up to 23 digits long, including decimal places. A Double Precision Operand occupies 2 words on the stack.

**A SERIES AND B 5000/B 6000/B 7000 CONCEPTS  
CONTROL WORDS**

```

1000 BEGIN
1100 REAL HOURS_WORK, HOURLY_RATE, GROSS_PAY,
1200     NET_PAY, TOTAL_TAXES;
1300 FILE EMPLOYEES (KIND = DISK, TITLE = "EMPLOYEES/ACTIVE.");
1400 ARRAY TAX_TABLE [ 0 : 9 ];
1500 PROCEDURE CALC_TAXES;
2100 PROCEDURE CALC_CHECK;

```

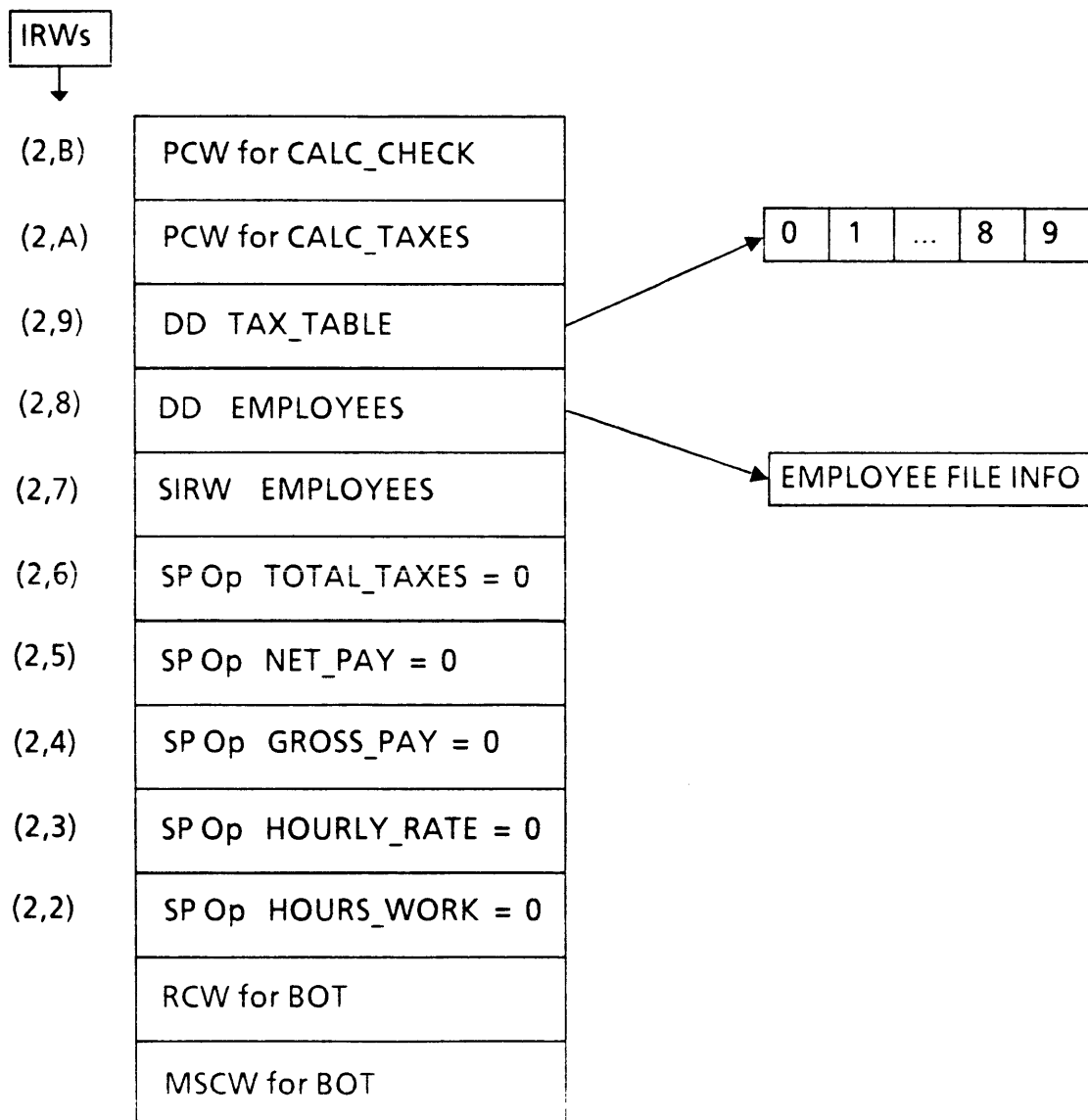


Figure 9-11 Control Words in Stack for Declarations in Sample ALGOL Program

## A SERIES AND B 5000/B 6000/B 7000 CONCEPTS CONTROL WORDS

### *Indirect Reference Word*

An Indirect Reference Word (IRW) or Normal Indirect Reference Word (NIRW) is the address of a word in the stack.

An IRW consists of two numbers in the form ( *x*, *y* ), sometimes called an "address couple". The significance of the two numbers will be discussed in unit 5.

The compiler assigns an IRW to each variable in the program, as the variables are declared.

The Indirect Reference Words for the declarations in lines 1100-1500 and 2100 of the sample ALGOL program are printed to the left of the stack in Figure 9-11.

A Stuffed Indirect Reference Word (SIRW) is the address of data in this stack, or in another stack. For example, an SIRW appears on the stack when the program is passing parameters to a procedure or another program.

### *Data Descriptor*

A Data Descriptor (DD) is the address of a file or data structure that is stored outside the stack. A Presence Bit indicates whether the data is currently in memory or on disk.

Each file declared in a program will have a DD in the stack, which will point to the file attributes and buffers elsewhere in memory.

A Data Descriptor will be placed on the stack for each ALGOL array or COBOL table (with an OCCURS clause) declared in the program. The DD will point to the table of values, located outside the stack.

Figure 9-11 shows the Data Descriptors placed on the stack for the declarations in lines 1300-1400 of the sample ALGOL Program. In addition to the DD, an SIRW has been built for the EMPLOYEES file, to point to an entry in the Segment Dictionary that is required for file handling.

## A SERIES AND B 5000/B 6000/B 7000 CONCEPTS CONTROL WORDS

### *Program Control Word*

A Program Control Word (PCW) is the address of the first object code instruction in a paragraph or procedure. The PCW consists of an object code segment number, and a word and byte number within that segment.

COBOL places a PCW on top of the stack for each paragraph performed.

ALGOL puts a PCW on the stack for each procedure declared, so that the procedure can be located when it is invoked by name later.

Figure 9-11 shows the Program Control Words added to the stack for the procedure declaration at lines 1500 and 2100 in the sample ALGOL program.

### *Software Control Word*

A software control word may be placed on the stack when a procedure is invoked, to indicate that declarations which refer to memory outside the stack are located below. These declarations will require the MCP to do additional memory deallocation later when they are removed from the stack.

In the sample ALGOL program, an SCW will be used to indicate that a file and an array were declared in the program.

## A SERIES AND B 5000/B 6000/B 7000 CONCEPTS CONTROL WORDS

### Practice

Match the terms on the left with the descriptions on the right.

- |           |                          |                                                                                                       |
|-----------|--------------------------|-------------------------------------------------------------------------------------------------------|
| _____ 1.  | Return Control Word      | a. A word placed on the stack when a procedure is invoked, to maintain the program history.           |
| _____ 2.  | Indirect Reference Word  | b. A word that contains the address of an object code segment.                                        |
| _____ 3.  | Single Precision Operand | c. A word that specifies the address where control should return after a procedure has been executed. |
| _____ 4.  | Presence Bit             | d. A portion of a word used by the system in checking for read and write errors.                      |
| _____ 5.  | Program Control Word     | e. A portion of a word that indicates the type of information contained in the word.                  |
| _____ 6.  | Segment Descriptor       | f. A word that contains the address of an item in the stack.                                          |
| _____ 7.  | Tag Field                | g. A word that contains the address of data outside the stack.                                        |
| _____ 8.  | Parity Bit               | h. A portion of some descriptors, that indicates whether the item referenced is in memory or on disk. |
| _____ 9.  | Data Descriptor          | i. A word that contains the address of the first object code instruction in a paragraph or procedure. |
| _____ 10. | Mark Stack Control Word  | j. A word that contains the value of a numeric variable declared in a program.                        |



**A SERIES AND B 5000/B 6000/B 7000 CONCEPTS  
STACK ARCHITECTURE CONCEPTS**

***UNIT 5***

***DISPLAY REGISTERS***

**Objective**

Recognize the purpose of the display registers.

**Purpose**

Familiarity with display registers is necessary when reading reference manuals and program dumps.

**Resources**

A Series System Architecture, Volume 2, Section 1 - Data Structures

# A SERIES AND B 5000/B 6000/B 7000 CONCEPTS DISPLAY REGISTERS

## Display Registers

Registers are storage locations in the processor with designated purposes. Depending on their defined function, registers may contain data, object code, or addresses of data or object code. Units 5 and 6 describe some of the major registers and their significance.

The display registers, or D registers, are a series of registers called D[0] through D[15]. Their functions are described below.

### *D[0] Register*

D[0] always contains the memory address of the segment dictionary for the MCP. The ?SC Cande command displays this address.

Figure 9-12 illustrates D[0] pointing to the segment dictionary for the MCP.

### *D[1] Register*

The D[1] register usually contains the memory address of the currently active program's segment dictionary, as shown in Figure 9-12.

When the MCP is executing in a program stack (for example, producing a program dump), D[1] points to a location in that stack.

### *D[2] Register*

D[2] always contains the memory address of the currently active program's stack. It points at the Mark Stack Control Word where the program began executing, just above the MCP accounting area, as shown in Figure 9-12.

# A SERIES AND B 5000/B 6000/B 7000 CONCEPTS DISPLAY REGISTERS

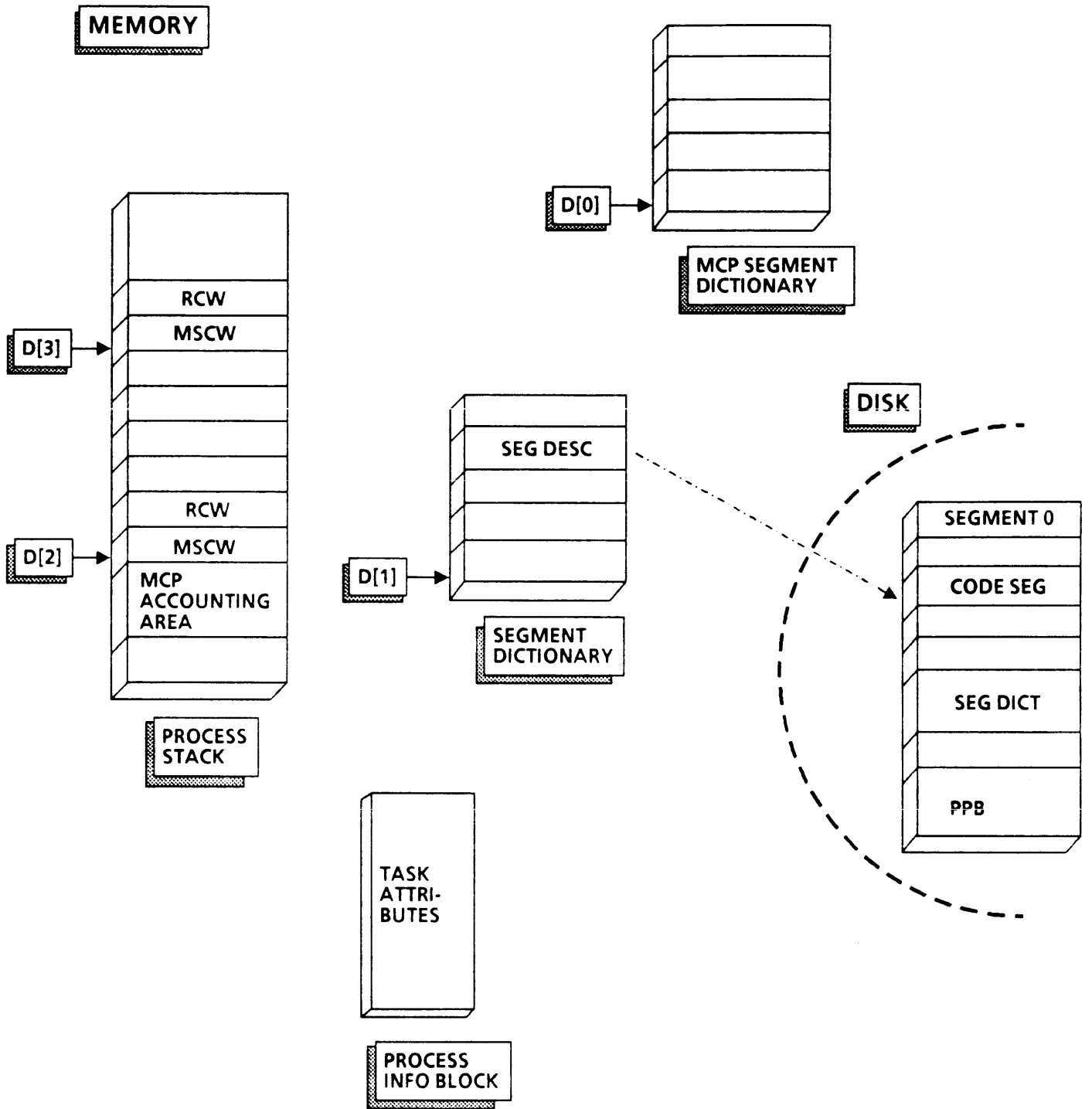


Figure 9-12 Display Registers used in Program Execution

## A SERIES AND B 5000/B 6000/B 7000 CONCEPTS DISPLAY REGISTERS

### *D[3] - D[15] Registers*

D[3] through D[15] point at Mark Stack Control Words in the currently active program's stack, to control the addressing environment, or lexicographical level, of the program.

As defined in Unit 4, addresses of items in the stack are specified by Indirect Reference Words or address couples of the form ( *x* , *y* ).

The left digit, *x*, indicates the number of a D register, and also specifies the lexicographical ("lex") level of the referenced item.

The right digit, *y*, specifies the offset in words from the D Register where the referenced item can be found in the stack.

Example:

In Figure 9-11, D[2] would point to the MSCW for beginning-of-task.

The operand HOURLY\_RATE has address (2,3) because it is 3 words up from D[2] on the stack.

HOURLY\_RATE is declared at lex level 2, or relative to D[2].

ALGOL and COBOL programs use these registers in different ways, as described in the following text.

### *COBOL Use of D Registers*

The Data Division of a COBOL program is always relative to D[2], so all the data items and files will have addresses of ( 2 , *y* ).

When the Procedure Division is entered, an MSCW and RCW are placed on the stack. The D[3] register points to this MSCW.

COBOL programs do not use D registers above D[3] unless they call MCP procedures such as SORT.

Figure 9-12 illustrates the D registers used when a COBOL program is executing.

## A SERIES AND B 5000/B 6000/B 7000 CONCEPTS DISPLAY REGISTERS

### *ALGOL Use of D Registers*

ALGOL is a block-structured language, which means that some variables may be declared within procedures (local to the procedure), rather than at the beginning of the program. The addresses of these local variables are assigned when the variables are declared.

ALGOL programs begin executing at D[2], so variables declared immediately after the BEGIN statement have addresses of ( 2 , y ), as in Figure 9-11.

When a procedure is invoked by mentioning its name, it executes at one lex level higher than the level where it was declared.

An MSCW and RCW are placed on the stack when the procedure is invoked.

The appropriate D register points to the MSCW.

Any variables declared within the procedure have addresses relative to this D register.

Examples:

In the sample ALGOL program in Figures 9-10, when the procedure CALC\_CHECK is invoked at line 3600, the following words are placed on the stack, as in Figure 9-13.

An MSCW and RCW are placed on the stack. The SCW at (2,C) indicates that a file and array were declared below, for memory deallocation later.

The procedure executes at D[3], because it was declared at D[2].

The real variables REG\_PAY and OT\_PAY declared at line 2300 have addresses ( 3 , 2 ) and ( 3 , 3 ) respectively. CALC\_OT is at ( 3 , 4 ).

If CALC\_OT is invoked at line 2900:

An MSCW and RCW are placed on the stack.

D[4] points to the MSCW, as in figure 9-14.

**A SERIES AND B 5000/B 6000/B 7000 CONCEPTS  
DISPLAY REGISTERS**

3600 CALC\_CHECK;

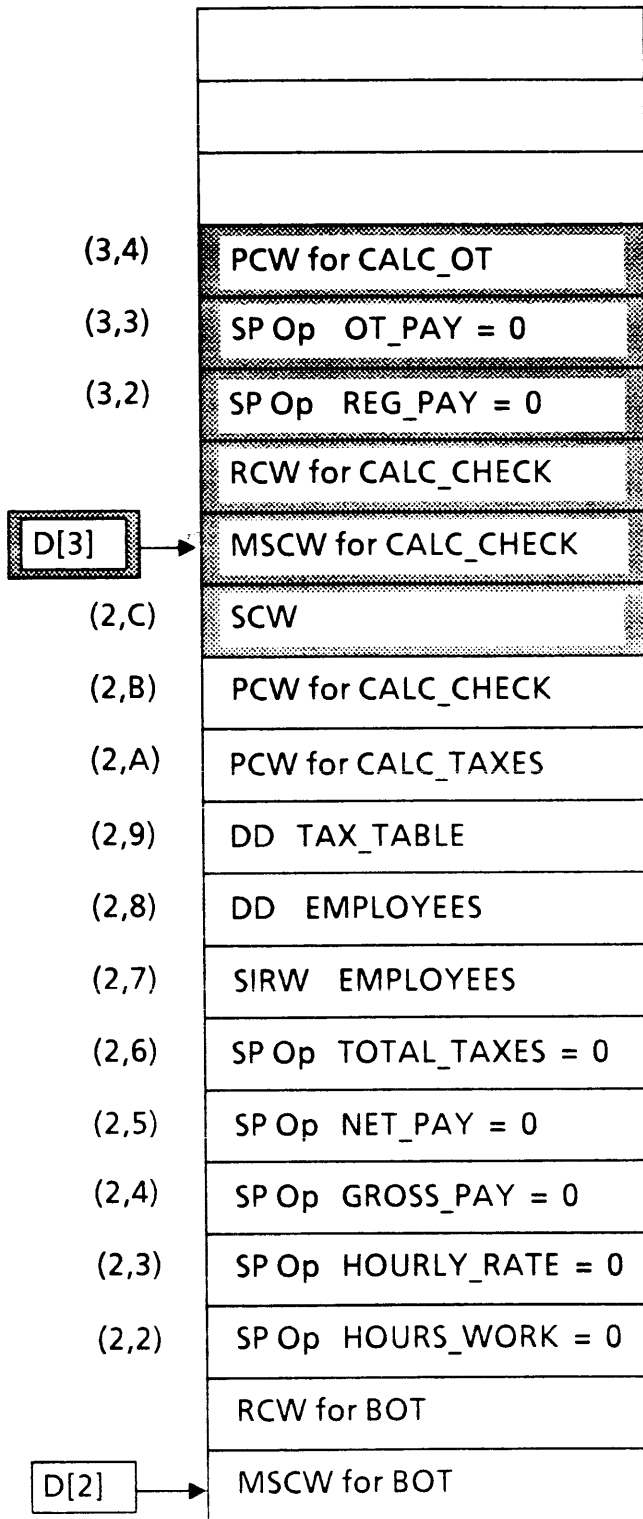


Figure 9-13 Stack after Invocation of Procedure CALC\_CHECK

**A SERIES AND B 5000/B 6000/B 7000 CONCEPTS  
DISPLAY REGISTERS**

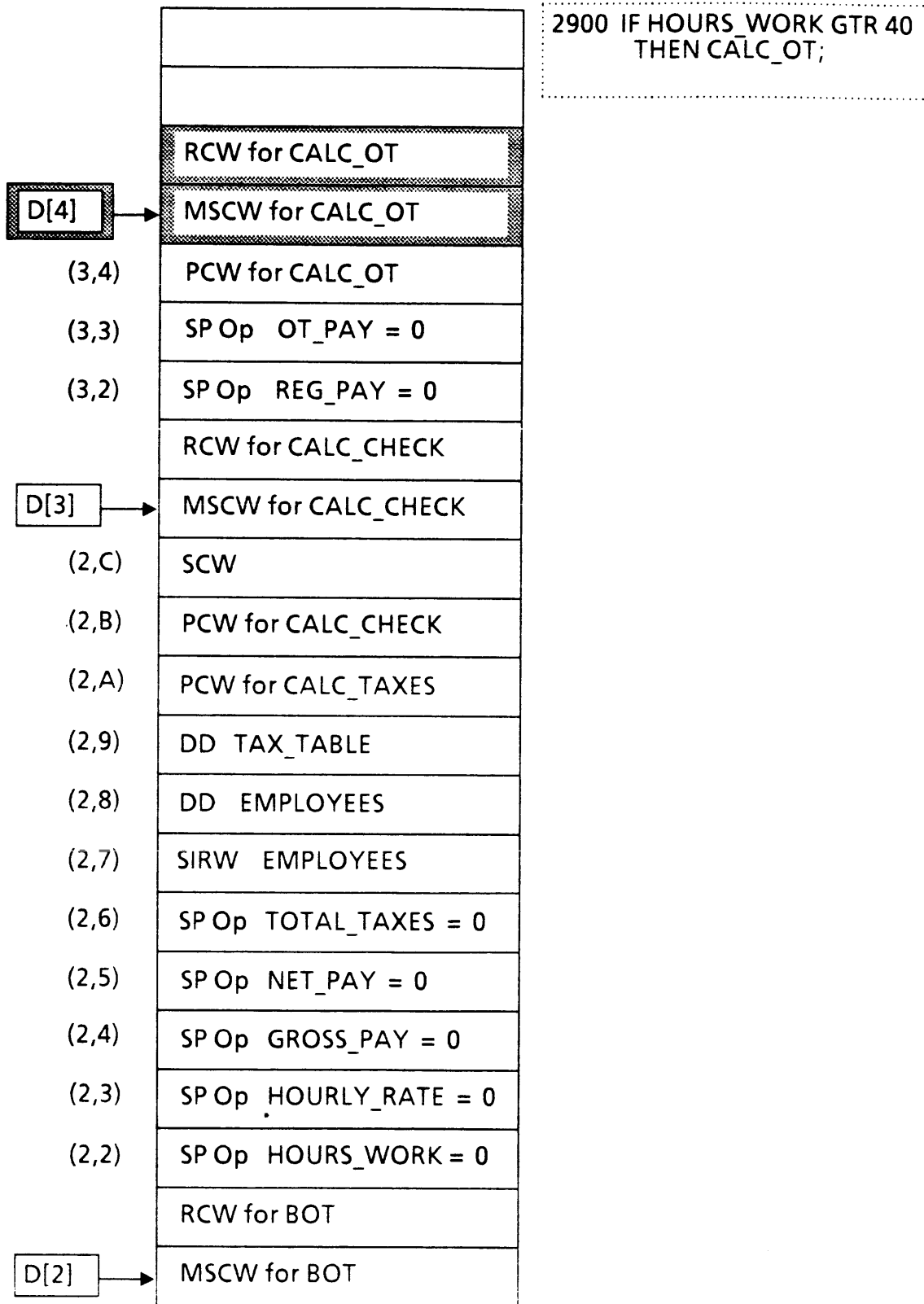


Figure 9-14 Stack after Invocation of Procedure CALC\_OT

## A SERIES AND B 5000/B 6000/B 7000 CONCEPTS DISPLAY REGISTERS

When the end of a procedure is reached, all words that were added to the stack for the procedure are removed from the stack, and the program returns to the next statement after the invocation.

### Example:

At the end of `CALC_OT` at line 2700, the `MSCW` and `RCW` are removed from the stack, as in Figure 9-13.

Control returns to line 3000, the next statement after the invocation.

The program is again executing at `D[3]`.

### *Addressing Environments*

Local variables can be referenced only in the procedure where they are declared. The `D` registers control the addressing environment, by controlling which variables may be accessed at each point in the program.

### Example:

In the sample ALGOL program in Figures 9-10, the procedure `CALC_TAXES` is invoked at line 3100. The stack is built as shown in Figure 9-15.

An `MSCW` and `RCW` are placed on the stack.

`D[3]` now points at this `MSCW`, because the procedure was declared at `D[2]`.

The real variables `FICA_TAX` and `FED_TAX` declared at line 1700 have addresses ( 3 , 2 ) and ( 3 , 3 ) respectively.

None of the declarations in `CALC_CHECK` can be referenced while in this procedure, because they are not in this environment.

At the end of `CALC_TAXES`, the words that were added to the stack for this procedure are removed, and `D[3]` is restored as in Figure 9-13.



## A SERIES AND B 5000/B 6000/B 7000 CONCEPTS DISPLAY REGISTERS

### *Mark Stack Control Word Linkages*

Mark Stack Control Words contain two types of linkages that are involved in controlling the addressing environment and program history.

The **history link** points back to the previous MSCW on the stack. This is used when words are removed from the stack at the end of a procedure.

The **address environment link, or displacement**, points to the previous D register on the stack. This is used when the program is reactivated after being suspended by the MCP.

Figure 9-15 shows the MSCW linkages when the procedure `CALC_TAXES` is executing.

### *F Register*

The F register points to the MSCW for the procedure that is currently executing.

In ALGOL, the F register points to the same word as the highest D register in use, as in Figure 9-15.

While a COBOL program is executing in the Procedure Division, the F register points to the same word as `D[3]`, where the Procedure Division was entered.

### *Top-of-Stack Control Word (TOSCW)*

When a stack is inactive, a TOSCW is placed at the base of the stack, to store information needed to reactivate the program later. The information stored relates to the current top of the stack, such as the setting of the F register, and the current lex level.

**A SERIES AND B 5000/B 6000/B 7000 CONCEPTS  
DISPLAY REGISTERS**

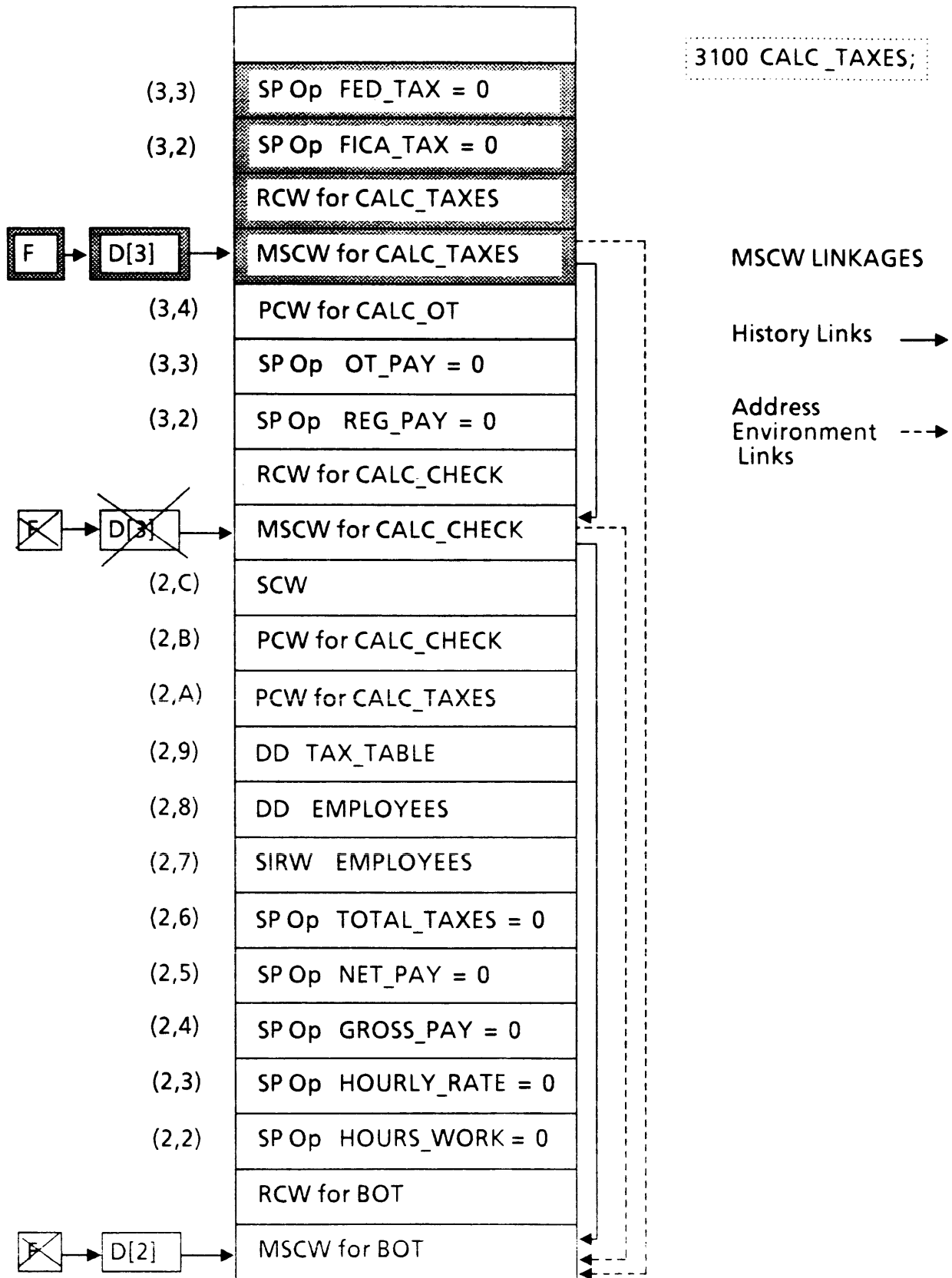


Figure 9-15 Stack after Invocation of Procedure CALC\_TAXES

**A SERIES AND B 5000/B 6000/B 7000 CONCEPTS  
STACK ARCHITECTURE CONCEPTS**

***UNIT 6***

***TOP-OF-STACK REGISTERS***

**Objectives**

Recognize the purpose of the top-of-stack registers.

Identify the operators used in basic calculations.

**Purpose**

Familiarity with top-of-stack registers and operators is necessary when reading reference manuals, compile listings, and program dumps.

**Resources**

A Series System Architecture Reference Manual, Volume 2, Section 3 - Operator Set and Common  
Actions

# A SERIES AND B 5000/B 6000/B 7000 CONCEPTS

## TOP-OF-STACK REGISTERS

### Stack Limit Registers

The Base of Stack Register and Limit of Stack Register are called Stack Limit Registers, because they define the memory boundaries of the currently active process stack.

#### *Base of Stack Register (BOSR)*

The Base of Stack Register contains the memory address of the very first word of the active stack, below the MCP accounting area. This is illustrated in Figure 9-16, which shows the register settings for the sample ALGOL program as the level 2 declarations at lines 1100-1200 are being built on the stack.

#### *Limit of Stack Register (LOS R)*

The Limit of Stack Register contains the memory address of the very last (topmost) word allocated for the active stack, as shown in Figure 9-16.

### Top-of-Stack Registers

The top-of-stack registers, named S, A, B, X, and Y, are all related to the currently active process stack.

#### *S Register*

The S register contains the memory address of the last (topmost) valid word in the active stack. The contents of the S register change as words are added and deleted on the stack.

#### *A and B Registers*

Programs are executed in processor registers, not in memory. Most calculations are done in the A and B registers, as described later in this unit. Logically, the A and B registers are located at the top of the active stack, as in Figure 9-16. Actually, the A and B registers are in the processor, and the stack is in memory.

#### *X and Y Registers*

If Double Precision Operands are used in calculations, the X register is used in conjunction with the A register, and the Y register with the B register, to hold the two words of each operand. This is illustrated in Figure 9-16.

**A SERIES AND B 5000/B 6000/B 7000 CONCEPTS  
TOP-OF-STACK REGISTERS**

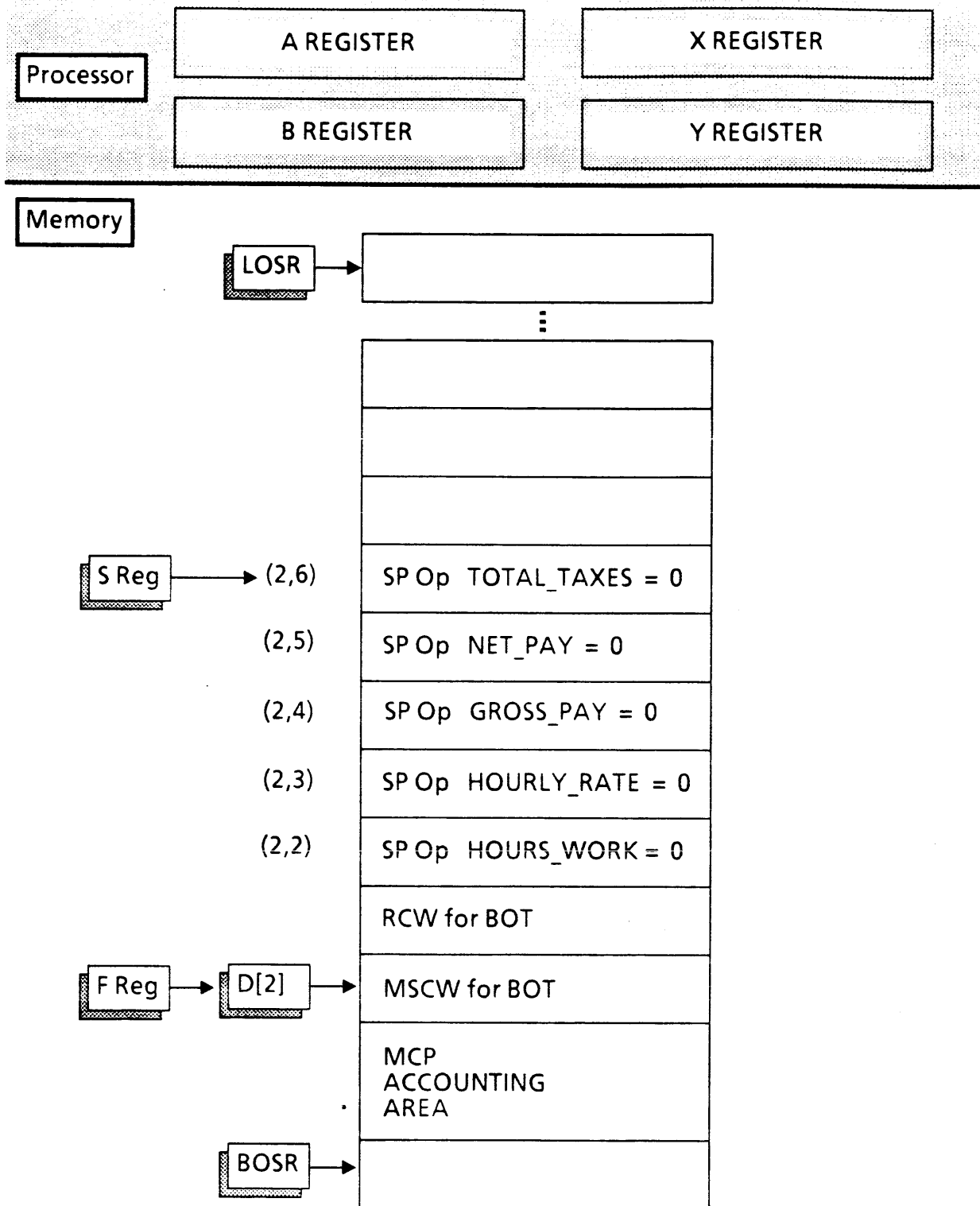


Figure 9-16 Top-of-Stack and Stack Limit Registers during Stack Building in Sample ALGOL Program

# A SERIES AND B 5000/B 6000/B 7000 CONCEPTS

## TOP-OF-STACK REGISTERS

### Execution of Calculations

The A Series systems execute calculations by putting the operands (numbers to be operated on) in the A and B registers, doing the calculation, and leaving the result in the B register. The compilers and hardware operators are designed to use these registers as described below.

### *Operators used in Calculations*

The object code that compilers generate consists of a series of hardware operators and operands. The operators used in simple calculations are detailed below.

<b>ADD</b>	Adds, subtracts, multiplies, or divides the operands in the A and B registers. Leaves the result in the B register, with the A register not occupied by an operand.
<b>SUBT</b>	
<b>MULT</b>	
<b>DIVD</b>	
<b>VALC ( x , y )</b>	Value Call reads the value at the specified stack address, and puts the value in the A register. If there was an operator in the A register before this, it is moved to the B register.
<b>NAMC ( x , y )</b>	Name Call places the specified stack address in the A register. If there was an operator in the A register before this, it is moved to the B register.
<b>STOD</b>	Store Destructive stores the contents of the B register at the address given by the A register, and then destroys the contents of both A and B.
<b>LT8</b>	These operators generate literal values 8, 16, or 48 bits long as required for the calculation. The literal value is placed in register A. If there was an operator in register A before this, it is moved to the B register.
<b>LT16</b>	
<b>LT48</b>	

## A SERIES AND B 5000/B 6000/B 7000 CONCEPTS TOP-OF-STACK REGISTERS

### *Reverse Polish Notation*

The compilers convert calculations into a form called Reverse Polish Notation, so that the object code generated will execute efficiently. Reverse Polish Notation is based on the rules of precedence used in algebra. Its purpose is to rewrite the calculation so that it can be executed from left to right, without any parentheses to group the operands.

Example from the sample ALGOL program in Figures 9-10:

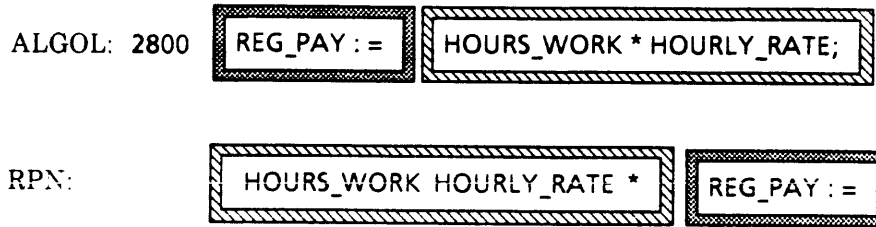


Figure 9-17 Reverse Polish Notation

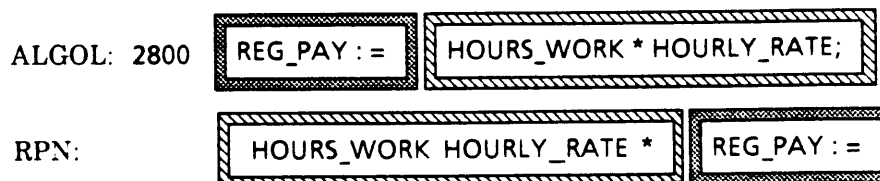
Execution: Read left to right. When you reach an operator (for example, \*), go back to the left to get the appropriate number of operands (2 for multiplication). Perform the calculation, which will probably create a new operand. Then continue to the right.

This notation was invented by a Polish logician named Lukasiewicz, and is called reverse because the operands are stated before the operators.

## A SERIES AND B 5000/B 6000/B 7000 CONCEPTS TOP-OF-STACK REGISTERS

Example : Assume that the sample ALGOL program is processing the paycheck for an employee whose HOURLY\_RATE is \$8.00. The employee had 45 HOURS\_WORK this week. Figure 9-18 shows the stack after executing lines 3500 and 3600, just before the calculation at line 2800.

The object code that is generated by the compiler for the calculation at line 2800 is based on the Reverse Polish Notation for the ALGOL statement.



<u>Object Code</u>	<u>Action</u>	<u>Registers</u>
VALC ( 2 , 2 )	Put the value from ( 2 , 2 ) into the A register. (This is HOURS_WORK, as shown in Figure 9-18).	A <span style="border: 1px solid black; padding: 2px 10px;">45</span> B <span style="border: 1px solid black; padding: 2px 10px;"> </span>
VALC ( 2 , 3 )	Put the value from ( 2 , 3 ) into the A register (this is HOURLY_RATE). Move the value from A into B.	A <span style="border: 1px solid black; padding: 2px 10px;">8</span> B <span style="border: 1px solid black; padding: 2px 10px;">45</span>
MULT	Multiply the contents of the A and B registers, put the result in B, and leave A not occupied.	A <span style="border: 1px solid black; padding: 2px 10px;"> </span> B <span style="border: 1px solid black; padding: 2px 10px;">360</span>
NAMC ( 3 , 2 )	Put the address ( 3 , 2 ) in the A register (the address of REG_PAY).	A <span style="border: 1px solid black; padding: 2px 10px;">(3,2)</span> B <span style="border: 1px solid black; padding: 2px 10px;">360</span>
STOD	Store the contents of B (the result of the calculation) at the address given by A, to update REG_PAY. Destroy the contents of A and B.	A <span style="border: 1px solid black; padding: 2px 10px;"> </span> B <span style="border: 1px solid black; padding: 2px 10px;"> </span>



**A SERIES AND B 5000/B 6000/B 7000 CONCEPTS  
TOP-OF-STACK REGISTERS**

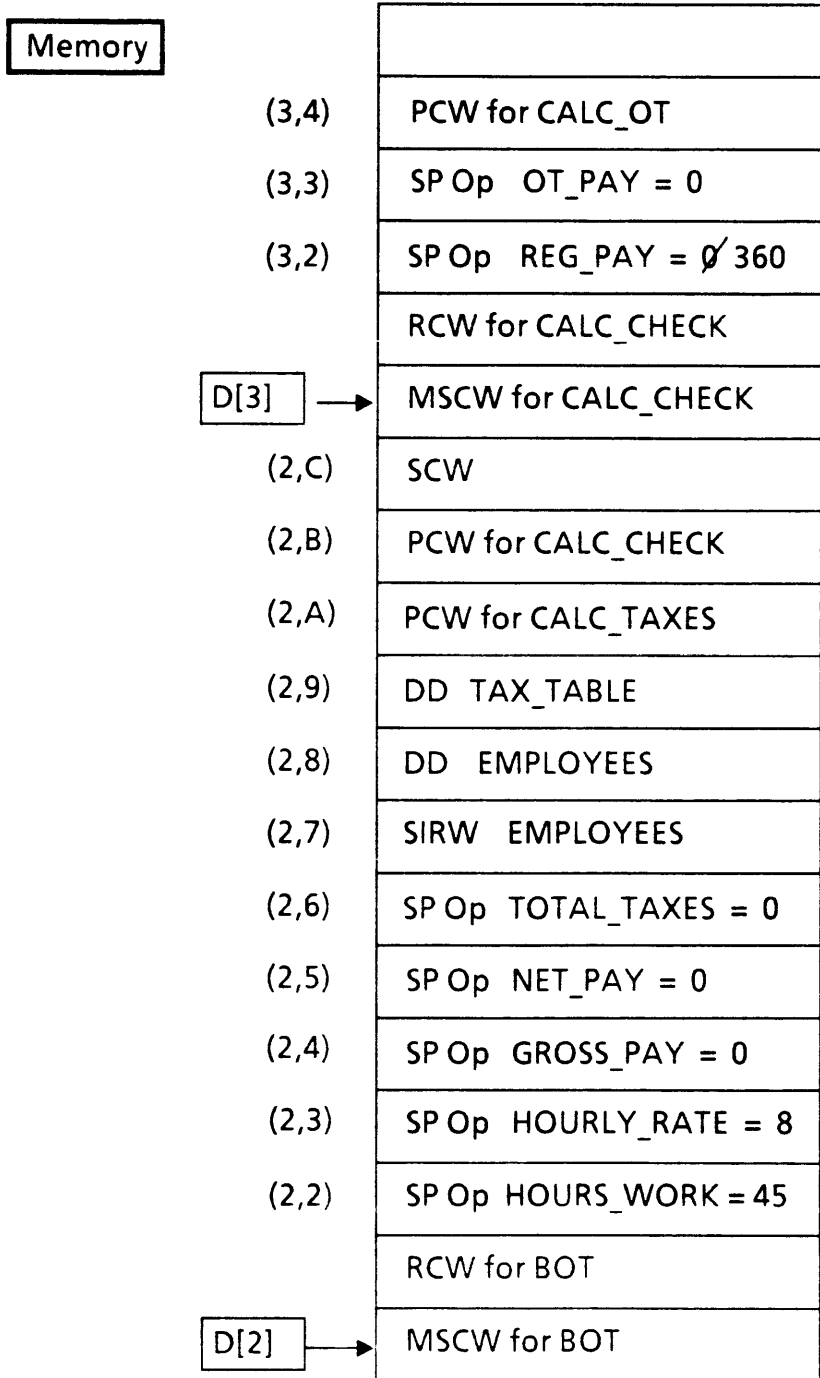
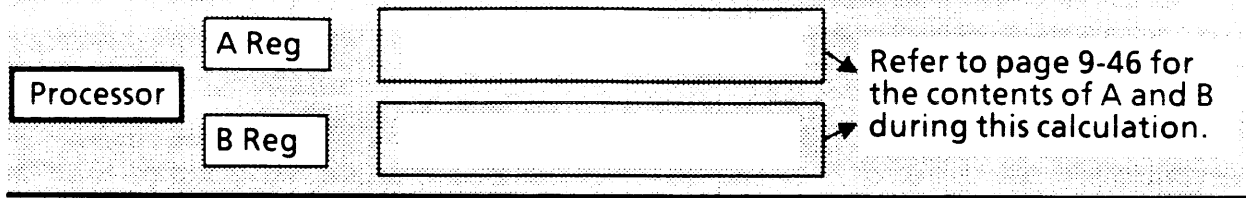


Figure 9-18 Stack during Execution of Calculation at line 2800

## A SERIES AND B 5000/B 6000/B 7000 CONCEPTS TOP-OF-STACK REGISTERS

### Stack Adjustments

Calculations involving more than 2 operands may require more storage words than just registers A and B. In these cases, the extra operands are automatically **pushed** onto the top of the stack, and **popped** off later when they are needed.

Example: The calculation at line 2600 of the sample ALGOL program uses several operands. For the employee who had 45 HOURS\_WORK, this calculation would execute as shown below.

ALGOL: 2600    OT\_PAY :=    HOURLY\_RATE \* .5    \*    (HOURS\_WORK - 40);

RPN:    HOURLY\_RATE .5 \*    HOURS\_WORK 40 -    \*    OT\_PAY :=

<u>Object Code</u>	<u>Action</u>	<u>Registers</u>
VALC ( 2 , 3 )	Put the value from ( 2 , 3 ) into the A register. (This is HOURLY_RATE, as shown in Figure 9-19).	A <span style="border: 1px solid black; padding: 2px;">8</span> B <span style="border: 1px solid black; padding: 2px;"> </span>
LT16 for .5	Generate the literal .5, and put it into the A register. Move the value from A into B.	A <span style="border: 1px solid black; padding: 2px;">.5</span> B <span style="border: 1px solid black; padding: 2px;">8</span>
MULT	Multiply the contents of the A and B registers, put the result in B, and leave A not occupied.	A <span style="border: 1px solid black; padding: 2px;"> </span> B <span style="border: 1px solid black; padding: 2px;">4</span>
VALC ( 2 , 2 )	Put the value from ( 2 , 2 ) in the A register (this is HOURS_WORK ).	A <span style="border: 1px solid black; padding: 2px;">45</span> B <span style="border: 1px solid black; padding: 2px;">4</span>

## A SERIES AND B 5000/B 6000/B 7000 CONCEPTS TOP-OF-STACK REGISTERS

<u>Object Code</u>	<u>Action</u>	<u>Registers</u>
LT8 for 40	Generate the literal 40, and put it into the A register. Move the value from A into B. At this time the value from B is <b>pushed</b> onto the top of the stack.	A <input type="text" value="40"/> B <input type="text" value="45"/>
SUBT	Subtract the contents of the A and B registers, put the result in B, and leave A not occupied.	A <input type="text"/> B <input type="text" value="5"/>
MULT	Multiply requires 2 operands, but only the B register is occupied. The value that was pushed onto the stack earlier is <b>popped</b> back into the B register because it is needed now.	A <input type="text" value="5"/> B <input type="text" value="4"/>
	Multiply the contents of the A and B registers, put the result in B, and leave A not occupied.	A <input type="text"/> B <input type="text" value="20"/>
NAMC (3,3)	Put the address (3,3) in the A register.	A <input type="text" value="(3,3)"/> B <input type="text" value="20"/>
STOD	Store the value in B at the address in A, and destroy the contents of A and B.	A <input type="text"/> B <input type="text"/>

**A SERIES AND B 5000/B 6000/B 7000 CONCEPTS  
TOP-OF-STACK REGISTERS**

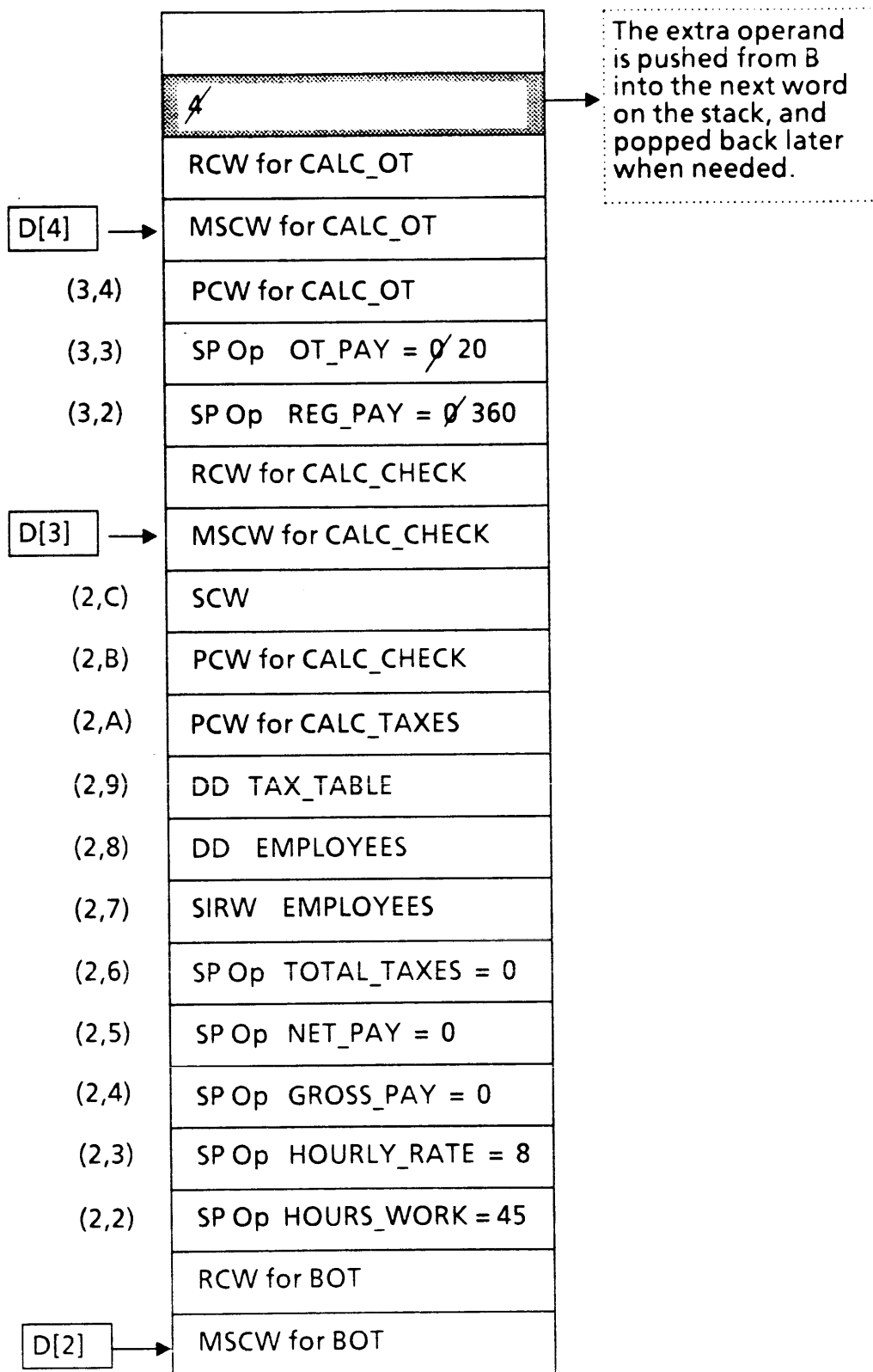


Figure 9-19 Stack during Execution of Calculation at line 2600

## A SERIES AND B 5000/B 6000/B 7000 CONCEPTS TOP-OF-STACK REGISTERS

The S register is adjusted when the stack is pushed or popped, because a different word is now the topmost valid word on the stack.

Stack adjustments can sometimes be avoided by rewriting the calculation with the higher priority algebraic operations on the left.

Example: The calculation at line 2600 could be written to execute more efficiently, without the push and pop of the stack, by rearranging the statement to read:

```
2600 OT_PAY := (HOURS_WORK - 40) * HOURLY_RATE * .5;
```

The Reverse Polish Notation for this calculation would be:

```
HOURS_WORK 40 - HOURLY_RATE * .5 * OT_PAY :=
```

The object code for this calculation would be:

```
VALC ( 2, 2 )  
LT8 for 40  
SUBT  
VALC ( 2, 3 )  
MULT  
LT16 for .5  
MULT  
NAMC ( 3, 3 )  
STOD
```

Push and pop are not required for this calculation, because there are never more than 2 operands to be stored in the A and B registers.

### *Differences in Top-of-Stack Registers between Systems*

Some of the A Series systems improve throughput by using more registers than A and B to hold the current operands.

The A 9 and A 10 systems have a pool of 16 register pairs that correspond to the A and B registers. The MCP will assign from 1 to 8 register pairs to a task while it is executing, so that push and pop are not necessary as often as on other systems

The A 15 and B 7900 Central Processor Modules contain a register file called the Central Data Buffer, which includes 32 registers capable of holding double-precision operands. These registers will be used to store the current operands, in a manner similar to the A and B registers.

## A SERIES AND B 5000/B 6000/B 7000 CONCEPTS TOP-OF-STACK REGISTERS

### *Summary of the Sample ALGOL Program*

Below is a summary of the execution of the sample ALGOL program shown in Figures 9-10.

1. At line 1000, BEGIN causes an MSCW and RCW to be placed on the stack above the MCP accounting area. D[2] points to the MSCW.
2. Lines 1100-1500, and 2100 contain declarations that cause the stack to be built up as in Figure 9-11.
3. Line 3500 would read an employee record from the disk file, and move values into HOURS\_WORK, HOURLY\_RATE, etc. Assume that the employee worked 45 hours this week.
4. Line 3600 invokes CALC\_CHECK, and the stack is built as in Figure 9-13. The program is now executing at lex level 3.
5. Line 2800 calculates REG\_PAY, as described on page 9-45.
6. Line 2900 invokes CALC\_OT, and adds to the stack as in Figure 9-14. CALC\_OT executes at lex level 4.
7. Line 2600 calculates OT\_PAY, as on pages 9-47 and 9-48.
8. Line 2700 ends CALC\_OT, and the stack is cut back as in Figure 9-13. Control returns to line 3000.
9. Line 3000 calculates GROSS\_PAY in the A and B registers. The operators are: VALC (2,4), VALC (3,3), ADD, NAMC (2,4), STOD.
10. Line 3100 invokes CALC\_TAXES, and the stack is built as in Figure 9-15. CALC\_TAXES executes at lex level 3.
11. Line 1800 would do the tax calculations in the A and B registers, using GROSS\_PAY and TAX\_TABLE.
12. Line 1900 calculates TOTAL\_TAXES in the A and B registers, using the operators: VALC (3,2), VALC (3,3), ADD, NAMC (2,6), STOD.
13. Line 2000 ends CALC\_TAXES, and the stack is cut back as in Figure 9-13. Control returns to line 3200.
14. Line 3200 calculates NET\_PAY in the A and B registers: VALC (2,4), VALC (2,6), SUBT, NAMC (2,5), STOD.
15. Line 3300 ends CALC\_CHECK, and the stack is cut back as in Figure 9-11. Control returns to line 3700.
16. Line 3700 would print the check, and loop back to read and process another employee.
17. Line 3800 ends the program. The stack will be removed from memory.

**A SERIES AND B 5000/B 6000/B 7000 CONCEPTS  
TOP-OF-STACK REGISTERS**

**Practice**

Match the registers on the left with the functions on the right.

- |          |                   |                                                                                                                        |
|----------|-------------------|------------------------------------------------------------------------------------------------------------------------|
| _____ 1. | BOSR              | a. Contain the operands used by the current operator.                                                                  |
| _____ 2. | S Register        | b. Contain portions of double-precision operands.                                                                      |
| _____ 3. | LOSR              | c. Control the addressing environment for an ALGOL program.                                                            |
| _____ 4. | X and Y Registers | d. Points to the very first word of the currently active stack.                                                        |
| _____ 5. | D[2] Register     | e. Points to the very last word allocated for the currently active stack.                                              |
| _____ 6. | A and B Registers | f. Points to the last (topmost) valid word of the currently active stack.                                              |
| _____ 7. | D[3-15] Registers | g. Points to the word on the stack where the currently active program began executing (above the MCP accounting area). |

**A SERIES AND B 5000/B 6000/B 7000 CONCEPTS  
TOP-OF-STACK REGISTERS**

This page left blank for formatting.



**SECTION 10**

**SYSTEMIDUMPALL - LISTING DISK FILES**

# A SERIES AND B 5000/B 6000/B 7000 CONCEPTS SYSTEM/DUMPALL

## *INTRODUCTION*

### **Section Objective**

List disk files using the system software utility SYSTEM/DUMPALL.

### **Purpose**

You need to know how to use SYSTEM/DUMPALL to verify the integrity of data contained in a disk file.

### **Unit Objectives**

Identify the system software utility SYSTEM/DUMPALL.

Use the LIST command which is part of SYSTEM/DUMPALL.

**A SERIES AND B 5000/B 6000/B 7000 CONCEPTS  
SYSTEM/DUMPALL**

**UNIT 1**

**SYSTEMIDUMPALL - LISTING DISK FILES**

**Objective**

Identify the system software utility SYSTEM/DUMPALL.

Use the LIST command which is part of SYSTEM/DUMPALL.

**Purpose**

You need to know how to use the utility SYSTEM/DUMPALL in order to determine the integrity of your data files.

**Resources**

A Series Systems An Introduction, Section 3 - A User's View of System Functions

A Series System Software Utilities Reference Manual, Section 4 - DUMPALL

# A SERIES AND B 5000/B 6000/B 7000 CONCEPTS SYSTEM/DUMPALL

## What Is SYSTEM/DUMPALL?

SYSTEM/DUMPALL is a system software utility that performs media conversion. It is often referred to as DUMPALL.

The main functions of SYSTEM/DUMPALL are to:

- Copy files from one medium to another.
- Control the dumping of tapes.
- Generate printouts of files.

## Listing Files Using SYSTEM/DUMPALL

SYSTEM/DUMPALL has a LIST command which allows you to produce a graphic display or printout of the contents of a file. The file may be labeled or unlabeled.

Simplified Syntax of LIST Command:

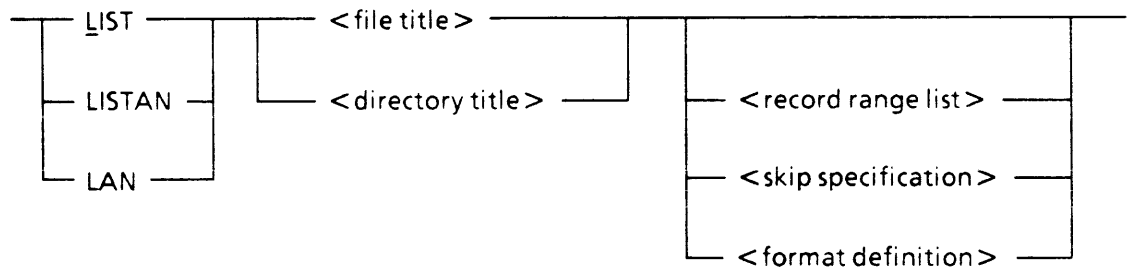


Figure 10-1 SYSTEM/DUMPALL Simplified Syntax

LIST or L will list a file or group of files in EBCDIC format.

LISTAN or LAN will list a file or group of files in the format that corresponds to the INTMODE file attribute of the file.

<record range list> and <skip specification> allows you to specify what portion of a file is to be printed.

<format definition> allows you to specify a particular field within a record along with its format that you want printed.

## A SERIES AND B 5000/B 6000/B 7000 CONCEPTS SYSTEM/DUMPALL

Examples:

```
LIST (USER1)PAYROLL/HOURS/WORKED
```

```
LIST (CONCEPTS)STUDENT/FILE1 ON EDUCATION
```

```
L *BOOK/MARC/ENGLISH REC 1 THRU 10 SKIP + 2 INCL 5
```

```
L (USER2)ACCOUNTS/PAYABLE/= ON PAYPACK REC 10 THRU END
```

```
LIST (USER1)TESTFILE; LIST (USER1)MYDIRECTORY/NEWFILE
```

### SYSTEM/DUMPALL EXECUTION

SYSTEM/DUMPALL can be executed in a **Parameter Mode**, **Card Mode** or an **Interactive Mode**. The DUMPALL execution statement can be entered at the ODT through the MARC screens or in CANDE depending on the mode of execution desired.

#### *Parameter Mode*

When executing DUMPALL in a parameter mode, you are required to provide the input specification for the function DUMPALL is to perform at the time the utility is executed.

#### **ODT Input:**

```
RUN SYSTEM/DUMPALL (" <list command >")
```

Example:

```
RUN SYSTEM/DUMPALL ("LIST (USER1)MYDIRECTORY/MYFILE")
```

#### **MARC Input:**

MARC Home Menu - select UTIL

On Utility Screen - select DALL

On Dumpall Utility Screen - fill in the necessary parameters for the file you want listed.

Example of parameters for Dumpall Utility Screen (DALL screen).

```
LIST (USER1)MYDIRECTORY/MYFILE
```

## A SERIES AND B 5000/B 6000/B 7000 CONCEPTS SYSTEM/DUMPALL

### *Card Mode*

When executing DUMPALL in a card mode, you place the required parameters in a disk file. When the utility is executed, it will read the disk file to determine what file(s) are to be listed.

#### **ODT Input:**

```
RUN SYSTEM/DUMPALL ("CARD"); FILE CARD (TITLE = MYCARDFILE)
```

### *Interactive Mode Input*

Interactive mode of DUMPALL allows you to enter in DUMPALL commands one at a time and receive the response to that command at your terminal. DUMPALL runs as an on-line program in interactive mode.

#### **From a MARC terminal, enter:**

Display the Dumpall Utility Screen (DALL screen).

Enter INTER as the requested parameter.

A message PLEASE ENTER DUMPALL COMMAND will appear on your terminal.

Enter the LIST command.

```
Example: LIST MYFILE RECORD 10 THRU 30
```

The file attributes will be displayed on the terminal.

After you have looked at the file attributes, transmit CONT to inform DUMPALL to continue the display with the next piece of information.

The first record of the file will be displayed.

Transmit CONT to see the next record of the file.

By transmitting CONT, one record after another will be displayed until the end of file is reached. You may terminate this listing of this file by entering QUIT.

To end Interactive Dumpall, enter QUIT.

## A SERIES AND B 5000/B 6000/B 7000 CONCEPTS SYSTEM/DUMPALL

From a CANDE terminal, enter:

```
RUN $SYSTEM/DUMPALL ("INTER")
```

A message PLEASE ENTER DUMPALL COMMAND will appear on your terminal.

You would enter DUMPALL commands as described for a MARC terminal.

DUMPALL defaults to listing the file on your terminal when running interactively. To request a hardcopy when running interactively, add the word PRINT to your list command.

Example: LIST MYFILE RECORD 10 THRU 30 PRINT

SYSTEM/DUMPALL has many other capabilities. All of the other Dumpall capabilities, the full syntax and DUMPALL execution using the Work Flow Language, are covered in the Operations class and the Work Flow and Utilities class.





**SECTION 11**  
**LIBRARIES OVERVIEW**

# A SERIES AND B 5000/B 6000/B 7000 CONCEPTS LIBRARIES OVERVIEW

## *INTRODUCTION*

### **Section Objective**

Recognize the purpose of libraries, intrinsics, and binding.

### **Purpose**

Libraries are used extensively in the A Series system and environmental software, and can also be written for applications.

### **Unit Objectives**

Recognize the purpose of libraries.

Use ODT commands associated with libraries and intrinsics.

# **A SERIES AND B 5000/B 6000/B 7000 CONCEPTS LIBRARIES OVERVIEW**

## ***UNIT 1***

### ***LIBRARIES OVERVIEW***

#### **Objective**

Recognize the purpose of libraries.

Use ODT commands associated with libraries and intrinsics.

#### **Purpose**

Libraries are used extensively in the A Series system and environmental software, and can also be written for applications.

#### **Resources**

A Series System Software Utilities, Section 9 - Libraries

A Series Systems An Introduction, Section 2 - Virtual Memory, Stack, and Other System Concepts

# A SERIES AND B 5000/B 6000/B 7000 CONCEPTS LIBRARIES OVERVIEW

## Libraries

A library is a program that provides one or more procedures that can be called by other programs to perform specific functions.

Libraries simplify program creation and maintenance by providing routines that can be shared by many programs on the system.

A library is usually a collection of related routines, such as data conversion routines or mathematical functions, that are required by multiple programs.

The procedures that are provided are called **entry points**, and the programs that call the library are called **user programs**.

## *User-Written Libraries*

Users may write libraries to provide routines that can be accessed by multiple programs. Some of the features of user-written libraries are:

Standard functions, such as plotting and statistics, can be maintained in one file, rather than copied and compiled into every user program that requires them.

Individual users can create their own libraries.

Libraries may be written in ALGOL, COBOL, COBOL74, FORTRAN, NEWP, PL/I, or PASCAL, although COBOL or COBOL74 libraries may contain only 1 entry point corresponding to the Procedure Division.

A user program written in one language may call a library written in another language, if the parameters are compatible in both languages.

Libraries can contain initialization and termination code.

A library can have its own variables, global files, data bases, etc.

A library can call a procedure in another library, as in Figure 11-1.

## A SERIES AND B 5000/B 6000/B 7000 CONCEPTS LIBRARIES OVERVIEW

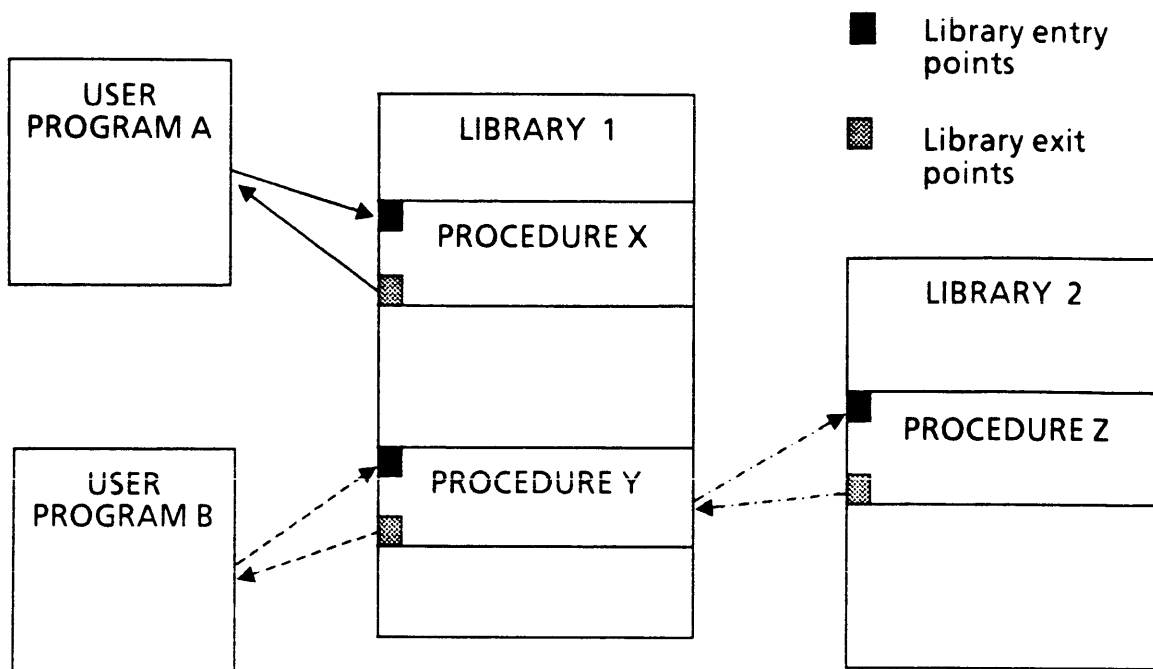


Figure 11-1 Library Linkages

### *Library Execution and Stacks*

When a user program calls a procedure in a library, the MCP links the program and the library.

The user program is suspended.

If the library is not already in the mix, it is executed.

A separate stack is built to store the library's variables and history, as in Figure 11-2.

The library's initialization code is executed, until a **FREEZE** statement is reached. The **FREEZE** makes the procedure entry points available to other programs.

The MCP locates the requested procedure in the library, and links it to the user program.

The procedure executes on top of the user program's stack.

At the end of the procedure, the words for the procedure are deleted from the stack.

The user program resumes.

## A SERIES AND B 5000/B 6000/B 7000 CONCEPTS LIBRARIES OVERVIEW

A library may be written to stay in the mix until all of its user programs have ended, or it may be **frozen** permanently. A frozen library avoids repeated initialization by remaining in the mix, even if it has no user programs linked to it, until a DS (DiScontinue) or THAW ODT command is entered.

The LIBS (Libraries) ODT command displays the names of the libraries currently in the mix.

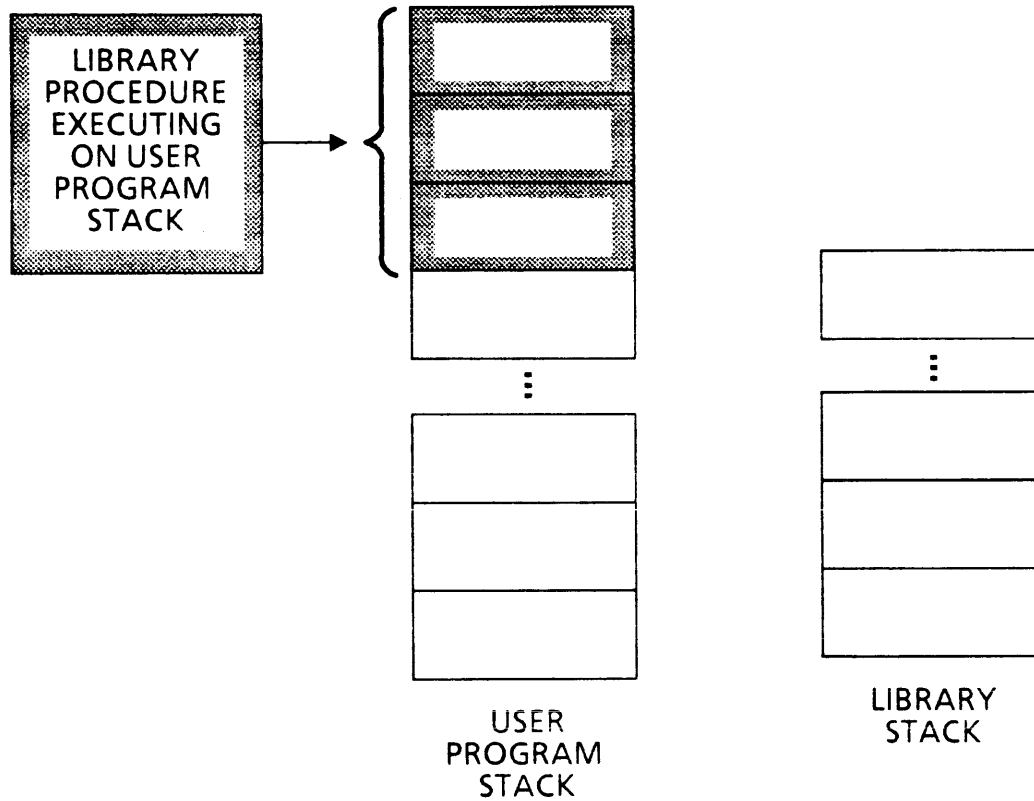


Figure 11-2 Stacks used during Library Execution

## A SERIES AND B 5000/B 6000/B 7000 CONCEPTS LIBRARIES OVERVIEW

### *System Libraries*

The MCP and other system software programs use standard libraries extensively. Some of the standard libraries are:

#### SYSTEM/GENERALSUPPORT

Contains common routines such as mathematical functions (for example, square root, random numbers).

#### SYSTEM/PLISUPPORT

Contains routines required by PL/I programs, as well as some routines used by GENERALSUPPORT. Required even if PL/I is not used on the system.

#### SYSTEM/HELP

Contains procedures used by the HELP utility.

Most programs that access user-written libraries have the titles of the library object code files compiled into them. This is too restrictive for system libraries, because at some sites they may be located on a family other than DISK, or stored in a file with a non-standard name. In order to make the system software more flexible, the MCP maintains a table of functions performed by system libraries, and library titles where the entry points that perform those functions are located.

The SL (System Library) ODT command associates the functions with the libraries, or inquires into the functions and associated library titles.

Examples:

SL HELPSUPPORT = SYSTEM/HELP ON PACK

SL PLISUPPORT = SYSTEM/PLISUPPORT ON TEST

SL GENERALSUPPORT = SYSTEM/GENERALSUPPORT ON DISK

SL displays the functions and associated library titles

Users may also define their own function names through the SL command, so that user-written libraries can be accessed by function. The default access to user-written libraries is by the object code file title.

# A SERIES AND B 5000/B 6000/B 7000 CONCEPTS LIBRARIES OVERVIEW

## Forerunners of Libraries

Before libraries were developed, other techniques were used to combine programming languages within a program, and to provide routines that programs could call. These methods are still used to a limited extent on the A Series systems.

## *Binding*

Binding allows programming languages to be mixed, and individual procedures of programs to be compiled separately from the rest of the program.

A program can be compiled with the bodies of one or more procedures missing from the source code (called "external" to the program).

The procedures can be compiled separately.

The main object code file and the procedure object code file(s) are bound together by SYSTEM/BINDER to form one large object code file, which can be executed.

For user programs, the major advantages of libraries over binding are:

Libraries offer more flexibility in mixing programming languages.

Libraries are easier to maintain, since changes need to be made to the library file only, rather than bound into several object programs.

The main portion of the MCP is bound together with some of its procedures, such as CONTROLLER. This allows these procedures to be compiled separately and then bound into the MCP, instead of compiling the entire MCP just for changes to the procedures.

## *Intrinsics*

Before libraries were implemented, an intrinsics file contained functions that programs could call, such as square root, trigonometric functions, and random number generation.

Most of these functions are now in the GENERALSUPPORT library.

The intrinsics file is still required to be resident on a pack.

SYSTEM/INTRINSICS is the standard name of this file, although it may be changed.

The SI (System Intrinsics) ODT command specifies or inquires into the name of the intrinsics file for the system.

Users may write intrinsic functions in ALGOL, and bind them into the standard intrinsics file, but this is much more restrictive than writing libraries.



**SECTION 12**

**MEMORY MANAGEMENT OVERVIEW**

# A SERIES AND B 5000/B 6000/B 7000 CONCEPTS MEMORY MANAGEMENT OVERVIEW

## *INTRODUCTION*

### **Section Objective**

Recognize the memory management methods available on the A Series systems.

### **Purpose**

Familiarity with memory management methods is necessary when reading reference manuals and managing a system.

### **Unit Objectives**

Identify the purpose of virtual memory.

Identify the memory management methods available on the A Series systems.

Identify the purpose of Memory Disk.

# A SERIES AND B 5000/B 6000/B 7000 CONCEPTS MEMORY MANAGEMENT OVERVIEW

## UNIT 1

### MEMORY MANAGEMENT OVERVIEW

#### Objective

Identify the purpose of virtual memory.

Identify the memory management methods available on the A Series systems.

Identify the purpose of Memory Disk.

#### Purpose

Familiarity with memory management concepts is necessary when reading reference manuals and managing a system.

#### Resources

A Series Systems An Introduction, Section 2 - Virtual Memory, Stacks, and Other System Concepts

A Series System Software Support Reference Manual, Section 9 - Memory Management  
Section 14 - SWAPPER

## A SERIES AND B 5000/B 6000/B 7000 CONCEPTS MEMORY MANAGEMENT OVERVIEW

### Virtual Memory

Virtual Memory is a technique that treats disk space as an extension of main memory, giving the appearance of a larger main memory than actually exists.

As discussed in Section 9, the compilers automatically divide object code programs into variable-length segments, based on the structure of the program. When the program is executed, only the object code segments needed at the time are brought into memory, and the other code segments for the program remain on disk until they are needed.

Data from disk files is also read into memory in variable-length blocks, so that most of the file remains on disk.

The segment descriptors and data descriptors defined in Section 9 are located in the program's stack, and contain the addresses of code and data located elsewhere in memory. Descriptors allow data and object code segments for the same program to be allocated throughout memory.

### Overlayable Memory

Overlayable memory is another aspect of virtual memory. If there is not enough available (unallocated) memory to bring a required segment into memory, the MCP can overlay to free some memory space. Overlaying involves writing the contents of memory areas to a disk pack, and then reallocating those memory areas to store other data. The data that was overlaid to a pack will be brought back into memory later when it is needed.

Structures that can be deallocated to free memory space, such as object code segments and data buffers, are called overlayable, or **nonsave**, memory.

Data areas (for example, file buffers) can be overlaid to a pack as described above and shown in Figure 12-1.

If the MCP needs to reallocate the memory space occupied by an object code segment, it does not overlay the segment to a pack, because it can read the segment from the object code file on disk again later if necessary. The memory space is freed, and the segment descriptor in the segment dictionary is modified to contain the disk address of the segment, which is now absent from memory.

The DL (Disk Location) ODT command can be used to specify the family name where overlay files should be placed.

**A SERIES AND B 5000/B 6000/B 7000 CONCEPTS  
MEMORY MANAGEMENT OVERVIEW**

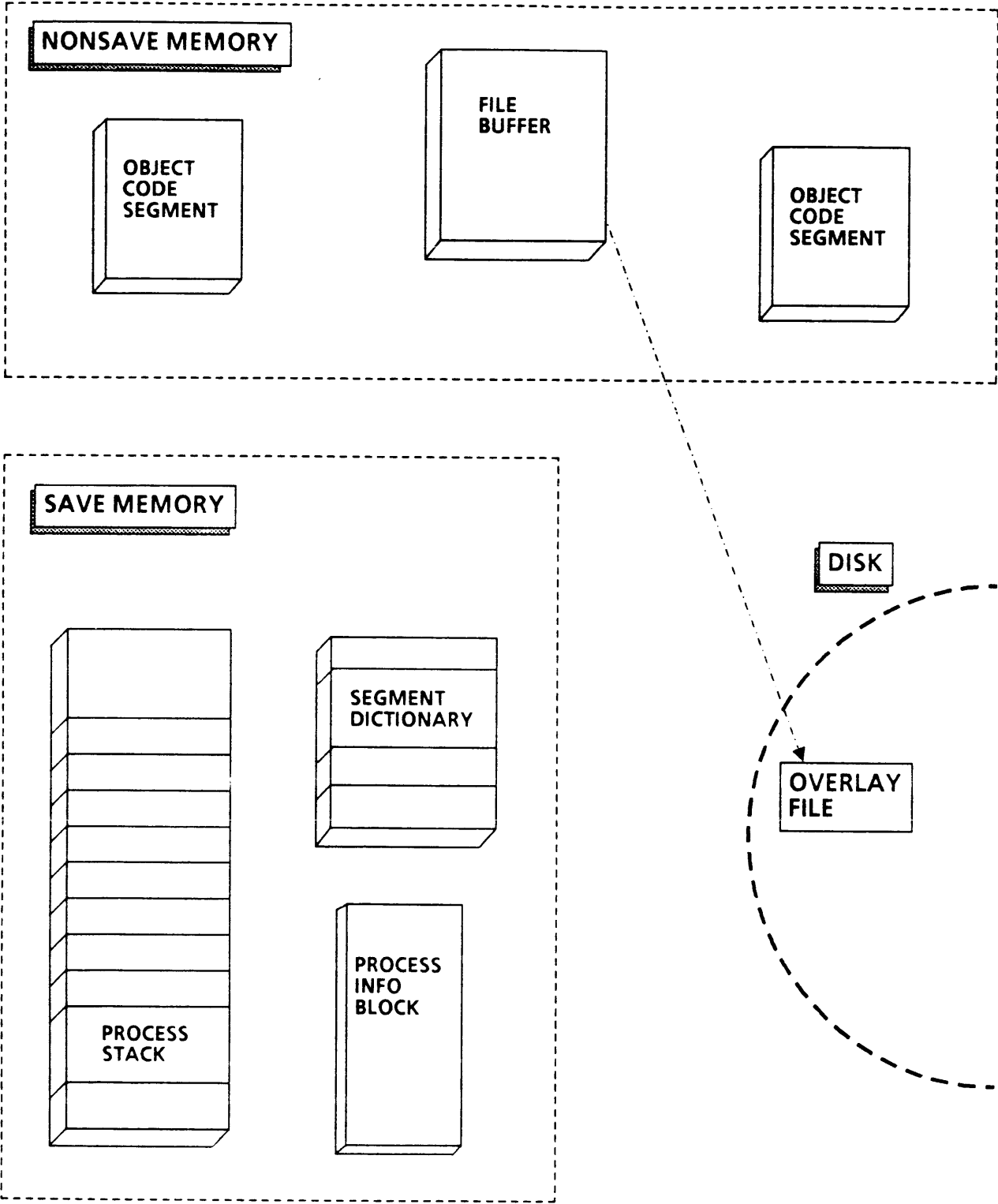


Figure 12-1 Save and Nonsave Memory

## A SERIES AND B 5000/B 6000/B 7000 CONCEPTS MEMORY MANAGEMENT OVERVIEW

### *Save Memory*

**Save** memory areas cannot be overlaid, but are required to be resident in memory until the program associated with them ends. Process Stacks, Segment Dictionaries, and Process Information Blocks are examples of structures that are placed in save memory, as shown in Figure 12-1.

The CU (Core Usage) ODT command displays the number of memory words that are available to be allocated, as well as the number of words currently assigned to nonsave and save memory.

### **Methods of Memory Management**

When there are many programs in the mix, the system may be required to overlay memory frequently as it requires different memory structures for different programs. Excessive overlays may consume system resources and degrade performance (this is called **thrashing**). Several mechanisms are available on A Series systems to manage memory allocation and overlays.

### *On Demand*

On Demand memory management finds memory space when there is not enough contiguous memory available to store a required structure.

First the MCP tries to make enough space by rearranging structures within memory to different locations (core to core overlay).

If memory space must be deallocated to create enough available space, the MCP chooses the structure to be overlaid.

Object code segments are deallocated, and their segment descriptors updated (core to limbo overlay).

Data is overlaid to an overlay file on a pack (core to disk overlay).

## A SERIES AND B 5000/B 6000/B 7000 CONCEPTS MEMORY MANAGEMENT OVERVIEW

### *Working Set Sheriff*

Working Set Sheriff memory management allows the site manager to specify a constant forced overlay rate to be achieved by all tasks on the system. WSSHERRIFF, an invisible independent runner, cyclically examines memory and overlays areas until the specified rate is achieved. This causes overlays to occur in bursts, rather than at random times.

Working Set Sheriff is controlled by factors that are specified by the SF (Set Factors) ODT command.

Factor 1 is OLAYGOAL, the percentage of overlayable memory that is to be overlaid per minute.

The default value of factor 1 is 0, in which case WSSHERRIFF is not used.

When OLAYGOAL is greater than 0, WSSHERRIFF will perform the overlays.

Factor 2 is AVAILMIN, the percentage of total memory to be kept available for use on demand.

The lowest priority job in the mix is suspended when the total amount of available memory drops below one-half AVAILMIN.

AVAILMIN takes effect only if OLAYGOAL is greater than 0.

The default value of factor 2 is 0.

Factor 3 is the scheduling factor, which is used to determine if enough memory is available to initiate the task.

The task's memory estimate from segment 0 is divided by factor 3, and compared to the amount of available memory.

The task will be scheduled if there is not enough memory available to execute at this time, based on this calculation.

The default value of factor 3 is 100%.

## A SERIES AND B 5000/B 6000/B 7000 CONCEPTS MEMORY MANAGEMENT OVERVIEW

Factor 4 is the Memory Priority Factor that controls the memory priority algorithm.

Priority is an attribute of each task, which specifies the relative importance of each task to the MCP. The highest priority is 99, the lowest is 0, and the default is 50.

The default value of factor 4 is 0, which indicates that the memory priority algorithm is not used.

Factor 4 represents the percentage of one second per increment of priority that a low-priority job must wait before it can overlay a memory area owned by a higher priority job.

### **SWAPPER**

SWAPPER is a memory management mechanism designed to service users in a time-sharing environment. SWAPPER manages only specified tasks, usually data communications tasks, rather than the entire mix.

A portion of memory called **swapspace** is allocated for the tasks running under SWAPPER. Each task is assigned one or more contiguous areas within swapspace, so that all memory for a task can be read or written in a single operation.

A disk file called **SYSTEM/SWAPDISK** contains the overlay areas for all tasks running under SWAPPER.

Tasks are swapped from memory to disk when they are interrupted or suspended, or when they have exceeded a specified time-slice.

Tasks that were previously swapped out, but are now ready to run, are swapped from disk to memory when areas are available within swapspace.

The SW (SWAPPER) ODT command initiates SWAPPER, and specifies the required parameters.

SWAPPER is not used in conjunction with MCP/AS.



# A SERIES AND B 5000/B 6000/B 7000 CONCEPTS MEMORY MANAGEMENT OVERVIEW

## Memory Disk

Memory Disk is a mechanism to improve system performance by designating a portion of the system's main memory as a disk unit.

Memory Disk can increase throughput, because the speed of memory access over disk access is significant.

Each unit of Memory Disk is identified by the unit mnemonic MD, a unit number, and a family name.

Memory Disk does not require any changes to application programs or WFL jobs. It is accessed by its family name.

The contents of a Memory Disk unit are vulnerable to power failure and off-line memory reconfiguration, so files stored here must not require recovery.

Overlay files and sort work files are examples of files that could safely be directed to Memory Disk, and may have a large impact on system performance.

Memory Disk is available only on the B 7900 and A Series systems beginning with release 3.6.

**A SERIES AND B 5000/B 6000/B 7000 CONCEPTS  
MEMORY MANAGEMENT OVERVIEW**

This page left blank for formatting.

**SECTION 13**  
**WORK FLOW MANAGEMENT**  
**SYSTEM**

# A SERIES AND B 5000/B 6000/B 7000 CONCEPTS WORK FLOW MANAGEMENT SYSTEM

## **INTRODUCTION**

### **Section Objective**

Use the Work Flow Management System.

### **Purpose**

In order to control programs in the system, you must be able to use the Work Flow Language.

### **Unit Objectives**

Identify the components of the Work Flow Management System.

Use Automatic Display Mode to monitor Work Flow jobs.

Identify the basic syntax of the Work Flow Language.

Identify the Work Flow Language structure.

Use the Work Flow Language file equations.

Compile programs using the Work Flow Language.

Write LIBRARY/MAINTENANCE statements.

**A SERIES AND B 5000/B 6000/B 7000 CONCEPTS  
WORK FLOW MANAGEMENT SYSTEM**

***UNIT 1***

***WORK FLOW MANAGEMENT SYSTEM OVERVIEW***

**Objective**

Identify the components of the Work Flow Management System.

**Purpose**

In order to use the Work Flow Management System, you need to be aware of how the Work Flow Management System functions.

**Resources**

# A SERIES AND B 5000/B 6000/B 7000 CONCEPTS WORK FLOW MANAGEMENT SYSTEM OVERVIEW

## Work Flow Management System

The Work Flow Management System is part of the MCP and is responsible for controlling and monitoring the flow and execution of jobs and tasks in the system (Refer to Figure 13-1).

A task is the execution of a program.

A job is a collection of 1 or more related tasks which are to be run in the sequence specified.

The Work Flow Management System allows tasks to execute in 2 modes.

### Synchronous

Tasks execute serially within the job; one task must finish before another task can begin.

### Asynchronous

Tasks within the same job can be run concurrently in a multiprogramming environment.

The Work Flow Language is used to interface with the Work Flow Management System.

The Work Flow Language is used to:

Write a Work Flow Job.

Control the flow of tasks in a job.

Test for completion of tasks or errors and take appropriate action.

Handle recovery procedures following a Halt/Load.

Minimize operator involvement.

Handle dynamic file equation.

Print all job output together.

WORK FLOW MANAGEMENT SYSTEM OVERVIEW

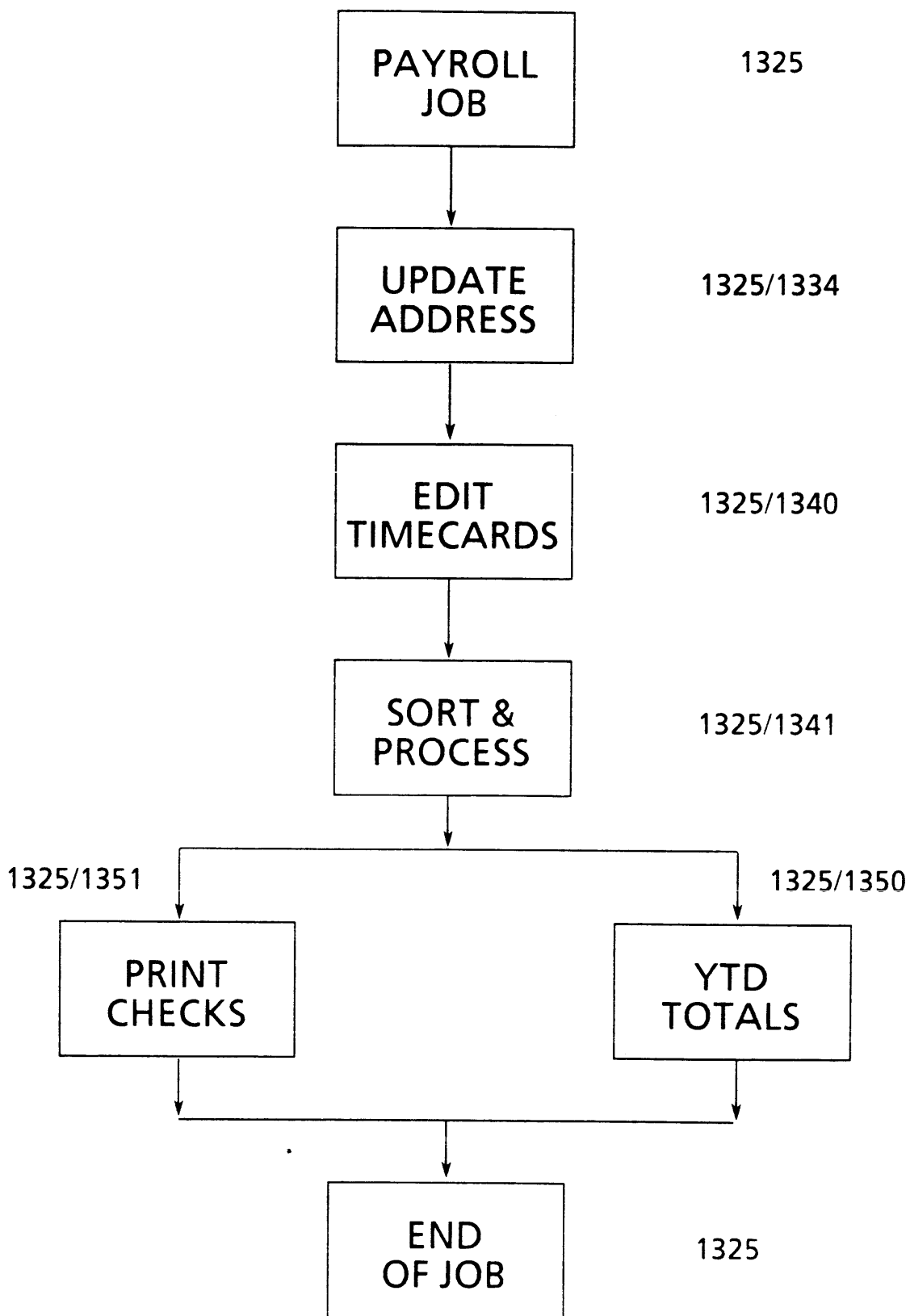
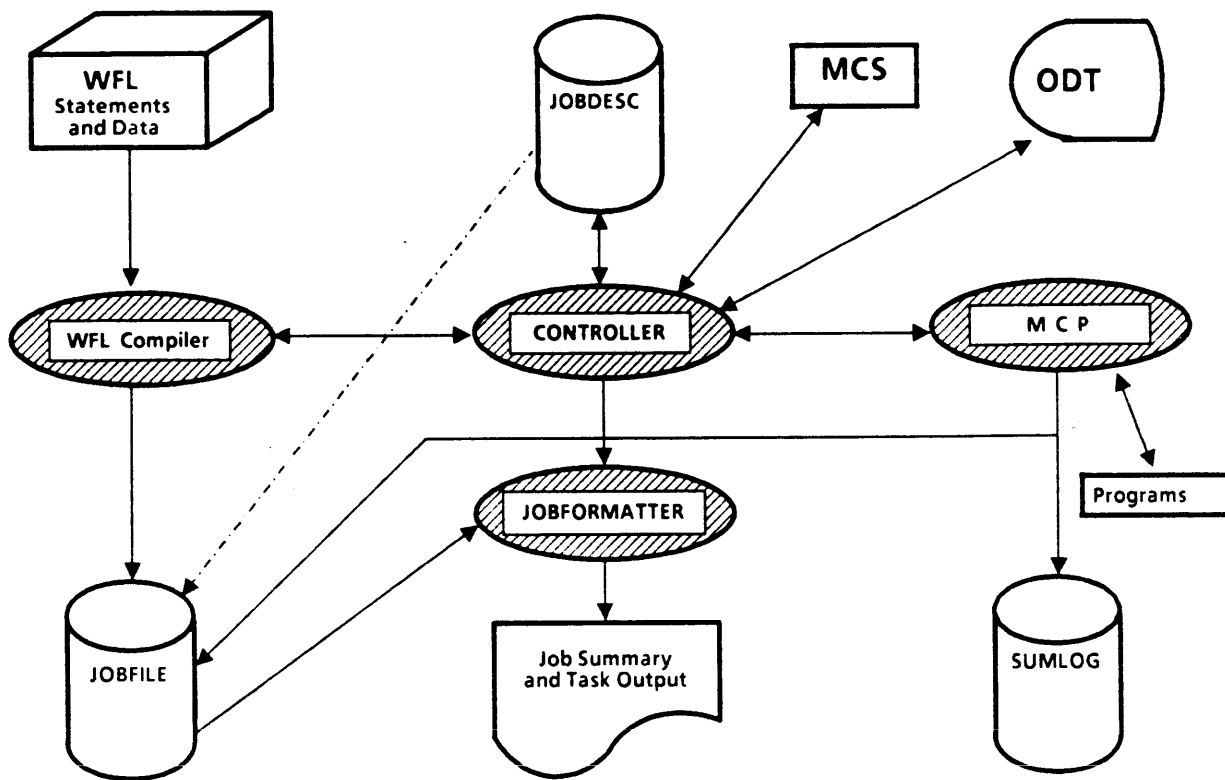


Figure 13-1 Sample Work Flow Job

# A SERIES AND B 5000/B 6000/B 7000 CONCEPTS WORK FLOW MANAGEMENT SYSTEM OVERVIEW

## Flow Of A Job Through The System



Shaded areas represent parts of the MCP.

Figure 13-2 Work Flow Management System Organization

A Work Flow job is executed by the START command.

Syntax:     START <WFL source file name >

Example:    START JOB/PAYROLL



## A SERIES AND B 5000/B 6000/B 7000 CONCEPTS WORK FLOW MANAGEMENT SYSTEM OVERVIEW

The Work Flow Language source statements are passed to the WFL Compiler.

The WFL Compiler:

Checks the syntax of each WFL statement.

Translates source into machine code for the job if syntax errors are not encountered.

Stores the machine code, WFL source, data decks, and space for logging and restart information in a special file type called a **JOBFILE**.

After a successful WFL compile, control is passed from the WFL Compiler to the **Controller** which is an Independent Runner.

The Controller is responsible for:

Controlling the system work load.

Queue level scheduling.

Communicating with the operator

The Controller maintains a Jobfile Description File (**JOBDESC**), which has the dual roles of directory for the jobfile disk areas and queue for the scheduling of the Controller.

The **JOBDESC** consists of:

File headers for the jobfiles.

Links used by the Controller to organize the jobs by class and priority.

The dotted line in Figure 13-2 represents the pointers that the job queues maintain to the various jobfiles to order them by queue and priority.

## **A SERIES AND B 5000/B 6000/B 7000 CONCEPTS WORK FLOW MANAGEMENT SYSTEM OVERVIEW**

The Controller procedure **ABSTRACT** is used to choose the appropriate class or job queue for the jobfile by matching requirements of the job with the specifications of the various queues.

A Job Queue is a list of jobs waiting to be executed and is used for scheduling. Refer to Figure 13-3.

A job queue is created by the MQ (Make or Modify Queue) ODT command. This command allows you to establish the characteristics of the job queue, some of which are Mixlimit, Tasklimit, Defaults and Limits for Priority and other system resources.

A site can define from 1 to 100 queues for a system. You must have at least 1 queue. The DQ (Default Queue) ODT command can be used to designate the default queue for the system.

The SQ (Show Queue) ODT command can be used to display information regarding the jobs in one or all of the job queues.

Figure 13-4 is the Job Enqueueing Algorithm that is used by the **ABSTRACT** procedure.

A SERIES AND B 5000/B 6000/B 7000 CONCEPTS  
WORK FLOW MANAGEMENT SYSTEM OVERVIEW

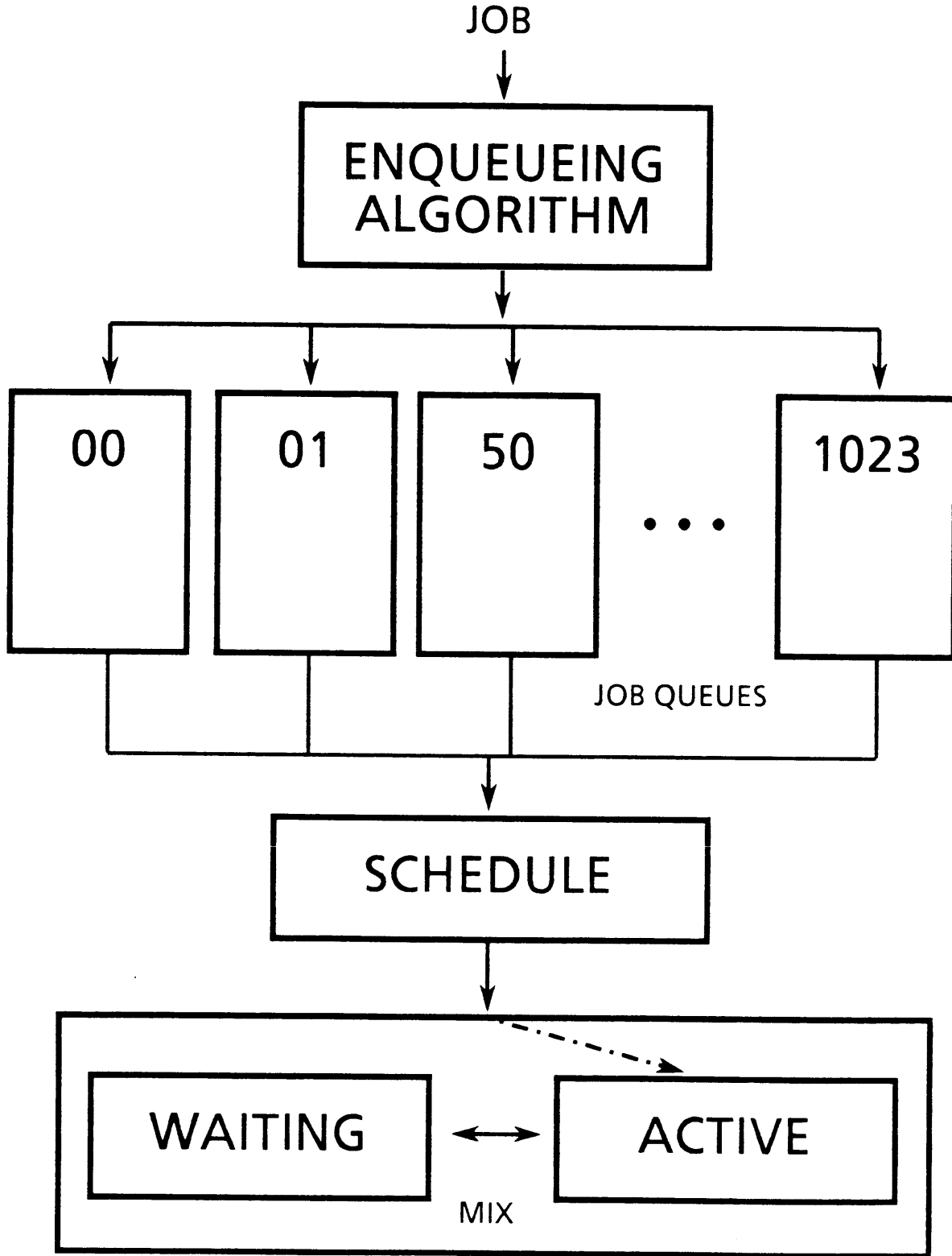


Figure 13-3 General Flow of a Job through the System

**A SERIES AND B 5000/B 6000/B 7000 CONCEPTS  
WORK FLOW MANAGEMENT SYSTEM OVERVIEW**

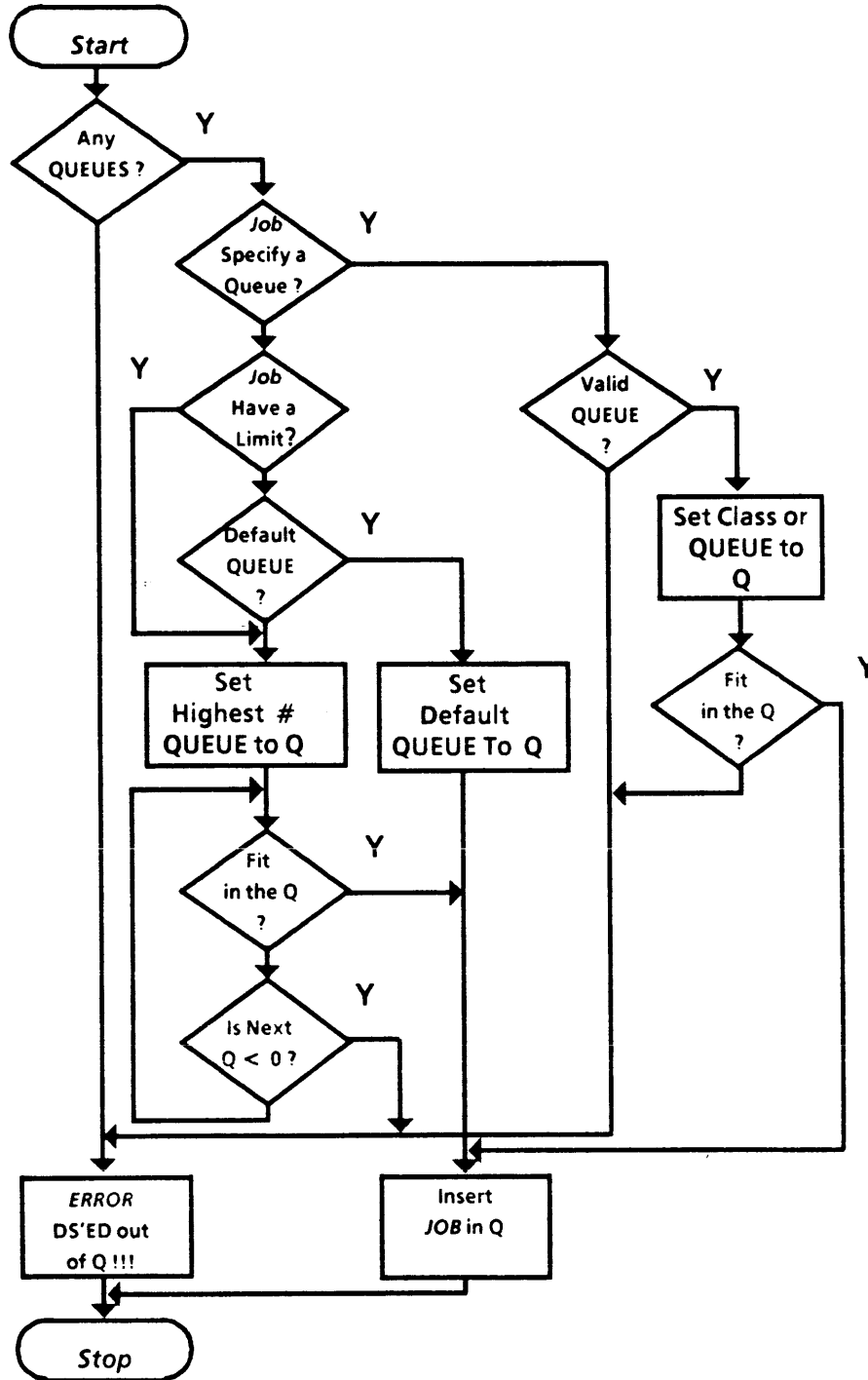


Figure 13-4 Job Enqueueing Algorithm

The **QUEUEINSERT** procedure in Controller is used to insert the job at the appropriate location in the queue through the use of doubly linked lists. The doubly linked lists provided an efficient way to make sure that jobs are in the proper place in the waiting line.

# A SERIES AND B 5000/B 6000/B 7000 CONCEPTS WORK FLOW MANAGEMENT SYSTEM OVERVIEW

The Controller **SELECTION** procedure is then used to choose what job is to be started next.

The job is selected using the Job Selection Algorithm in Figure 13-5 and removed from the job queue.

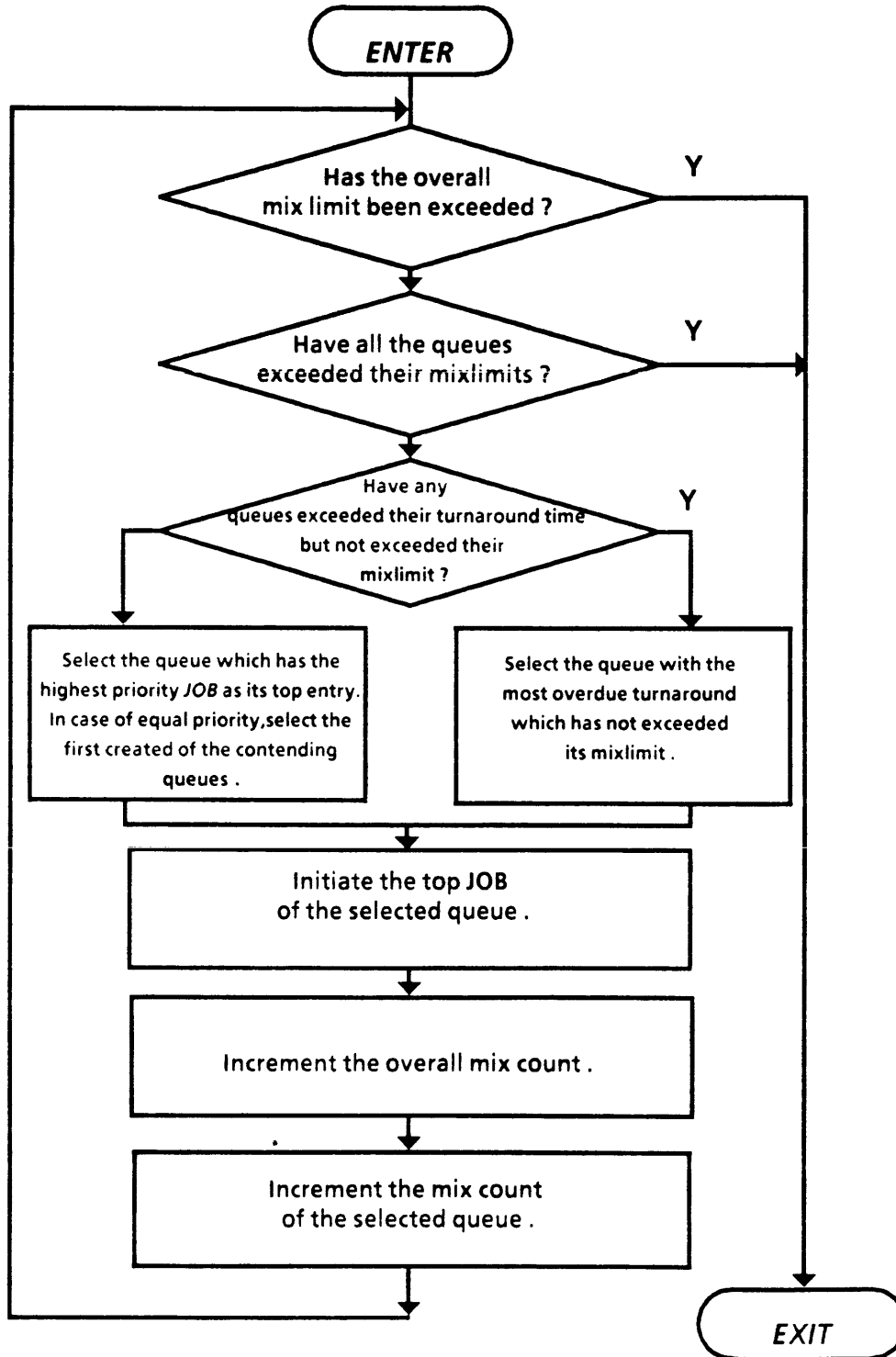


Figure 13-5 Job Selection Algorithm

## **A SERIES AND B 5000/B 6000/B 7000 CONCEPTS WORK FLOW MANAGEMENT SYSTEM OVERVIEW**

Following the Selection procedure, the Controller presents the job to the MCP JOBSTARTER Routine for processing.

The standard MCP scheduling algorithms based on memory estimates and priority discussed in Section 12 - Memory Management Overview take over.

Jobs then become Active or Scheduled depending on availability of system resources, as in Figure 13-3.

Jobstarter builds part of the Process Information Block (PIB) and transfers the job to stack structures as discussed in Section 9 -Stack Architecture Concepts.

When the job is completed, the NORMALEOJ Routine is called and places a message in the Controller's queue to initiate the printing procedure.

The Controller will interface to Autobackup or the Printing Subsystem and the MCP JOBFORMATTER routine to print or punch the output from the job that just finished.

The JOBSUMMARY that is printed contains:

WFL source.

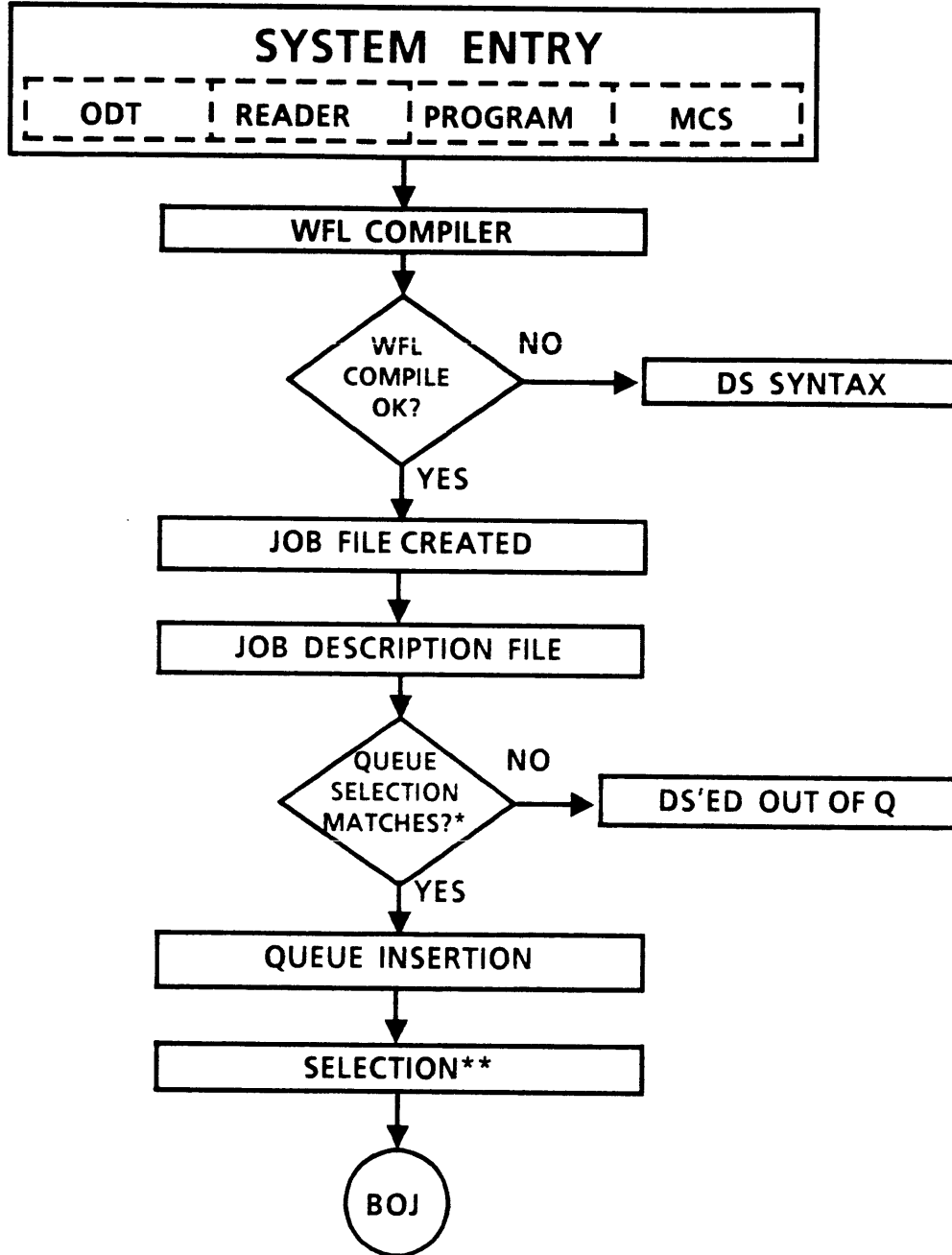
Job history.

Output from tasks, in the order that the tasks were executed.

# A SERIES AND B 5000/B 6000/B 7000 CONCEPTS WORK FLOW MANAGEMENT SYSTEM OVERVIEW

## Flowchart of Job Through the System

Figure 13-6 is a detailed flowchart of a job traveling through the system from the entering of the START command to the EOJ of the job.



\* For queue insertion details see Figure 13-4

\*\* For job selection details see Figure 13-5

Figure 13-6a Detailed Flowchart from START to BOJ

A SERIES AND B 5000/B 6000/B 7000 CONCEPTS  
 WORK FLOW MANAGEMENT SYSTEM OVERVIEW

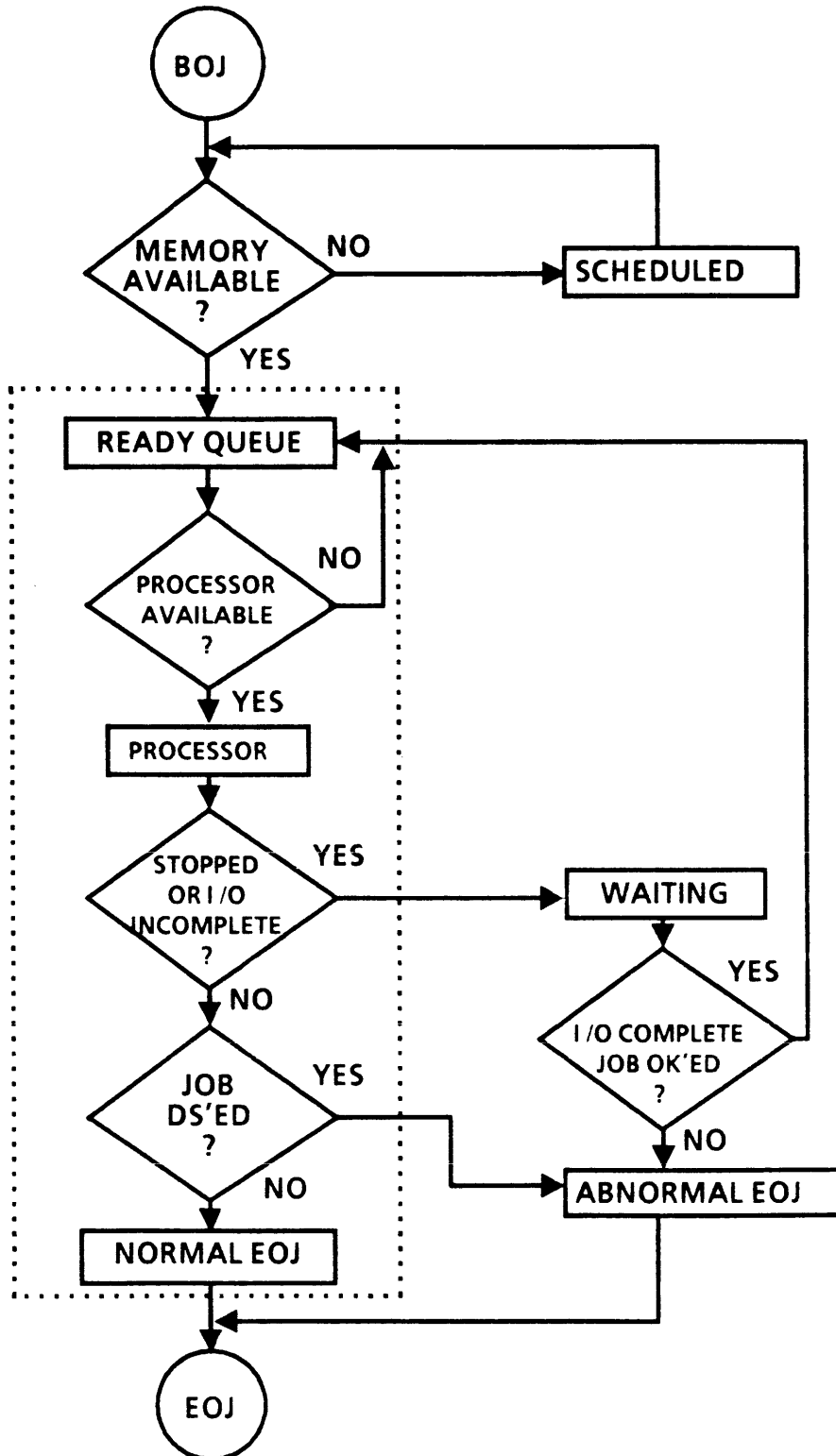


Figure 13-6b Detailed Flowchart from BOJ to EOJ



**A SERIES AND B 5000/B 6000/B 7000 CONCEPTS  
WORK FLOW MANAGEMENT SYSTEM**

**UNIT 2**

***AUTOMATIC DISPLAY MODE***

**Objective**

Use Automatic Display Mode to monitor Work Flow jobs.

**Purpose**

In order to monitor Work Flow jobs, you must be aware of how to read the ODT.

**Resources**

A Series ODT Reference, Section 2 - ODT Commands

# A SERIES AND B 5000/B 6000/B 7000 CONCEPTS AUTOMATIC DISPLAY MODE

## Automatic Display Mode

Automatic Display Mode (ADM) is a means of automatically displaying system status information on the ODTs. The ODTs display can be controlled by the ADM (Automatic Display Mode) ODT command.

ADM provides the tool for monitoring jobs and tasks.

The ADM command allows you to initiate or stop the automatic display at the ODT.

The ADM command allows you to declare what status information is to be displayed and how often the information is to be updated.

Two different ADMs exist.

### Event-driven ADM

Any new event that occurs will automatically update the display in event-driven ADM.

### Time-driven ADM

The display is updated at the specified time interval in time-driven ADM.

The status information is divided into categories. Some of the categories are:

#### Active

Displays the active task and jobs.

#### Waiting

Displays task or jobs that are waiting on an event such as operator input.

#### Scheduled

Displays jobs that are scheduled for execution because memory is currently unavailable. These jobs will become active as soon as system resources can be allocated to the job.

#### Completed

Displays jobs and tasks that have been completed.

#### Messages

Displays the most recent messages from tasks or jobs.

#### Per

Displays the status of the peripherals.

A SERIES AND B 5000/B 6000/B 7000 CONCEPTS  
AUTOMATIC DISPLAY MODE

```
----- 5 ACTIVE ENTRIES-----  
1225 JOB 60 JOBTWO  
*..... 1227 60 *ALGOL ON DISK PROG/TEST  
552 JOB 50 *SYSTEM/PRINT/ROUTER  
551 JOB 80 NSP 108/00  
  
----- 3 WAITING ENTRIES-----  
* 1220/1226 50 PAYROLL/WRITER  
OPERATOR STOPPED  
* 1100/1228 40 (USER1)TEST/EMP/HISTORY  
NO FILE EMP/MASTER ON EMPACK ELSE DISK (PK)  
1197/1198 50 CLASS/UPDATE  
ACCEPT: ENTER OPENING DATE MMDDYY  
  
----- 1 SCHEDULED ENTRIES-----  
*1230 JOB 65 (STUDENT1)OBJECT/LAB/1  
  
----- COMPLETED ENTRIES-----  
*1095/1100 EOT COBOL CLASS/TESTER  
1050/1210 SNTX ALGOL ACCTS/REC/UPDATE  
  
----- MESSAGES -----  
* 1235 (USER1)DATA REMOVED FROM DISK  
* 1200 LP4: (PAY)OUT/FILE ON PAYPK PRINTED  
  
----- MT STATUS -----  
14 P [099999] 1600 #1 1:0 BKTAPE/FILE000
```

Figure 13-7 Sample ODT Screen ADM (A5, W7, S2, C3, MSG3, PER MT2)

# A SERIES AND B 5000/B 6000/B 7000 CONCEPTS AUTOMATIC DISPLAY MODE

## Establishing ADM Options

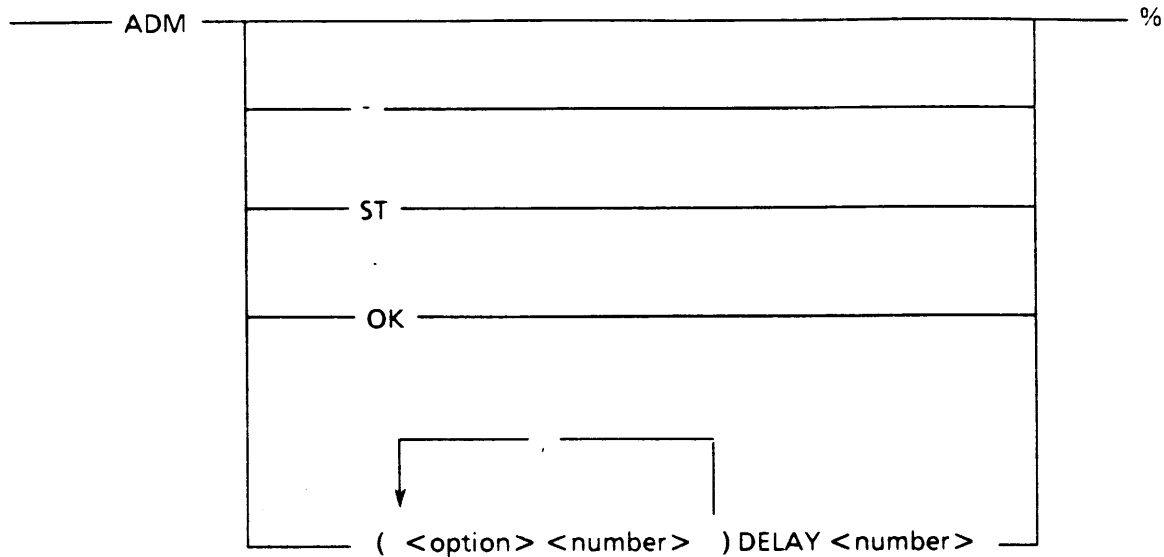


Figure 13-8 Simple ADM Syntax

### ADM

Displays the current ADM options. If there are no current ADM options, invokes the default ADM.

### ADM -

Cancels ADM operations.

### ADM ST

Stops the ADM but retains the current ADM settings.

### ADM OK

Resumes ADM operations following an ADM ST.

### <option>

Category of information to be displayed.

### <number>

Number of lines to be used for a category display. This number must include 1 line for the category heading.

### DELAY <number>

Number of seconds to wait before refreshing the ODT.

## A SERIES AND B 5000/B 6000/B 7000 CONCEPTS AUTOMATIC DISPLAY MODE

Examples:

ADM (A6, W4, C) DELAY 6

The ODT would display 6 lines of active entries (1 header, 5 active programs), 4 lines of waiting entries, and 12 lines of completed entries. This information will be refreshed every 6 seconds.

ADM (MSG 12, PER MT 3, S 7) DELAY 10

The ODT would display 12 lines of messages from programs, 3 lines of status information about the PE tape drives and 7 lines of scheduled entries. The screen will be refreshed every 10 seconds.

ADM (A 10, W 8, C 4, S 5, MSG) DELAY 5

This format requires 2 screens to display all of the requested information. The first screen will display 10 lines of active entries, 8 lines of waiting entries, and 4 lines of completed entries. After 5 seconds the second screen containing 5 lines of scheduled entries and 17 lines of messages will displayed. After 5 seconds, the display will be the first format of active, waiting and completed entries. ADM will alternate between the 2 pages of information. The same ADM settings can be obtained by entering ADM (A10, W8, C4) DELAY 5 (S5, MSG) DELAY 5.

### *ADM Options as Inquiries*

Some of the ADM options can be entered as ODT commands.

A	(Active Mix Entries)	- Displays active entries.
W	(Waiting Mix Entries)	- Displays the waiting entries.
S	(Scheduled Mix Entries)	- Displays the scheduled entries.
C	(Completed Mix Entries)	- Displays the completed entries.
MSG	(Display Messages)	- Displays the messages from tasks and jobs.

**A SERIES AND B 5000/B 6000/B 7000 CONCEPTS  
AUTOMATIC DISPLAY MODE**

**Practice**

Part A: Name the parts of the Work Flow Management System that perform the following functions.

1. This program checks the syntax of a WFL source file, and translates the WFL statements into machine code.  
\_\_\_\_\_
2. This file contains object code, WFL source, data decks, log entries, and restart information for the job.  
\_\_\_\_\_
3. This independent runner coordinates other parts of the WFL system, and handles all commands entered at the ODT.  
\_\_\_\_\_
4. This procedure places jobs in queues by matching the requirements of the jobs with the characteristics of the job queues.  
\_\_\_\_\_
5. The Controller organizes the jobs by class and priority within this file.  
\_\_\_\_\_
6. After a job ends, this procedure formats the printed output and enqueues the job for printing.  
\_\_\_\_\_

Part B: Under what ADM heading would the following information be displayed? (Or, what ODT command could you enter to display this information?)

1. An initiation message displayed by a program \_\_\_\_\_
2. Notification that a task has ended \_\_\_\_\_
3. The prompt for an Accept needed by a program \_\_\_\_\_
4. The name of a tape mounted on a tape drive \_\_\_\_\_
5. The job and task number for an executing task \_\_\_\_\_
6. The name of a missing file needed by a program \_\_\_\_\_
7. Notification that a file has been removed \_\_\_\_\_
8. The names of all the scheduled tasks \_\_\_\_\_

**A SERIES AND B 5000/B 6000/B 7000 CONCEPTS  
WORK FLOW MANAGEMENT SYSTEM**

***UNIT 3***

***BASIC WFL SYNTAX AND STATEMENTS***

**Objective**

Identify the basic syntax of the Work Flow Language.

Identify the Work Flow Language structure.

**Purpose**

In order to control the system, you must be able to write Work Flow jobs.

**Resources**

A Series Work Flow Language Reference, Section 3 - Job Structure

# A SERIES AND B 5000/B 6000/B 7000 CONCEPTS BASIC WFL SYNTAX AND COMMANDS

## WFL

The Work Flow Language is high-level language in which jobs are written. Work Flow jobs are used to control the system.

### *Syntax Rules*

The Work Flow Language is a free-format language except for some restrictions.

WFL source files have a file type of JOB.

The source statements must be located in columns 1 through 80 and the sequence numbers are located in columns 83 through 90.

Each WFL statement must end with a semi-colon ( ; ).

Documentation and comment information can be inserted into the WFL source by using a percent sign ( % ) at the start of the comment.

A WFL job is composed of statements that are grouped into sections.

The sections must appear in a specified order but the statements within each section can appear in any order.

Section headers do not appear in the source file. The compiler distinguishes each section by the syntax.

WFL has some reserved words. For the list of reserved words, refer to the Work Flow Language Reference Manual.



# A SERIES AND B 5000/B 6000/B 7000 CONCEPTS BASIC WFL SYNTAX AND COMMANDS

## *WFL Job Structure*

A WFL job is divided into 5 sections.

### *Begin Job Section*

This section identifies the beginning of a WFL job, declares the job name and the job disposition. The begin job statement is also known as the job header. This section is required.

Syntax:     BEGIN JOB <job title> <disposition>;

Example:    BEGIN JOB ACCTS/PRINTS FOR SYNTAX;

### *Job Attributes Section*

The characteristics of the job are declared in this section. These characteristics are used to control the job environment and behavior. This section is optional.

Syntax:     <attribute> = <value>;

Example:    FAMILY DISK = ACCTSPK ONLY;

### *Job Declarations Section*

This section is the area where variables are defined and the variable's intended use is expressed. Any variables to be used in the statement section must be declared here. This is an optional section.

Syntax:     <variable type> <variable name>;

Example:    INTEGER I1;

## A SERIES AND B 5000/B 6000/B 7000 CONCEPTS BASIC WFL SYNTAX AND COMMANDS

### *Statements Section*

This section is the working section of the job. The statements in this section initiate a process or task. This is an optional section, but a job will not perform any function without any WFL statements.

Syntax:     RUN <file title>;

Example:    RUN PROG/SAVINGS/REPORT;

### *End Job Section*

This section designates the end of the WFL job source. This section is required.

Syntax:     END JOB;

**A SERIES AND B 5000/B 6000/B 7000 CONCEPTS  
BASIC WFL SYNTAX AND COMMANDS**

*Sample WFL Job*

% Begin Job Section	100
BEGIN JOB SAMPLE/MYJOB;	200
% Job Attributes Section	300
USERCODE = CONCEPTS/STUDENT;	400
FAMILY DISK = EDUCATION OTHERWISE DISK;	500
PRIORITY = 75;	600
QUEUE = 20;	700
% Job Declarations Section	800
INTEGER I1, I2;	900
INTEGER COUNTER;	1000
REAL R1, R2, R3;	1100
% Statements Section	1200
RUN MYPROG ON EDUCATION;	1300
COUNTER := COUNTER + 1;        %Assignment Statement	1400
EXECUTE PRINT/CHECK/PROGRAM;	1500
%End Job Section	1600
END JOB.	1700

Figure 13-9 Sample Work Flow Job

**A SERIES AND B 5000/B 6000/B 7000 CONCEPTS  
WORK FLOW MANAGEMENT SYSTEM**

**UNIT 4**

**WORK FLOW LANGUAGE FILE EQUATIONS AND COMPILING**

**Objective**

Use Work Flow Language file equations.

Compile programs using the Work Flow Language.

**Purpose**

To provide file use flexibility in executing and compiling programs, you need to use file equations in your Work Flow job.

**Resources**

A Series Work Flow Language Reference, Section 5 - Task Initiation  
Section 6 - Statements

# A SERIES AND B 5000/B 6000/B 7000 CONCEPTS WFL FILE EQUATIONS AND COMPILING

## File Equation

A File Equation provides an easy way of changing various file attributes for this run of a task. The file equation overrides the file attributes that were declared in the source program.

File equations provide flexibility.

A task is allowed to read from or write to different files from those it normally would use.

A task is allowed to execute using different file attributes.

Some common file attributes that may be changed for different executions of a task are:

### Kind

This attribute specifies the hardware type for the file.

### Title

This attribute specifies the external name of the file which includes the file name and family name. For certain executions of a task, you may wish to use data located in a file with a different name than was specified in the program's source.

### Familyname

This attribute specifies the family location of the file.

### Filename

This attribute specifies the file name.

# A SERIES AND B 5000/B 6000/B 7000 CONCEPTS WFL FILE EQUATIONS AND COMPILING

## Syntax and Placement

File equations require that you specify the internal file name followed by the new values for the file attributes.

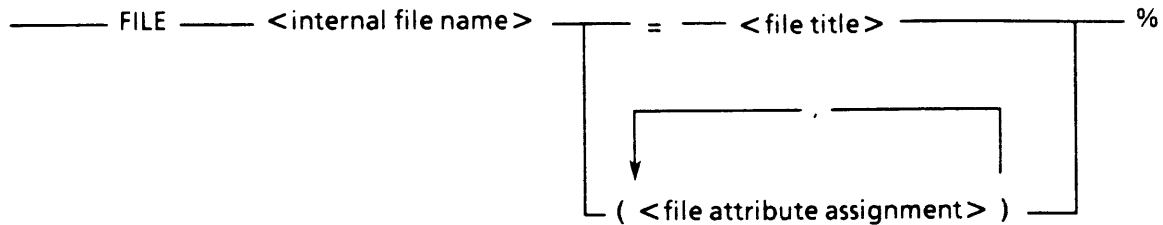


Figure 13-10 Simple File Equation Syntax

### Examples:

```
FILE EMPFILE (KIND = DISK, TITLE = TEST/EMP/NAMES);
```

```
FILE INPUTDATA (KIND = TAPE);
```

```
FILE OUTDATA (KIND= PRINTER);
```

The file equation is placed after the Run statement in the Statement Section.

Syntax:     RUN MYPROG;  
              <file equation>;

### Examples:

```
RUN MYPROG;  
  FILE DATAIN (TITLE = TEST/DATA/218);
```

```
RUN MYPROG;  
  FILE DATAIN = TEST/DATA/218;
```

```
RUN PRINT/REPORT;  
  FILE OUTREPT (KIND = TAPE);
```

**A SERIES AND B 5000/B 6000/B 7000 CONCEPTS  
WFL FILE EQUATIONS AND COMPILING**

*Sample WFL Job*

BEGIN JOB SAMPLE/MYJOB;	100
USERCODE = CONCEPTS/STUDENT;	300
FAMILY DISK = EDUCATION OTHERWISE DISK;	400
PRIORITY = 75;	500
QUEUE = 20;	600
INTEGER I1, I2;	700
INTEGER COUNTER;	800
REAL R1, R2, R3;	900
RUN MYPROG;	1100
% Task Execution Using File Equations	1150
RUN ACCT/REC/DEV/PROG;	1200
FILE BILLREC (TITLE = BILLS/DEV/TEST);	1250
RUN ACCT/REC /DEV/PROG;	1300
FILE BILLREC (KIND = DISK, TITLE = TEST/DATA/PART/2);	1350
FILE REPT (KIND = DISK);	1370
.	
ENDJOB;	1700

Figure 13-11 Sample Work Flow Job Using File Equations

# A SERIES AND B 5000/B 6000/B 7000 CONCEPTS WFL FILE EQUATIONS AND COMPILING

## Compiling In WFL

Compiles should be started using WFL:

If you want to compile programs at the ODT.

If you want to apply a patch file to a source file.

If the installation wants to control compiles.

The system can be set up so that all compiles must go through a specified job queue that will limit the amount of system resources that compiles can use.

If you want to do multiple compiles without monitoring the process.

## Compile Statement

Syntax:

```
COMPILE <object code file name> <compiler> <disposition>;
```

<disposition>

SYNTAX	- Check for syntax errors only.
LIBRARY	- Generate and retain object code file.
LIBRARY GO	- Generate, retain and execute object code file.
blank or GO	- Generate and execute object code file. This is the default if no disposition is specified.

Examples:

```
COMPILE MYPROG/TEST COBOL74 LIBRARY;
```

```
COMPILE BILLING/PROG/21 ALGOL SYNTAX;
```



## A SERIES AND B 5000/B 6000/B 7000 CONCEPTS WFL FILE EQUATIONS AND COMPILING

### *File Equations for Compile Statement*

The primary input file for the different compilers is **CARD** and is assigned to a card reader device. Unless the program source will be entered by way of a card reader, a file equation is necessary to change the file attributes **Kind** and **Title** for this file **CARD**.

Examples:

```
COMPILE MYPROG/TEST COBOL74 LIBRARY;  
COMPILER FILE CARD (KIND = DISK, TITLE = SOURCE/MYPROG/TEST);
```

```
COMPILE REPORTS/CLASS ALGOL SYNTAX;  
COMPILER FILE CARD (TITLE = SOURCE/REPORTS/1A);
```

```
COMPILE PAYROLL/WRITER COBOL LIBRARY GO;  
COMPILER FILE CARD (KIND = DISK);
```

### *Compiler Dollar Sign Options*

The output of the compile can be controlled by specifying dollar sign options in the source file.

Examples:

```
$SET LIST
```

This option causes a listing of the source file to be produced from the compile.

```
$SET STACK CODE
```

The stack option causes the stack information such as stack addresses to be printed. The code option causes the machine code information to be printed.

CANDE and WFL will set different dollar sign options automatically when a compile is requested.

**A SERIES AND B 5000/B 6000/B 7000 CONCEPTS  
WFL FILE EQUATIONS AND COMPILING**

**Practice**

Identify the lines of the sample WFL job below by their sequence numbers.

BEGIN JOB SAMPLE;	100
FAMILY DISK = PRODUCTION OTHERWISE DISK;	200
PRIORITY = 75;	300
INTEGER MONTH;	400
MONTH := 9;	500
RUN WEEKLY/REPORTS;	600
FILE WEEK (TITLE = THIS/WEEK/DATA ON ACCT);	700
RUN WEEKLY/TOTALS;	800
COMPILE MONTHLY/REPORTS COBOL LIBRARY;	900
COMPILER FILE CARD (KIND = DISK, TITLE =	1000
SOURCE/MONTHLY/REPORTS);	1100
END JOB	1200

1. Which lines form the following sections of the job?
  - a. Job Attributes \_\_\_\_\_
  - b. Job Header \_\_\_\_\_
  - c. Working Section \_\_\_\_\_
  - d. End Job Section \_\_\_\_\_
  - e. Job Declarations \_\_\_\_\_
  
2. Which lines illustrate the following types of WFL statements?
  - a. File Equation \_\_\_\_\_
  - b. Assignment \_\_\_\_\_
  - c. Compiler Initiation \_\_\_\_\_
  
3. Which lines cause tasks to be executed? \_\_\_\_\_

**A SERIES AND B 5000/B 6000/B 7000 CONCEPTS  
WORK FLOW MANAGEMENT SYSTEM**

**UNIT 5**

**LIBRARY/MAINTENANCE**

**Objective**

Write LIBRARY/MAINTENANCE statements.

**Purpose**

In order to maintain files in your library and on the system, you must know how LIBRARY/MAINTENANCE is used.

**Resources**

A Series Work Flow Language Reference, Section 6 - Statements

## **A SERIES AND B 5000/B 6000/B 7000 CONCEPTS LIBRARY/MAINTENANCE**

### **File Maintenance**

LIBRARY/MAINTENANCE is a program that is used to:

Copy files to/from disk, pack, or tape.

Remove files from disk or pack.

Change file names on disk or pack.

Change security restrictions.

LIBRARY/MAINTENANCE is a WFL function even though the LIBRARY/MAINTENANCE program can be initiated through WFL, CANDE, MARC, or ODT commands.

The statements Copy, Add, Change, Remove and Security are part of the LIBRARY/MAINTENANCE program.

## A SERIES AND B 5000/B 6000/B 7000 CONCEPTS LIBRARY/MAINTENANCE

### **COPY**

The copy statement copies files from one location to another. LIBRARY/MAINTENANCE copy creates a library tape, usually for backup purposes, or reads a library tape.

Syntax:

```
COPY <file name> FROM <volume name> (<volume kind>)  
TO <volume name> (<volume kind>);
```

<volume name> is the name of a disk, pack or tape.

<volume kind> is the peripheral type such as disk or tape.

Examples:

```
COPY A/B FROM PRODUCTION (KIND = PACK) TO X (KIND = PACK);
```

```
COPY A/B FROM PRODUCTION (PACK) TO X (TAPE);
```

```
COPY A/B FROM DISK TO X(PACK);
```

```
COPY A/B/= FROM X;
```

```
COPY = FROM X TO TEST (PACK);
```

```
COPY A/B TO T1, TO T2;
```

```
COPY A/B, A/C FROM PACK;
```

```
COPY (USER1)A/= FROM EDUCATION (PACK) TO BACKUP (TAPE);
```

```
COPY A/= AS C/= TO EDUCATION (PACK);
```

**A SERIES AND B 5000/B 6000/B 7000 CONCEPTS  
LIBRARY/MAINTENANCE**

**LIBRARY/MAINTENANCE DEFAULTS**

Statement	If omit FROM specification	If omit TO specification	If specify FROM without KIND	If specify TO without KIND
<b>COPY</b>	DISK, or Substitute	DISK, or Substitute	TAPE*	TAPE*
<b>ADD</b>	DISK, or Substitute	DISK, or Substitute	TAPE*	CAN'T ADD TO TAPE
<b>CHANGE</b>	DISK, or Substitute	N/A	Specified Family **	N/A
<b>REMOVE</b>	DISK, or Substitute	N/A	Specified Family **	N/A
<b>SECURITY</b>	DISK, or Substitute	N/A	Specified Family **	N/A

\* Unless specify DISK or PACK which are reserved names

\*\* Cannot do change, remove or security on tape

Figure 13-12 LIBRARY/MAINTENANCE Defaults

## A SERIES AND B 5000/B 6000/B 7000 CONCEPTS LIBRARY/MAINTENANCE

### **ADD**

The add statement copies files from one location to another unless the files already exist on the destination location. Files cannot be added to a tape.

#### Syntax:

```
ADD <file name> FROM <volume name> (<volume kind>)  
TO <volume name> (<volume kind>);
```

<volume name> is the name of a disk, pack or tape.

<volume kind> is the peripheral type such as disk or tape.

#### Examples:

```
ADD A/B FROM PRODUCTION (KIND = PACK) TO X (PACK);
```

```
ADD A/B FROM PRODUCTION (TAPE) TO X (PACK);
```

```
ADD A/B FROM DISK TO X(PACK);
```

```
ADD A/= FROM EDBK (TAPE) TO EDUCATION (PACK);
```

```
ADD A/= AS C/= TO EDUCATION (PACK);
```

## A SERIES AND B 5000/B 6000/B 7000 CONCEPTS LIBRARY/MAINTENANCE

### **CHANGE**

The change statement changes names of files on disk or packs. Files names cannot be changed on a tape. The change statement causes the directory to be updated to reflect the new file name.

#### Syntax:

```
CHANGE <file name> TO <file name>  
FROM <volume name> (<volume kind>);
```

<volume name> is the name of a disk, or pack.

<volume kind> is the peripheral type such as disk.

#### Examples:

```
CHANGE A/B TO C/D FROM PRODUCTION (KIND = PACK);
```

```
CHANGE FILE/1 TO FILE/PAYROLL, FILE/2 TO FILE/TIME/CARDS FROM DISK;
```

```
CHANGE A/= TO B/= FROM EDUCATION;
```



## A SERIES AND B 5000/B 6000/B 7000 CONCEPTS LIBRARY/MAINTENANCE

### *REMOVE*

The remove statement erases files from disk or pack but not from a tape.

#### Syntax:

```
REMOVE <file name> FROM <volume name> (<volume kind>);
```

<volume name> is the name of a disk or pack.

<volume kind> is the peripheral type such as disk.

#### Examples:

```
REMOVE A/B FROM PRODUCTION (KIND = PACK);
```

```
REMOVE FILE/1, FILE/PAYROLL, FILE/2 FROM DISK;
```

```
REMOVE A/=, B/= FROM EDUCATION;
```



**SECTION 14**  
**SECURITY OVERVIEW**

# A SERIES AND B 5000/B 6000/B 7000 CONCEPTS SECURITY OVERVIEW

## *INTRODUCTION*

### **Section Objective**

Identify the factors that control access to the system and the disk files.

### **Purpose**

Several different factors control who can access the system through terminals and which disk files they can use. Programmers and managers should be aware of the most common security controls.

### **Unit Objectives**

Recognize the role of usercodes in security.

Identify the file attributes that control access to disk files.

Recognize the purpose of guardfiles.

Change the security attributes of a disk file through the SECURITY statement.

# **A SERIES AND B 5000/B 6000/B 7000 CONCEPTS SECURITY OVERVIEW**

## **UNIT 1**

### **SECURITY OVERVIEW**

#### **Objectives**

Recognize the role of usercodes in security.

Identify the file attributes that control access to disk files.

Recognize the purpose of guardfiles.

Change the security attributes of a disk file through the SECURITY statement.

#### **Purpose**

Several different factors control who can access the system through terminals and which disk files they can use. Programmers and managers should be aware of the most common security controls.

#### **Resources**

A Series I/O Subsystem Reference Manual, Section 9 - Disk File and System-Access Security

A Series System Software Utilities Reference Manual, Section 7 - GUARDFILE

A Series System Software Site Management Reference Manual, Section 10- MAKEUSER

# A SERIES AND B 5000/B 6000/B 7000 CONCEPTS SECURITY OVERVIEW

## Security Through Usercodes

Usercodes are involved in two types of security:

**Access to the system through a terminal** is controlled by an MCS, which usually requires that a valid usercode be entered. The MCS may restrict each usercode to certain terminals, certain programs, or certain commands.

**Access to disk files** is controlled by usercodes and file attributes, as described below.

## SYSTEM/USERDATAFILE

SYSTEM/USERDATAFILE contains all the usercodes, and their associated characteristics, that have been defined for the system. Examples of usercode characteristics are:

Password(s)

Family substitution statement

Chargecode

Job queues that the user can or cannot use

Nonprivileged or privileged status

SYSTEM/USERDATAFILE is maintained through the utility SYSTEM/MAKEUSER, which can be executed by privileged users only.

## *Types of Usercodes*

The status of each usercode is stored in SYSTEM/USERDATAFILE.

**Nonprivileged** users may access their own files and public files.

**Privileged** users may access any file, may invoke certain MCP procedures, and may execute SYSTEM/MAKEUSER.

**Systemusers** are given a system-wide view through MARC at a terminal, as if they were at the ODT. The MARC home menu that is displayed when a systemuser logs on includes additional choices. A systemuser is also either privileged or nonprivileged.

An object code file can be designated as a **privileged program** through the PP ODT command. A privileged program always executes as if it were initiated under a privileged usercode, and can access any file.

## A SERIES AND B 5000/B 6000/B 7000 CONCEPTS SECURITY OVERVIEW

### Security Through Disk File Attributes

The three file attributes below control access to disk files.

**SECURITYTYPE** is a mnemonic-valued attribute that indicates which users may access a file.

**PRIVATE** indicates that only the owner or a privileged user may access the file.

**PUBLIC** allows access by any user.

**GUARDED** allows access by a user specified in a guardfile (identified by the **SECURITYGUARD** attribute), the owner, or a privileged user.

**CONTROLLED** allows access by a user specified in a guardfile, or a privileged user. If the owning usercode is nonprivileged, even that usercode is subject to the guardfile named by **SECURITYGUARD**.

**SECURITYUSE** is a mnemonic-valued attribute that specifies the manner in which **PRIVATE** and **PUBLIC** files may be accessed.

**IN** specifies that the file may be used only for input (read-only).

**OUT** specifies that the file may be used only for output (write-only).

**IO** allows input and output (read and write) access to the file.

**SECURED** allows object code files to be executed, but not accessed in other ways. If other types of files (for example, data files) are secured, no access is allowed.

**SECURITYGUARD** attribute specifies the title of the guardfile for guarded and controlled files.

# A SERIES AND B 5000/B 6000/B 7000 CONCEPTS SECURITY OVERVIEW

## Security Through Guardfiles

Guardfiles provide flexibility in defining security for individual files or databases by specifying the access rights allowed:

Usercodes that may or may not access the file or database.

Programs that may or may not access the file or database.

Actions allowed against the file or database (for example, read only or read/write).

DMS II verbs that may be used against the database (for example, FIND and DELETE).

Combinations of usercodes, programs, actions, and DMS II verbs that may be used.

Guardfiles are created and maintained through the utility SYSTEM/GUARDFILE.

## LIBRARY/MAINTENANCE SECURITY Statement

The security attributes of a disk file can be changed through a SECURITY statement entered at the ODT, in a WFL job, or in a CANDE or MARC session.

Examples:

```
SECURITY (USER1)FILE/ONE PUBLIC IN
```

```
SECURITY FILE/TWO PRIVATE IO
```

The SECURITY statement can be used to associate a guardfile with a file. This affects both the SECURITYTYPE and the SECURITYGUARD attributes.

Examples:

```
SECURITY FILE/THREE GUARDED WATCH/DOG
```

```
SECURITY FILE/FOUR CONTROLLED BODY/GUARD
```



**SECTION 15**  
**SOFTWARE PRODUCTS OVERVIEW**

# A SERIES AND B 5000/B 6000/B 7000 CONCEPTS SOFTWARE PRODUCTS OVERVIEW

## ***INTRODUCTION***

### **Section Objective**

Identify the major software products available for the A Series systems.

### **Purpose**

You should be familiar with the functions and capabilities of the software products, for decision making and planning.

### **Unit Objectives**

Identify the major components of the printing subsystem.

Identify the major A Series utilities.

Identify the major software components of DMS II.

Identify other major software products available for the A Series systems.

# A SERIES AND B 5000/B 6000/B 7000 CONCEPTS SOFTWARE PRODUCTS OVERVIEW

## *UNIT 1*

### *PRINTING SUBSYSTEM OVERVIEW*

#### **Objective**

Identify the major components of the printing subsystem.

#### **Purpose**

You should be aware of the features and options of the printing software.

#### **Resources**

A Series Print System (PrintS/ReprintS) User's Guide, Entire Manual

A Series Printing Utilities User's Guide, Section 2 - Backup Processor Utility  
Section 3 - SYSTEM/BACKUP Utility

## A SERIES AND B 5000/B 6000/B 7000 CONCEPTS PRINTING SUBSYSTEM OVERVIEW

### Printer Backup Files

The purpose of printer backup is to **spool** printer files to disk as the program creates them, and print them later when a printer is available. This allows many print programs to execute at once, although there may be only 1 printer on the system, and allows programs to run faster because they are not tied to the speed of a printer.

Most A Series installations force all printer files to backup disk by setting the LPBDONLY option with the OP (options) ODT command.

The family name where backup files are placed is specified by a combination of the SB (Substitute Backup) and DL (Disk Location) ODT commands.

The naming convention for printer backup files is:

BD/000 <job number > /000 <task number > / <modified file name > .

Job and task number refer to the program that created the output.

Modified file name includes three digits, 000 to 999, prefixed to the internal file name, to prevent duplicate file names if the printer file is repeatedly opened and closed during the program.

Printer backup files are often called **BD files**, because they are under the BD directory.

By default, printer backup files under the BD directory on the DL backup family will be printed on an available printer and removed from the disk, when the job that created them ends. The programmer and/or operator can specify several file and task attributes to override the defaults.

## A SERIES AND B 5000/B 6000/B 7000 CONCEPTS PRINTING SUBSYSTEM OVERVIEW

### *File Attributes for Printer Files*

Several file attributes provide control over the output of printer files. Some examples of printer file attributes are:

#### PRINTCOPIES

Specifies the number of copies to be printed. The default value is 1.

#### SAVEBACKUPFILE

Indicates whether the backup file should be saved on disk after printing. The possible values are TRUE and FALSE. The default is FALSE.

#### DESTINATION

Specifies the name of the printer to be used for output. By default, the system uses an available printer from the default printer pool.

These file attributes can be set through application program statements or WFL file equation.

Example of WFL file equation:

```
BEGIN JOB MYJOB;
```

```
RUN MY/PROGRAM;
```

```
FILE PRTR (PRINTCOPIES = 2, SAVEBACKUPFILE = TRUE,  
          DESTINATION = "LP4");
```

```
END JOB
```

## A SERIES AND B 5000/B 6000/B 7000 CONCEPTS PRINTING SUBSYSTEM OVERVIEW

### *Job and Task Attributes for Printing*

Some job and task attributes also apply to the printing process. Two examples are:

#### **BDNAME** job or task attribute

Specifies the prefix to be used for backup file names, instead of BD. The rest of the file naming convention still applies. This protects backup files created by the job or task from being printed and removed automatically. They can be printed on request later (see the PRINT statement below).

#### **JOBSUMMARY** job attribute

Indicates whether or not a job summary should be printed. JOBSUMMARY has several values, such as SUPPRESSED and UNCONDITIONAL.

These job and task attributes can be specified in WFL.

Example:

```
BEGIN JOB ALPHABET;  
JOBSUMMARY = UNCONDITIONAL;  
BDNAME = ABC;  
RUN XYZ/PROG/1;  
RUN XYZ/PROG/2;  
ENDJOB
```

This job will always print a job summary.

Any printer backup files created by the job will be named ABC/000<job number>/000<task number>/<modified file name>. The backup files will not be printed automatically.

# A SERIES AND B 5000/B 6000/B 7000 CONCEPTS PRINTING SUBSYSTEM OVERVIEW

## Print System (PrintS)

The Print System includes several system software programs, as well as parts of the MCP, that deal with spooling and printing backup files. The programs included in PrintS are:

### SYSTEM/PRINT/ROUTER

This program remains in the mix, to route print files when they are ready to be printed.

### SYSTEM/PRINT/SUPPORT

This library freezes in the mix and provides entry points for the other printing programs.

### SYSTEM/PRINT/BACKUP/PROCESSOR

This is the program initiated when a BACK command is entered in CANDE to view printer backup files at the terminal.

Two files are maintained by PrintS:

### SYSTEM/BACKUPFILELIST

Stores the current print requests. A print request is created when an operator, a program, or the system issues a request to print a backup file.

### SYSTEM/PRINTERINFO

Contains the characteristics of the printing devices attached to the system.

The PS (Printing System) ODT and MARC commands allow the operator to inquire into the status of print requests and to modify print requests if necessary. The operations manager can use the PS command to configure printing the devices on the system.

Example:

PS SHOWREQUESTS displays the current print requests.

## A SERIES AND B 5000/B 6000/B 7000 CONCEPTS PRINTING SUBSYSTEM OVERVIEW

The PRINT statement can be entered through the ODT, a CANDE session, or a WFL job, to create a print request. Printer file attributes can be specified on a PRINT statement.

Examples:

```
PRINT MYFILE (DESTINATION = "LP4", PRINTCOPIES = 3);
```

```
PRINT ABC/= (SAVEBACKUPFILE = TRUE );
```

### Remote Print System (ReprintS)

ReprintS is an optional extension to the Print System, which sends output to specified remote printers.

All Print System commands can be used for remote printing, just as for on-site printing.

The DESTINATION attribute is used to specify the name of a station controlled by MARC and COMS, where the printing is to occur.

ReprintS requires that COMS or COMS (Entry) be installed on the system also.

Example:

```
PRINT ABC/= (DESTINATION = "STATION LP2032A");
```

### AUTOBACKUP and SYSTEM/BACKUP

PrintS and ReprintS did not exist prior to release 3.6. Instead, an MCP procedure called AUTOBACKUP handled the default printing of backup files, and a program named SYSTEM/BACKUP was provided for situations that required more control over printing.

AUTOBACKUP has been replaced by the PrintS programs and the PS ODT command.

SYSTEM/BACKUP is similar to the PRINT statement, although it is not as flexible as PRINT. SYSTEM/BACKUP still exists because it has some features not found in PrintS (for example, you can start printing a file from a page where a specified string is found in a specified field of the print line).

SYSTEM/BACKUP is initiated by a PB statement in a WFL job, or a ?PB command from the ODT.

Example:

```
PB "MYFILE" LP4 COPIES 3 SAVE
```

Prior to release 3.6, a special MCS (for example, RJE) was required for remote printing



# A SERIES AND B 5000/B 6000/B 7000 CONCEPTS SOFTWARE PRODUCTS OVERVIEW

## UNIT 2

### UTILITIES OVERVIEW

#### Objective

Identify the major A Series utilities.

#### Purpose

You should be familiar with the capabilities of the major utilities so that you can use them when necessary.

#### Resources

A Series System Software Utilities Reference Manual, Section 4 - DUMPALL  
Section 5 - FILECOPY  
Section 6 - FILEDATA  
Section 11 - PATCH  
Section 12 - SORT

A Series System Software Site Management Reference Manual, Section 7 - LOGANALYZER

A Series System Software Support Reference Manual, Section 3 - DCSTATUS  
Section 4 - DUMPANALYZER

# A SERIES AND B 5000/B 6000/B 7000 CONCEPTS UTILITIES OVERVIEW

## Utilities Overview

The paragraphs below describe the major capabilities of the most frequently used standard utilities. Details on the operation and syntax of these utilities is covered in the WFL and Utilities course and/or the Operations course, as appropriate.

Several utilities have been described in previous sections of this Student Guide:

SYSTEM/LOADER	Section 4, System Initialization Concepts
SYSTEM/UTILOADER	Section 4, System Initialization Concepts
SYSTEM/DUMPALL	Section 10, SYSTEM/DUMPALL - Listing Disk Files
SWAPPER	Section 12, Memory Management Overview
SYSTEM/MAKEUSER	Section 14, Security Overview
SYSTEM/GUARDFILE	Section 14, Security Overview
SYSTEM/BACKUP	Section 15, Printing Subsystem Overview

## *Additional Capabilities of SYSTEM/DUMPALL*

SYSTEM/DUMPALL is a comprehensive media conversion program.

Producing listings of disk files in various formats, as described in Section 10, is a form of media conversion.

DUMPALL can also copy files between media such as packs, unlabeled tapes, tapes with non-standard labels, punched cards, and punched paper tape.

DUMPALL can copy only selected portions of files, or concatenate individual files into a large file.

File attributes, such as BLOCKSIZE and AREASIZE, can be changed during the DUMPALL copy process.

DUMPALL can be executed directly through WFL, CANDE, MARC, or the ODT by specifying parameters in a RUN statement. DUMPALL can also be executed interactively at a terminal.

## *SYSTEM/DCSTATUS*

This utility produces run-time reports on the data communications subsystem. Reports can be requested for the entire network, or for specified stations, lines, NSPs, LSPs, etc.

DCSTATUS output can be directed to the line printer or to a terminal.

Interpreting the DCSTATUS reports requires an understanding of NDL II and NSP functions.

## **A SERIES AND B 5000/B 6000/B 7000 CONCEPTS UTILITIES OVERVIEW**

### **SYSTEM/DUMPANALYZER**

SYSTEM/DUMPANALYZER extracts and analyzes user-specified subsets of information from a full memory dump.

This utility may be run in batch mode or interactively at a terminal.

### **SYSTEM/FILECOPY**

This utility simplifies the maintenance of pack files, by creating WFL job files to copy files from packs for backup purposes. FILECOPY is usually executed through WFL, with specifications to:

- Consider files from a certain family or directory for copying.

- Copy files that were created or updated since a certain date and time.

The files that have been updated can be copied daily, and the entire family can be copied less frequently.

Another option of FILECOPY is to copy files that have not been accessed since a certain date, and then remove the files from the pack.

The WFL job created by FILECOPY will execute automatically by default, or it can be saved and started manually later.

### **SYSTEM/FILEDATA**

SYSTEM/FILEDATA produces reports regarding files. The most common reports are:

- Filenames, which is a hierarchical list of files (similar to a PD, but with more information).

- Map, which lists all the files on a family, and the addresses of the rows of each file.

- Checkerboard, which shows the location and size of all in-use and available disk space.

- Attributes, which lists the attributes of 1 or more files.

- Tape directory, which lists the names of the files on a library tape.

FILEDATA can be executed in several ways:

- Directly through WFL, CANDE, MARC, or the ODT by specifying parameters in a RUN statement.

- Indirectly through the CANDE LFILES command, to display file attributes on the screen.

- Indirectly through the DIR (Directory) and TDIR (Tape Directory) ODT commands.

## **A SERIES AND B 5000/B 6000/B 7000 CONCEPTS UTILITIES OVERVIEW**

### ***SYSTEM/LOGANALYZER***

SYSTEM/LOGANALYZER produces reports from the entries in the SYSTEM/SUMLOG file. Examples of possible reports are:

- Log entries for a specified time range.
- Log entries for a specified job number.
- Log entries for tasks that were DSed (DiScontinued).
- Errors on a specified tape drive.

This utility can be executed through a RUN statement with parameters, or through the LOG command with parameters at the ODT or a CANDE terminal.

The output can be directed to a line printer, or to the terminal or ODT.

### ***SYSTEM/PATCH***

This utility merges several individual patch files into a single patch file, which can be input to a compiler along with the original source file.

SYSTEM/PATCH is the preferred method for making changes to the MCP or other system software, because the patches are kept in separate files, rather than losing their identity in the source file. This keeps a record of all the patches made to the file and allows patches to be removed easily if necessary,

SYSTEM/PATCH can also be used to control patches to application software.

### ***SORT***

SORT is a procedure in the MCP that sorts a file or set of records into a single file of ordered records, based on input parameters.

SORT can be activated through application programs written in ALGOL, COBOL, COBOL74, and PL/I.

SORT can also be executed through the SORT compiler, which allows the user to enter SORT parameters without writing a complete application program. The SORT compiler is a chargeable program.

# A SERIES AND B 5000/B 6000/B 7000 CONCEPTS SOFTWARE PRODUCTS OVERVIEW

## UNIT 3

### DMS II OVERVIEW

#### Objective

Identify the major software components of DMS II.

#### Purpose

A large percentage of A Series installations use DMS II databases. You should be familiar with the major capabilities and components of DMS II.

#### Resources

A Series DMS II Data and Structure Definition Language (DASDL), Section 1 - Introduction to  
DMS II

A Series DMS II Utilities and Operations Guide, Section 5 - Controlling Data Base Operations  
Section 11 - Monitoring the Data Base  
Section 12 - Analyzing the Data Base

A Series Advanced Data Dictionary System (ADDS) User's Guide

A Series Extended Retrieval with Graphic Output (ERGO) User's Manual

A Series DMS II Inquiry Software Operation Guide

A Series DMS II DataAid User's Guide

# A SERIES AND B 5000/B 6000/B 7000 CONCEPTS DMS II OVERVIEW

## DMS II Database Management System

A **database** is a centralized collection of data placed into one or more files. Multiple application programs running on-line or in batch mode may all access the database concurrently.

Data Management System (DMS II) is a **database management system**, the software required to control and manage the files in the database. Some of the major functions of DMS II are to:

Centralize the definition and organization of the data so that each application program does not need to redefine the contents of the files.

Assist in the design, development, and maintenance of application systems, by controlling changes to the database definition.

Insure that all files in the database are current (for example, do not allow an old database file to be used with other more recently updated files).

Provide programs to copy the database to and from backup tapes.

Provide recovery routines to be used in case of program or system failure.

## *Database Creation*

Creation and maintenance of the database are considered **database administration** functions.

To create or change the database, the database administrator (DBA) writes the descriptions of the files and their records in a Data and Structure Definition Language (DASDL) source file, as shown in Figure 15-1.

The DASDL compiler checks the DASDL source for syntax errors, and creates a file called DESCRIPTION/<database name>. The description file is not an object code file, but a translation of the DASDL that can be read by other DMS software programs.

Information from the description file and standard DMS II symbolics, or source files, are compiled together to produce the **tailored** software, specifically for this database.

Tailored software includes a library called DMSUPPORT/<database name>, which contains details needed to access this particular database.

Recovery programs (for example, RECOVERY/<database name>) are also tailored for this database.

**A SERIES AND B 5000/B 6000/B 7000 CONCEPTS  
DMS II OVERVIEW**

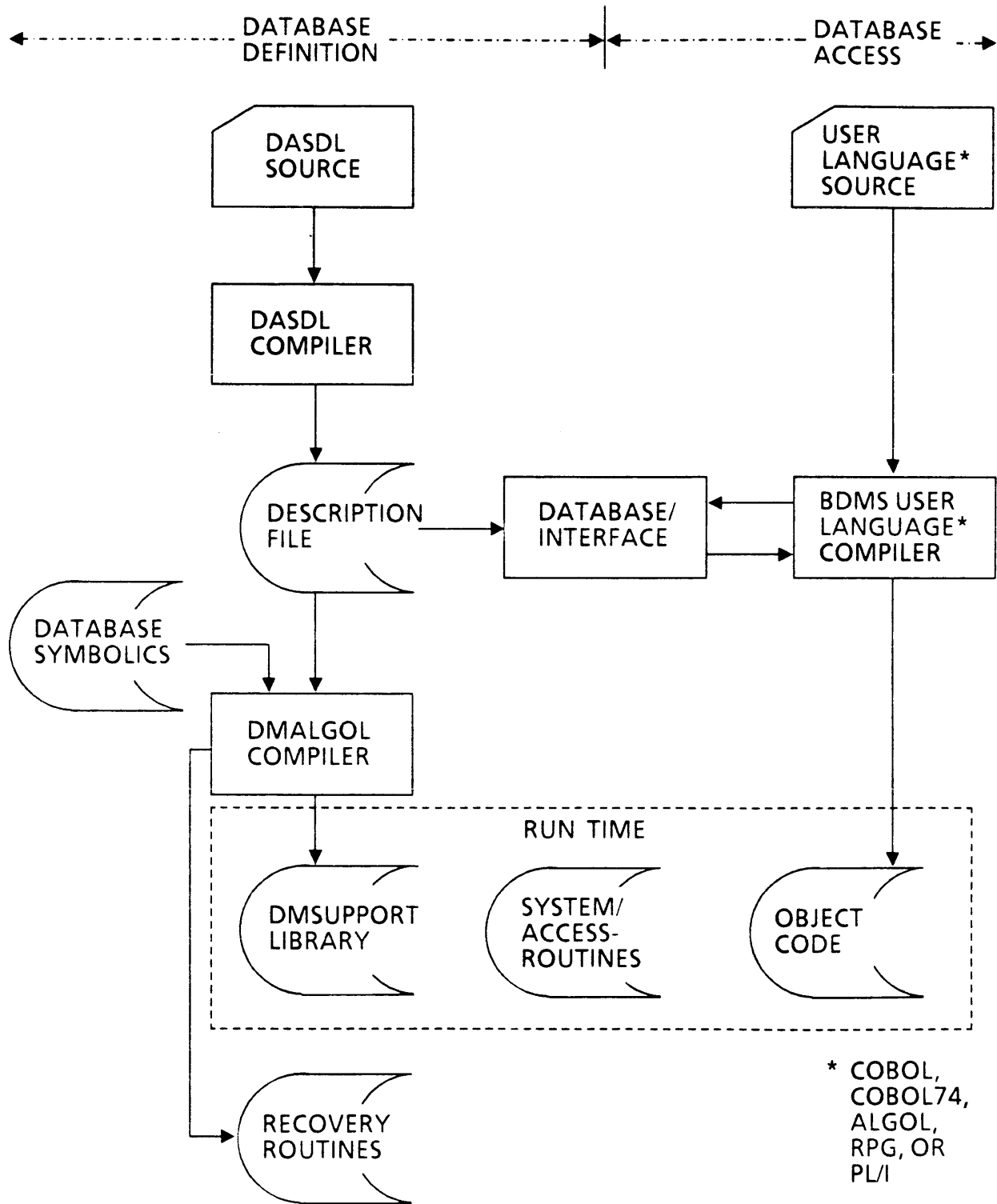


Figure 15-1 Components of DMSII

## A SERIES AND B 5000/B 6000/B 7000 CONCEPTS DMS II OVERVIEW

### *DataBase Access*

In order to access and update the database, application programs can be written in several user languages (ALGOL, COBOL, COBOL74, RPG, and PL/I).

The user language compilers are named SYSTEM/BDMSALGOL, SYSTEM/BDMSCOBOL74, etc.

The user language compilers include extensions to allow special verbs and constructs for accessing the database.

The application program invokes the database and the desired files by name. The user language compiler initiates the DATABASE/INTERFACE program to read the file layout information from the description file.

When the application program is executed, it communicates with two DMS II software programs, as shown in Figure 15-1.

SYSTEM/ACCESSROUTINES is a standard program that receives requests from application programs to read or write in the database, performs the requests, and returns the results to the programs. The same SYSTEM/ACCESSROUTINES services all programs running against all databases.

The DMSUPPORT library that was tailored for this database is in the mix whenever any application program is using the database. The ACCESSROUTINES call the DMSUPPORT library to provide details about this particular database.

The standard program SYSTEM/DMUTILITY allows the operators to copy the database to and from backup, and allows the database administrator to initiate various forms of recovery for the database.



# A SERIES AND B 5000/B 6000/B 7000 CONCEPTS DMS II OVERVIEW

## Software Products Related to DMS II

The products described below extend the capabilities of DMS II.

### *Advanced Data Dictionary System (ADDS)*

The Advanced Data Dictionary System is a menu-driven InterPro product that allows the user to maintain a DMS II database to store all the information descriptions used on the system.

Information descriptions can be stored for DSMII databases, conventional files, COBOL74 application programs, and Screen Design Facility (SDF) formats.

The status of each entity is kept in the dictionary: **Definitional, Test, Production, or Historical.**

The dictionary maintains a version number for each entity, to control system integrity.

ADDS screens may also be used to describe a new database, and then generate the DASDL from the information descriptions in the dictionary.

### *Extended Retrieval with Graphic Output (ERGO)*

ERGO is a query program that provides on-line access to DMS II databases in tabular and graphic forms. It is intended for use by managers, programmers, analysts, and support personnel involved in preparing reports from DMS II databases.

The user can specify parameters for the desired reports through a series of menus, or through complete commands.

Reports can be produced on the screen, or on the printer.

The output can be formatted in a table, or in a graph such as a bargraph or a histogram.

Information from a database can be extracted into a conventional file, which may be used in another application, or transferred to a B 20 using Data Transfer System (DTS).

ERGO is not tailored for any database, although it requires a program called DMINTERPRETER/<database name> to be compiled for each database used in reports.

ERGO can produce a single report from multiple databases and conventional files.

ERGO allows the user to browse through records in the database, and to update records if DMINTERPRETER was compiled to allow updates.

The database administrator indicates during the DMINTERPRETER compilation process whether users will be allowed to update the database through ERGO.

## A SERIES AND B 5000/B 6000/B 7000 CONCEPTS DMS II OVERVIEW

### *DMS II Inquiry*

DMS II Inquiry is also an on-line, interactive system that can be used to examine or modify information in a DMS II database.

Report specifications are given to Inquiry as complete commands.

Reports can be produced on the screen, or on the printer.

The output is formatted in columns in a table.

Inquiry must be compiled for each specific database. The default name of the object code is OBJECT/INQUIRY/<database name>.

Inquiry includes data from only 1 database on each report.

Inquiry allows the user to browse through records in the database.

The database administrator indicates during the Inquiry compilation process whether users will be allowed to update the database through Inquiry.

### *DataAid*

DataAid automates the process of defining, generating, and accessing a database. DataAid menus provide a linkage between ADDS to define the database, database compilation to create the tailored software, and ERGO to produce reports. DataAid has 2 basic applications:

#### DataBase Administration

DataAid could be used in application system design, to create a prototype database, enter representative data, and produce sample reports.

DataAid can simplify the execution of SYSTEM/DMUTILITY, by providing menus for database backup and recovery.

#### Personal DataBase Usage

DataAid can be used to manage a personal database, containing a small amount of non-production data. For example, a manager could keep project status records in a personal database.

## **A SERIES AND B 5000/B 6000/B 7000 CONCEPTS DMS II OVERVIEW**

### ***DBANALYZER***

DBANALYZER produces statistical reports on the current status of the database (for example, the number of active and deleted records in each file). This information helps the database administrator to determine when database attributes should be changed to improve performance, and when reorganization is needed (for example, to physically remove deleted records).

### ***DBMONITOR***

DBMONITOR displays statistics on a terminal about the current usage of the database (for example, number of user programs, current memory usage). The database administrator could change specified database parameters dynamically, to see their effect on the performance of the database.

# A SERIES AND B 5000/B 6000/B 7000 CONCEPTS SOFTWARE PRODUCTS OVERVIEW

## UNIT 4

### OTHER SOFTWARE PRODUCTS

#### **Objective**

Identify other major software products available for the A Series systems.

#### **Purpose**

You should be aware of the software products and capabilities available on the A Series systems so that you can make plans regarding their use.

#### **Resources**

SMF II System Resource Management Manual

SMF II Site Management Manual

A Series Site Management Reference Manual, Section 2A - BARS

LINC II Reference Manual

Reporter III Report Language User's Guide

A Series MultiLingual System User's Guide

A Series ALGOL Test and Debug System (TADS) User's Guide

A Series COBOL74 Test and Debug System (TADS) User's Guide

A Series Burroughs Network Architecture (BNA) User's Guide

Office Management System (OMS II) Planning and Installation Guide

A Series Intelligent Distributed Editor (IDE) User's Guide

A Series Data Transfer System (DTS) User's Guide

## A SERIES AND B 5000/B 6000/B 7000 CONCEPTS OTHER SOFTWARE PRODUCTS

### Other Software Products

Some of the major software products which have not been introduced previously in the course are grouped by function below.

#### *Performance Evaluation*

##### *Site Management Facility II (SMF II)*

SMF II produces reports on system performance and utilization, using data consolidated from the system log files and other sources.

Several standard report specifications are provided with the SMF II software.

SMF II users may define their own report specifications using a language called QUERY.

Users may also specify the types of data that should be extracted from multiple log files and consolidated for reports.

Two modules of SMF II are provided with the standard system software package.

The Hardware module reports on hardware errors in the mainframe or peripherals.

The Availability module reports on the availability of the system, using data entered by an operator, rather than data from log files (for example, the system was deliberately turned off for 2 hours because the air conditioning was broken).

Two modules of SMF II are chargeable software.

The Workload module produces reports on the work done by various usercodes, chargecodes, jobs, and tasks, using data from the log files.

The Utilization module reports on the usage of the processor, memory, and I/O devices.

The Utilization module includes a Sampler program that collects real-time performance data, and stores it in disk files for subsequent analysis by a QUERY program.

Sampler can also display current system utilization data on a terminal at regular intervals.

## **A SERIES AND B 5000/B 6000/B 7000 CONCEPTS OTHER SOFTWARE PRODUCTS**

### *SYSTEM/BARS*

SYSTEM/BARS is a utility that monitors the system performance and displays it periodically in the form of numeric values and bar graphs.

The user can specify the frequency of the screen updates.

The user can also specify the type of information to be displayed (for example, amount of save memory, amount of available memory, number of tasks in the mix).

### *Application Development and Testing Software*

#### *Logic and Information Network Compiler II (LINC II)*

LINC II is a fourth-generation application development language.

LINC II allows application systems to be developed more quickly than traditional programming methods, and facilitates prototyping in the development process.

From a single set of source specifications, LINC II generates NDL II, parameters for COMS or GEMCOS, DASDL and tailored database software, and a COBOL74 application program for on-line file maintenance and transaction posting.

Additional specifications can be written to produce reports on the data in the databases maintained by LINC II applications.

#### *REPORTER III*

REPORTER III allows programmers to create reports on data in databases or in conventional files, without writing application programs.

The user specifies the format and sequence for the report by giving simple English-like specifications.

REPORTER III generates a COBOL74 program to produce the report.

Output can be directed to a printer or to a terminal

## **A SERIES AND B 5000/B 6000/B 7000 CONCEPTS OTHER SOFTWARE PRODUCTS**

### *MultiLingual System (MLS)*

The MultiLingual System is a mechanism for providing output messages in several languages, for both application programs and system software.

These messages can inform the computer user or operator about the status of the system, or about the kind of data to enter, in the translated language.

MLS is used by Burroughs to translate system software output messages for international use.

MLS would be useful to a company with branches and computer users in several countries.

### *Test and Debug System (TADS)*

TADS is an interactive test and debugging system available for the ALGOL, COBOL74, and FORTRAN77 languages. TADS allows the programmer to:

- Control program execution, and suspend the execution of the program at selected points.

- Inquire into the values of program variables when the program is suspended.

- Change the values of variables when the program is suspended.

- Collect program coverage information (for example, 90% of the statements were executed).

- Interrogate the history of program execution (for example, procedure A was invoked by procedure B).

The output from TADS can be directed to a printer or disk file, instead of to a terminal.

TADS also provides on-line help documentation as requested by the programmer.

## **A SERIES AND B 5000/B 6000/B 7000 CONCEPTS OTHER SOFTWARE PRODUCTS**

### *Networking Software*

#### *Burroughs Network Architecture (BNA)*

BNA software is designed to manage distributed networks of Burroughs mainframes. BNA provides the services similar to NDL II and MCS capabilities across the network.

BNA considers each mainframe to be a cooperating peer, so there are no host and user restrictions.

Some of the capabilities of BNA are:

File Transfer.

Inter-process Communication between Systems.

Resource Sharing.

Virtual Terminals.

Message Routing.

#### *Office Management System II (OMS II)*

OMS II is an application that can connect an A Series mainframe, B20 microcomputers, and OFIS Writers in a network.

The goal of OMS II is to reduce paper handling in the office.

OMS II includes an electronic mail capability, which is implemented through a DMS II database.

### *Workstation Integration*

#### *Intelligent Distributed Editor (IDE)*

IDE is an editor that provides editing functions for ALGOL, COBOL74, and FORTRAN77 programs.

IDE can be used on a B 25 or ET 2000 in conjunction with INFOVIEW, to load source files into the microcomputer, maintain them on the workstation, and then send them back to the host for compilation and execution.

IDE can also be executed at the host, for use with a standard Burroughs terminal.

#### *Data Transfer System (DTS)*

DTS transfers files from an A Series host to a B 20 microcomputer. The data can then be manipulated with Multiplan or a word processing utility to produce reports, graphs, and projections.



**A SERIES AND B 5000/B 6000/B 7000 CONCEPTS  
OTHER SOFTWARE PRODUCTS**

**Practice**

Match the programs and software products on the left with the capabilities on the right.

- |          |                           |                                                                                                   |
|----------|---------------------------|---------------------------------------------------------------------------------------------------|
| _____ 1. | SYSTEM/LOGANALYZER        | a. Produces user-defined reports on the performance and utilization of the system.                |
| _____ 2. | ReprintS                  | b. Produces reports on disk files and their attributes.                                           |
| _____ 3. | DASDL                     | c. Produces tabular or graphic reports, using the data in databases or conventional files.        |
| _____ 4. | SMF II                    | d. Defines the structure of DMS II databases.                                                     |
| _____ 5. | SYSTEM/FILEDATA           | e. Produces reports of selected information from the system logs, on a printer, terminal, or ODT. |
| _____ 6. | LINC II                   | f. Services all requests to read and write DMS II databases.                                      |
| _____ 7. | ERGO                      | g. Prints files on remote printers controlled by COMS and MARC.                                   |
| _____ 8. | SYSTEM/<br>ACCESSROUTINES | h. Generates application systems from a single set of specifications.                             |

**A SERIES AND B 5000/B 6000/B 7000 CONCEPTS  
OTHER SOFTWARE PRODUCTS**

This page left blank for formatting.

**SECTION 16**  
**SOFTWARE INSTALLATION**

# **A SERIES AND B 5000/B 6000/B 7000 CONCEPTS SOFTWARE INSTALLATION**

## ***INTRODUCTION***

### **Section Objective**

Identify the major steps in A Series software installation.

### **Purpose**

This is an **optional** section, to be discussed in class if the students are interested in software installation procedures. It **provides a summary of considerations and procedures for installing the system software.**

### **Unit Objectives**

Identify the major steps in A Series software installation.

# **A SERIES AND B 5000/B 6000/B 7000 CONCEPTS SOFTWARE INSTALLATION**

## **UNIT 1**

### **SOFTWARE INSTALLATION**

#### **Objective**

Identify the major steps in A Series software installation.

#### **Purpose**

You should be aware of the general procedures for installing system software releases.

#### **Resources**

A Series Mark 3.6.0 System Software Installation Guide

A 3 System Software 3.6 Installation Guide

A 9 System Software 3.6 Installation Guide

A 10 System Software 3.6 Installation Guide

A Series System Software Support Reference Manual, Section 12 - Software Compilation

# A SERIES AND B 5000/B 6000/B 7000 CONCEPTS SOFTWARE INSTALLATION

## Planning for a New Installation

You should begin making plans for the items below before the hardware is installed.

1. **Develop a plan for placement of files on packs. Determine which families will be used for system software, and which for application program and data files. Allow for printer backup and overlay files also.**
2. **Define the job queues that will be needed. Determine the attributes of each queue (for example, the maximum number of jobs and tasks from each queue that will be allowed in the mix at a time, the maximum processor time that will be allowed for jobs from each queue).**
3. **Define the usercodes that will be needed by individuals and groups. Assign attributes to the usercodes (for example, family substitution statements, privileged or nonprivileged).**

## Installing a System Software Release

Perform the following steps when installing software on a new system, or when installing a new release on an existing system.

1. **Consult the documentation provided with the software release for information on installing or converting to the release.**
  - a. **Read the hardcopy Support Release Document enclosed with the release tapes.**
  - b. **Execute OBJECT/LISTNOTES to print the Programming and Documentation (P & D) Notes. If you are installing a new system, you may order printed P & D Notes from the Publications Center, or print them on an existing system.**
2. **Install the Software.**
  - a. **Put the MCP on the desired pack. This may be done by a Coldstart , or by COPY if an MCP has already been loaded to another pack.**

**If COPY is used, also enter CM for the new MCP, change the BOOTUNIT (B 5900/B 6900) or HLUUNIT (other systems) to be on that pack, and Halt/Load to the new MCP.**
  - b. **Use the TR (Time Reset) and DR (Date Reset) ODT commands to set the system time and date**

## **A SERIES AND B 5000/B 6000/B 7000 CONCEPTS SOFTWARE INSTALLATION**

- c. Use the RC (Reconfigure Disk) ODT command to reconfigure the packs according to the plan above.
  - 1. For a new installation, the field engineers may have to Initialize and Verify (IVR) the packs using a Peripheral Test Driver (PTD) function.
  - 2. For an existing system, do not RC a pack containing valuable files. Copy the files elsewhere, or use the LB (Relabel Disk) to relabel the pack without destroying its contents.
  
- d. COPY the compilers, utilities, libraries, traintables, and other system and environmental software to the desired packs.
  
- e. Use the SL (System Library) ODT command to specify the titles of the system libraries that will be required by the system software.
  
- f. Set and reset the system options as desired through the OP (Options) ODT command.
  
- g. Use the DL (Disk Location) ODT command to indicate the families where system files such as logs, printer backup files, overlay files, USERDATAFILE, JOBDESC, and SORT work files will be placed.
  
- h. Use the SB (Substitute Backup) ODT command to set substitute printer backup destinations as desired.
  
- i. Use the SI (System Intrinsic) ODT command to specify the title of the SYSTEM/INTRINSICS file.
  
- j. Use the MQ ODT command to make and/or modify the job queues as desired. The job queues should be intact on an existing system, unless the pack containing JOBDESC was RCed.
  
- k. If desired, set the memory factors through the SF (Set Factors) ODT command.
  
- l. Initialize the MARC system by entering ??MARC at the ODT.

## A SERIES AND B 5000/B 6000/B 7000 CONCEPTS SOFTWARE INSTALLATION

- m. For a new system, run SYSTEM/MAKEUSER through MARC at the ODT to create some usercodes and specify their attributes.
  - 1. Create at least one privileged usercode.
  - 2. Most sites disable the MU ODT command to prevent additional usercodes from being created at the ODT.
  - 3. The remaining usercodes can be added by a privileged user at a terminal later.
  
- n. Install the data comm software.
  - 1. Copy the required FIRMWARE/NSP and FIRMWARE/LSP files to the Halt/Load pack.
  - 2. Run IDC through MARC to enter the lines, modems, terminals, and stations for the system.
  - 3. If necessary, modify and compile the NDL II for the required protocols.
  - 4. If necessary, run COMS UTILITY through MARC to create or update the configuration file. Enter the windows, programs, and other elements required for the system.
  
- o. Create another Halt/Load pack to be used if necessary.
  - 1. COPY the MCP to another pack.
  - 2. Do a CM to establish the link to the MCP on this pack.
  
- p. COPY all the packs containing system software to tapes for backup.
  
- q. If you are installing a new release on an existing system, refer to the P & D Notes for instructions on converting the DMS II databases to the new level.



## **A SERIES AND B 5000/B 6000/B 7000 CONCEPTS SOFTWARE INSTALLATION**

### **Installing a Support Release**

A Support Release is an intermediate release of selected system software and/or patch files to be applied against the original release. A Support Release may or may not contain new features.

If the Support Release includes object code files, copy the new files to your packs and remove the old files. A new MCP will also require a CM (Change MCP) ODT command.

The Support Release may include patch files called PATCHESFOR/ <software name > , but may not include the corresponding code files, especially for chargeable software products. If your installation requires any of these software products, use the following procedure to install the new version.

1. COPY the PATCHESFOR files to a pack that contains some available space.
2. COPY the symbol files for these software products from the original release tapes to the same pack.
3. Start the job WFL/COMPILE/SOFTWARE. This job will look for any PATCHESFOR files on the pack, and then patch and compile all the corresponding software.
4. COPY the compiled object files to the desired packs, and also to backup tapes.
5. Remove or change the names of the PATCHESFOR files so they are not used again by accident later.

**A SERIES AND B 5000/B 6000/B 7000 CONCEPTS  
SOFTWARE INSTALLATION**

**Practice**

Match the programs and commands on the left with the functions on the right.

- |          |                          |                                                                             |
|----------|--------------------------|-----------------------------------------------------------------------------|
| _____ 1. | OBJECT/LISTNOTES         | a. Reconfigures disk packs.                                                 |
| _____ 2. | SL                       | b. Specifies the disk locations of special types of files.                  |
| _____ 3. | MQ                       | c. Specifies the file names where certain library entry points are located. |
| _____ 4. | DL                       | d. Prints software documentation such as P & D notes from tapes.            |
| _____ 5. | CM                       | e. Makes or modifies job queues.                                            |
| _____ 6. | OP                       | f. Applies patches and compiles system software.                            |
| _____ 7. | RC                       | g. Specifies the file title of the MCP.                                     |
| _____ 8. | WFL/COMPILE/<br>SOFTWARE | h. Sets or resets system options.                                           |

**SECTION A 3**

**A 3 HARDWARE OVERVIEW**

# A SERIES AND B 5000/B 6000/B 7000 CONCEPTS

## A 3 HARDWARE OVERVIEW

### **Objective**

Identify the major hardware elements that constitute the A 3 system.

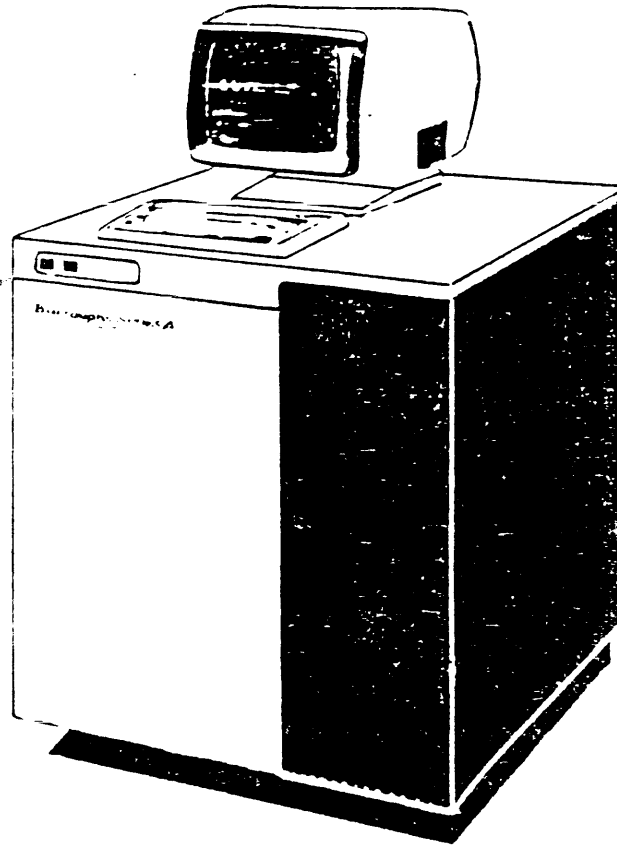
### **Purpose**

The A 3 system consists of a series of logical units housed in from 1 to 3 cabinets, and 1 or more Operator Display Terminals. This unit will introduce you to these elements.

### **Resources**

A 3 System Reference Manual Volume 1

**A SERIES AND B 5000/B 6000/B 7000 CONCEPTS  
A 3 HARDWARE OVERVIEW**



**Burroughs A3 System**

Figure A3-1 A 3 System

# A SERIES AND B 5000/B 6000/B 7000 CONCEPTS

## A3 HARDWARE OVERVIEW

### Central System Components

The A 3 central system consists of the Central Processing Unit, the Memory Subsystem, the Maintenance Subsystem, and the Input/Output and Data Communications Subsystem.

### Central Processing Unit (CPU)

The A 3 system includes 1 or 2 CPUs, each of which contains the hardware modules described below.

#### 1. Register File

The Register File is a series of 32 1-word registers which store the current processor state and the primary data items for the current operator.

#### 2. Data Section

The data section includes the Arithmetic/Logic Unit (ALU), which performs all the logical and arithmetic functions in the system, the Rotate/Merge Logic, which shifts bits in words, and the Time-of-Day clock.

#### 3. Condition Logic Module

The Condition Logic Module performs four functions:

- a. condition logic, to determine if the current operation should be aborted;
- b. tag storage and selection, to identify the type of data being operated on;
- c. counter and timer implementation, to check for time limits and time-outs on operators and I/Os;
- d. address decoupling, to decode stack addresses from the Data Section.

#### 4. Code-Isolate

The Code-Isolate module is a Program Controller (PC) that manages the object code stream. It captures object code, decodes the operators and parameters, and passes the operators to the Micro-Address Module.

#### 5. Micro-Address Module (MAM)

The MAM converts the operator passed from the Code-Isolate Module to the address of the corresponding Control Store algorithm.

#### 6. Control Store (CS)

The CS is programmable microcode memory which contains algorithms to execute operators, interrupts, and I/O routines. Because the CS is programmable, its contents can vary with different software releases as necessary.

A SERIES AND B 5000/B 6000/B 7000 CONCEPTS  
A3 HARDWARE OVERVIEW

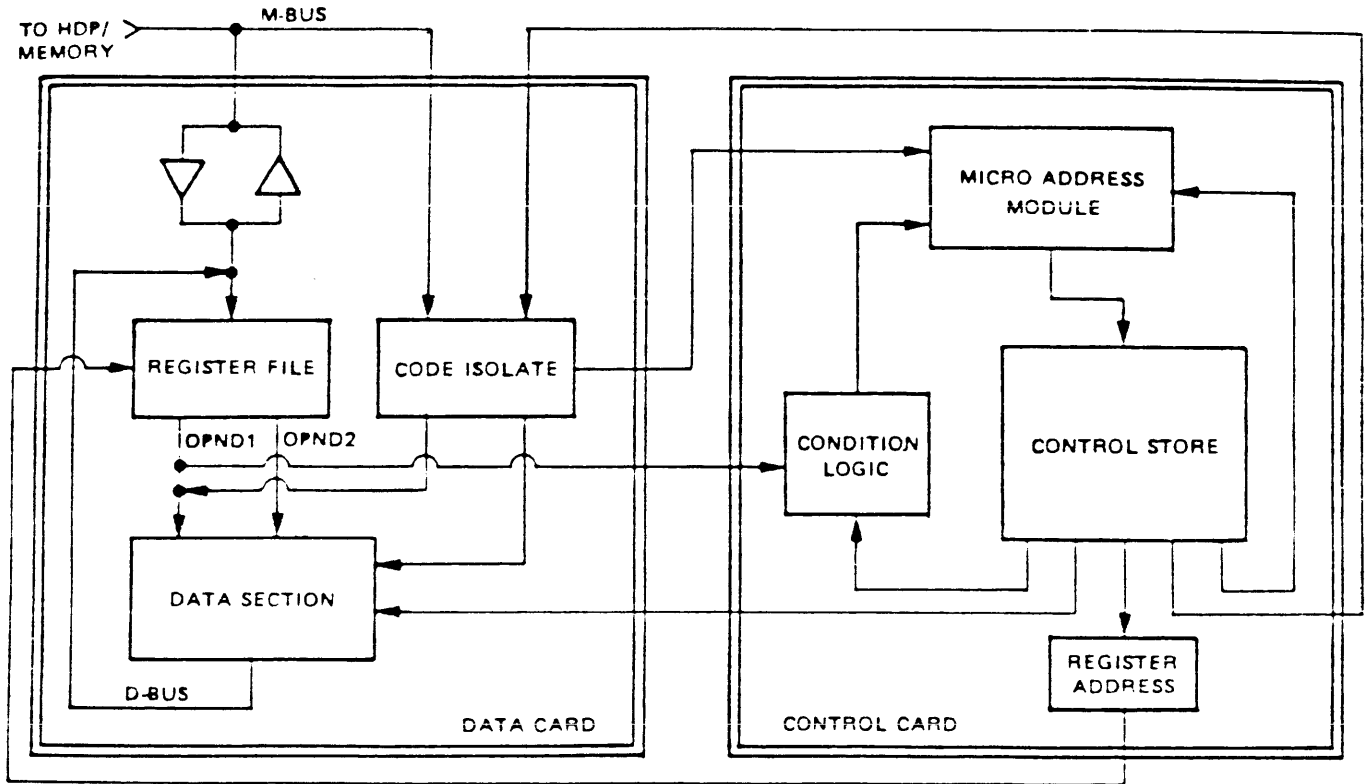


Figure A3-2 A 3 CPU Block Diagram

## **A SERIES AND B 5000/B 6000/B 7000 CONCEPTS A3 HARDWARE OVERVIEW**

### ***Memory Subsystem***

The A 3 memory subsystem features 256K RAM chip technology for speed, capacity, and reliability. The maximum memory size for the various A 3 models ranges from 1 million to 8 million words (6 million to 48 million bytes), in increments of 1/2 million words (3 million bytes).

Under MCP, the A 3 can address 1 million words of memory at a time. A 3 systems with more than 1 million words of memory are configured via ODT commands into Address Spaces (ASN = Address Space Number) and Environment Components. Each Address Space consists of a shared component, which is common to all the Address Spaces, and a local component, which is specific to that address space. The shared plus the local components of an Address Space may occupy a maximum total of 1 million words. Each program in the mix will be assigned to an Address Space, which will be accessed while that program has control of the processor.

Under MCP/AS, an A 3 can access all of its memory at any time. Address Spaces are not used with MCP/AS.

Single processor A 3 systems have a Memory Control Unit (MCU) which interfaces the CPU to the memory subsystem. The MCU checks the integrity of data read from the memory storage cards, and logs memory failures.

Dual processor A 3 systems use 2 Shared Access Memory (SAM) controls to interface the 2 central processors to main memory. The SAMs perform integrity checking functions similar to the MCU functions in a single processor system. The SAMs exchange data via the SAM bus, and communicate status via additional control lines.

### ***Maintenance Subsystem***

The User Interface Processor (UIP, pronounced "whip") is the major component of the maintenance subsystem. The UIP performs system initialization and diagnostic testing for the A 3. System Initialization can be accomplished by depressing a single button on the A 3 cabinet. The ODT becomes a Maintenance Diagnostic Terminal (MDT), so that field engineers can interact with the UIP to execute diagnostic tests.

An RS232 Remote Diagnostic Link to the UIP allows diagnostic programs to be executed by Burroughs personnel located at remote sites. The remote link can be disabled with a key switch on the A 3 cabinet.

The Power Control Card (PCC) maintains date and time-of-day information using a real-time clock with battery backup. This allows the A 3 to be powered on or off automatically at specified times.



# A SERIES AND B 5000/B 6000/B 7000 CONCEPTS

## A3 HARDWARE OVERVIEW

### Input/Output and Data Comm Subsystem

All A Series systems, as well as B 5900s, B 6900s, and B 7900s use Burroughs Universal I/O for both peripheral and data communications management.

Each peripheral subsystem is connected to a Data Link Processor (DLP) which is designed to control that specific type of peripheral. Some peripherals (e.g. image printers) can have only 1 device attached to each DLP, but other peripherals (e.g. disk packs, tapes) can have multiple devices attached to a single DLP through an exchange and/or controller. The A 3 offers a combination Printer/Tape DLP which connects to both a line printer and a tape streamer.

A Data Communications DLP is available on the A 3 for small- to-medium size networks. A system can have 1 to 6 Data Comm DLPs, depending on the model of A 3, with each Data Comm DLP controlling 4 lines. A 3 systems with larger networks or specialized protocols require two other types of DLPs: Network Support Processors (NSPs) and Line Support Processors (LSPs).

The DLPs are grouped into I/O-Data Comm (IODC) base modules. Each IODC base can hold up to 8 DLPs, depending on the number of circuit boards in the various types of DLPs used. Different A 3 models allow different numbers of IODC bases and I/O DLPs. The A 3 main cabinet can house 2 IODC bases; one or two expansion cabinets may be added to hold additional IODC bases.

The internal IODC base module is connected by a Data Link Interface (DLI) to the Host Dependent Port (HDP); additional IO bases are connected via Message Level Interface (MLI) cables to the HDP.

The HDP provides the system interface to the Universal I/O subsystem, and checks parity for the data being transferred.

Data to be output to a device is transferred from memory to the HDP, then through the DLI or MLI to the IODC base, to the DLP, and finally to the designated device. Data input from peripherals follows the reverse route.

The A 3 disk subsystem includes at least 1 spindle of 8-inch Winchester disk, with a capacity of 123 million bytes per spindle, built into the main cabinet. Depending on the model of A 3, up to 4 Winchester disks may be integrated into the main cabinet, and 4 more disks may be integrated into each expansion cabinet.

**A SERIES AND B 5000/B 6000/B 7000 CONCEPTS  
A3 HARDWARE OVERVIEW**

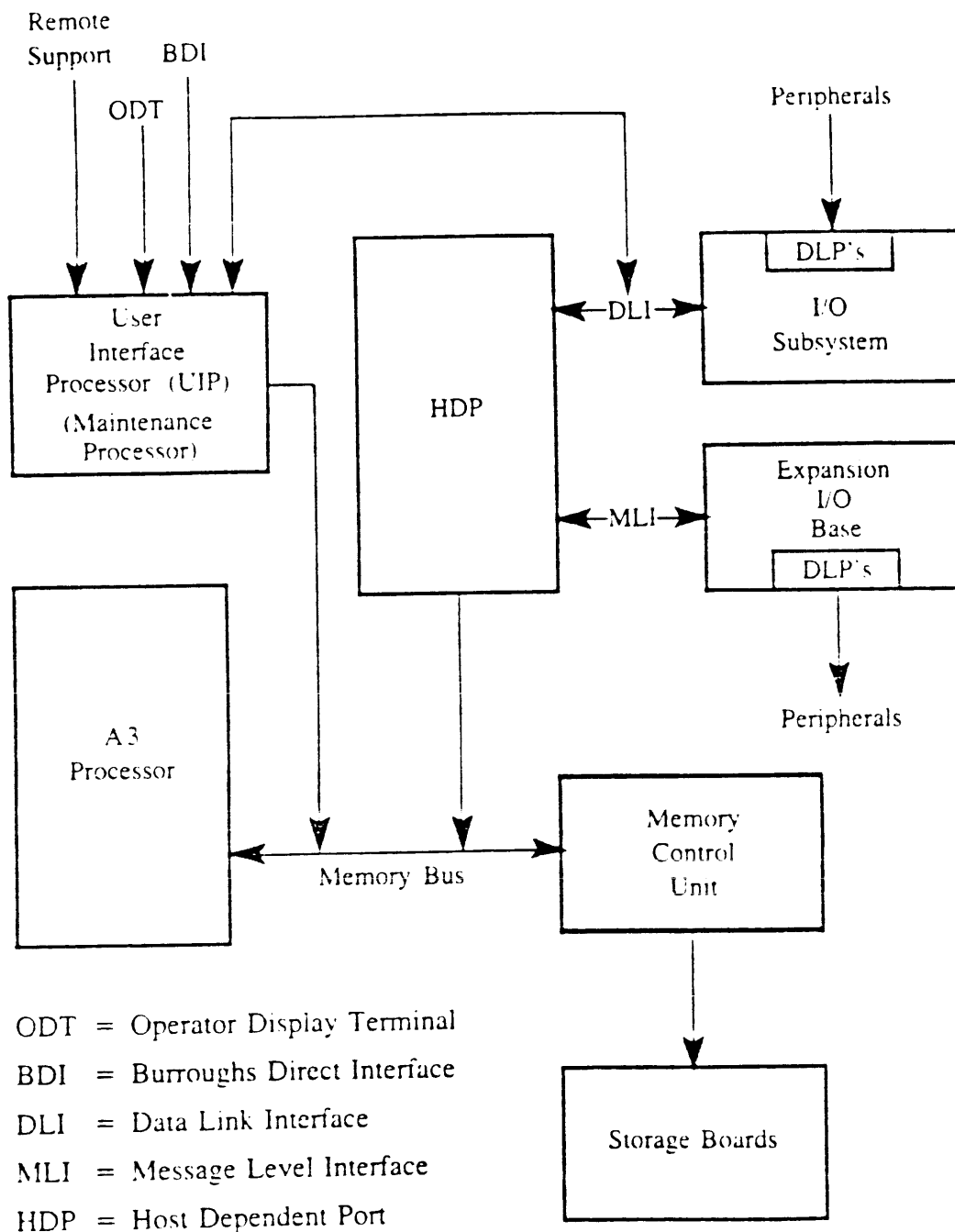


Figure A3-3 A3 System Configuration

**A SERIES AND B 5000/B 6000/B 7000 CONCEPTS  
A3 HARDWARE OVERVIEW**

**Practice**

Match the A 3 system elements on the left with the descriptions on the right.

- |                                  |                                                                          |
|----------------------------------|--------------------------------------------------------------------------|
| _____1. Control Store            | a. Interfaces the central system to the I/O Subsystem.                   |
| _____2. Data Link Processor      | b. Performs system initialization and diagnostic testing.                |
| _____3. User Interface Processor | c. Interfaces the CPU to the Memory Subsystem.                           |
| _____4. Host Dependent Port      | d. Contains the Arithmetic Logic Unit.                                   |
| _____5. Code-Isolate             | e. Is hardware designed to control a specific type of peripheral device. |
| _____6. Memory Control Unit      | f. Stores the operator microcode.                                        |
| _____7. Data Section             | g. Captures object code, and provides input to the Micro-Address Module. |

**A SERIES AND B 5000/B 6000/B 7000 CONCEPTS  
A3 HARDWARE OVERVIEW**

This page left blank for formatting.

***SECTION A 9***

***A 9 HARDWARE OVERVIEW***

# A SERIES AND B 5000/B 6000/B 7000 CONCEPTS A 9 HARDWARE OVERVIEW

## **Objective**

Identify the major hardware elements that constitute the A 9 system.

## **Purpose**

The A 9 system consists of a series of logical units housed in from 3 to 5 cabinets, and 2 Operator Display Terminals. This unit will introduce you to these elements.

## **Resources**

A 9 System Reference Manual Volume 1

A SERIES AND B 5000/B 6000/B 7000 CONCEPTS  
A 9 HARDWARE OVERVIEW

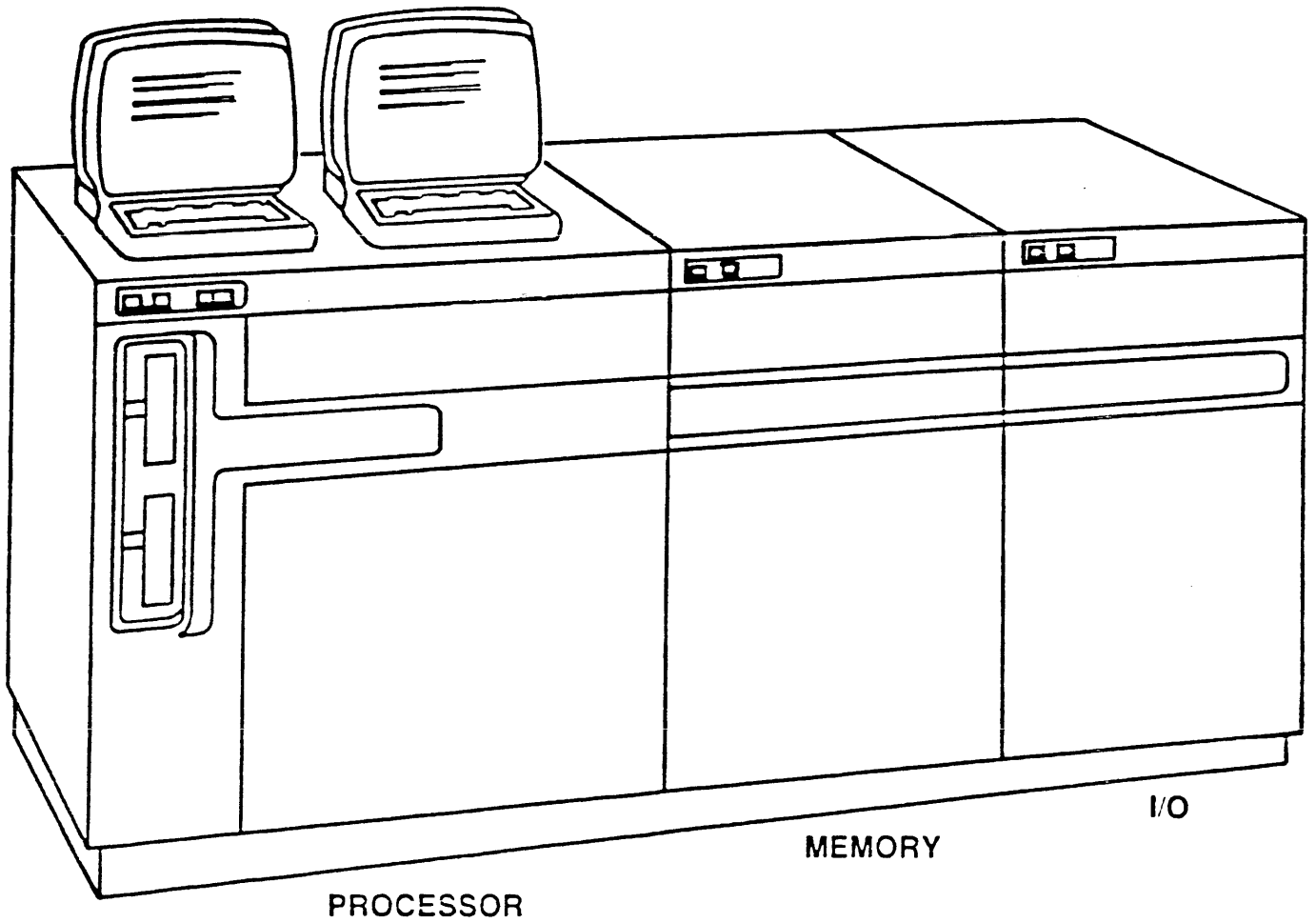


Figure A9-1 A 9 System

# A SERIES AND B 5000/B 6000/B 7000 CONCEPTS A 9 HARDWARE OVERVIEW

## Central System Components

The A 9 central system consists of the Multiple Logical Processor, the Memory Subsystem, the Maintenance Subsystem, and the Input/Output and Data Communications Subsystem. Modular power supplies are distributed throughout the A 9 cabinets, with externally accessible power control switches located on the cabinets.

### *Multiple Logical Processor (MLP)*

The A 9 central processor is called the Processor Element (PE) or the Multiple Logical Processor (MLP). The MLP considers each hardware operator to be a micro-program or task, and pipelines these tasks with a hardware operating system based on the Master Control Program (MCP) logic used in all Burroughs B 5000/B 6000/B 7000 and A Series systems. The pipeline contains 3 logical processors, which can execute 3 tasks concurrently. The MLP consists of the major modules described below.

1. Program Controller (PC)

The PC examines the object code stream, determines the tasks necessary to execute the object code, and establishes the requirements (for example, parameters) and priorities of the tasks. The PC then forwards the tasks to the Task Controller.

2. Task Controller (TC)

The TC allocates and controls system resources to accomplish tasks forwarded by the PC. The TC maintains the status of all ready and waiting tasks, selects the next task to be executed, suspends tasks when necessary, and synchronizes the execution of tasks. The TC also assigns Top-of-Stack register pairs to tasks which perform operations on data.

3. Data Path (DP)

The DP stores the primary data items for all the operators in progress, and performs the logical and arithmetic operations on these data items. The DP contains an array of 32 1-word registers to store data (the Top-of-Stack register pairs), and a register mapping device to record the register assignments made by the TC. Utility and Special Purpose registers are located in the DP as well. The DP also contains the arithmetic/logic unit (ALU), which performs arithmetic and logical operations on data in the registers.

4. Address and State Unit (ASU)

The ASU converts the relative memory addresses calculated at compilation time to absolute memory addresses assigned at execution time.

5. Stored Logic Controller (SLC)

The SLC is programmable microcode memory which contains the operator algorithms used to control the execution of the other parts of the processor. When the SLC receives an operator and step number from the TC, it locates the algorithm for that operator, and controls the execution of the requested step.



A SERIES AND B 5000/B 6000/B 7000 CONCEPTS  
A 9 HARDWARE OVERVIEW

E-MODE OPERATOR  
FROM MS (MEMORY)

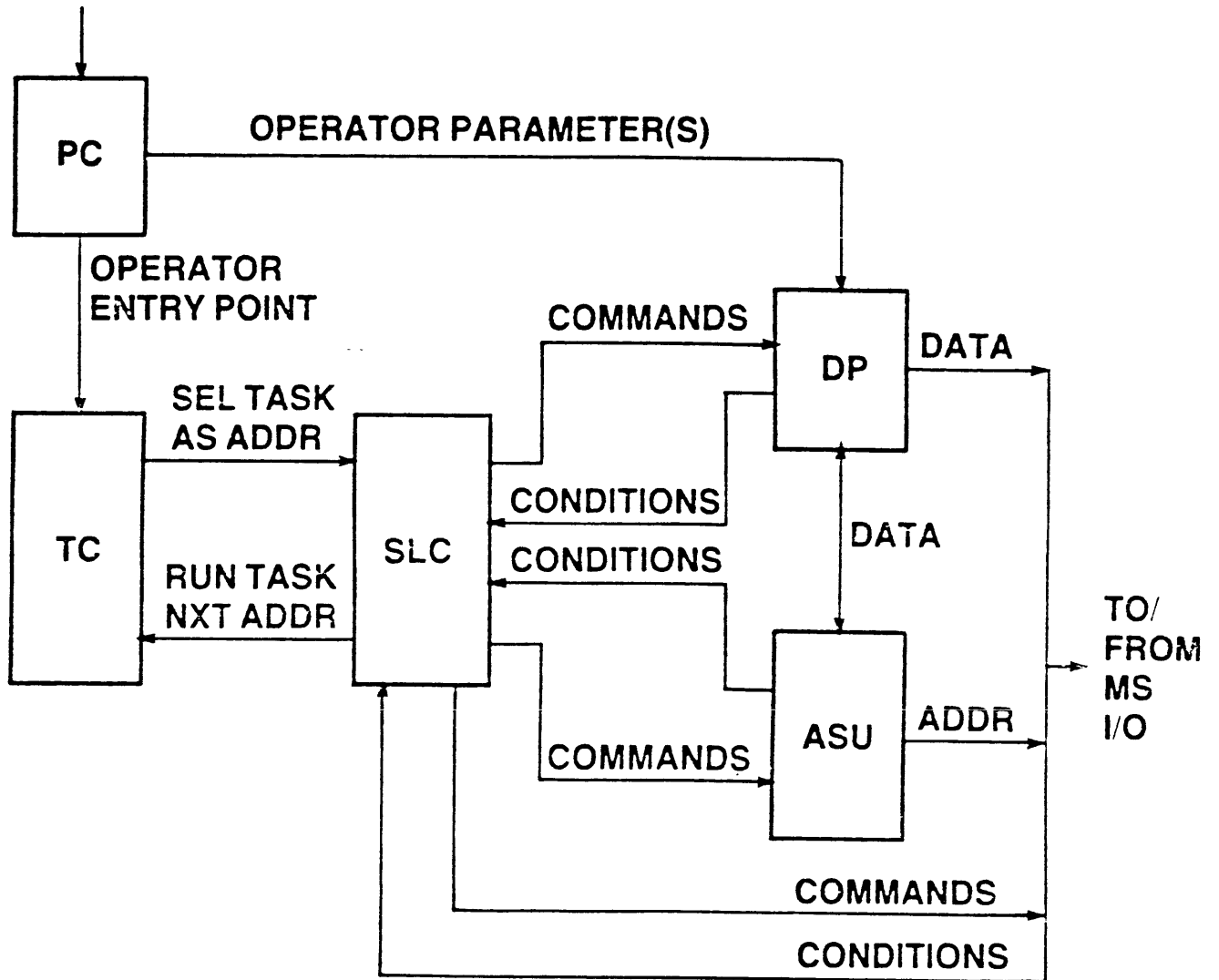


Figure A9-2 Processor Block Diagram

## **A SERIES AND B 5000/B 6000/B 7000 CONCEPTS A 9 HARDWARE OVERVIEW**

### ***Memory Subsystem***

The A 9 memory subsystem is capable of addressing 1, 2, 3, or 4 million words (6, 12, 18, or 24 million bytes) of memory in a single cabinet. A memory base can contain 1 or 2 million words; a second memory base is required in a 3 or 4 million word system.

Under MCP, the A 9 can address 1 million words of memory at a time. A 9 systems with more than 1 million words of memory are configured via ODT commands into Address Spaces (ASN = Address Space Number) and Environment Components (EC). Each Address Space consists of a shared component, which is common to all the Address Spaces, and a local component, which is specific to that Address Space. The shared plus the local components of an Address Space may occupy a maximum total of 1 million words. Each program in the mix will be assigned to an Address Space, which will be accessed when that program has control of the processor.

Under MCP/AS, the A 9 can access all of its memory at any time. Address Spaces are not used with MCP/AS.

The memory subsystem interfaces to the A 9 processor through the Memory Controller (MC).

The MC also controls high speed cache memory, which contains redundant copies of recently addressed blocks in memory. If the required block is present in the cache, memory read requests are satisfied by reading from the cache, rather than from main memory.

### ***Maintenance Subsystem***

The maintenance subsystem includes:

- 2 - Ergonomic Work Stations (EWS)
- 1 - 5 1/4" removable diskette
- 2 - 5 1/4" Winchester fixed disks
- 1 - Maintenance Interface Processor (MIP)

The Ergonomic Work Stations are ET 2000s which function as both the system ODTs, and as the System Control Processor (SCP). The SCP can initialize the system, display the A 9 system state, or cause execution of maintenance and diagnostic programs.

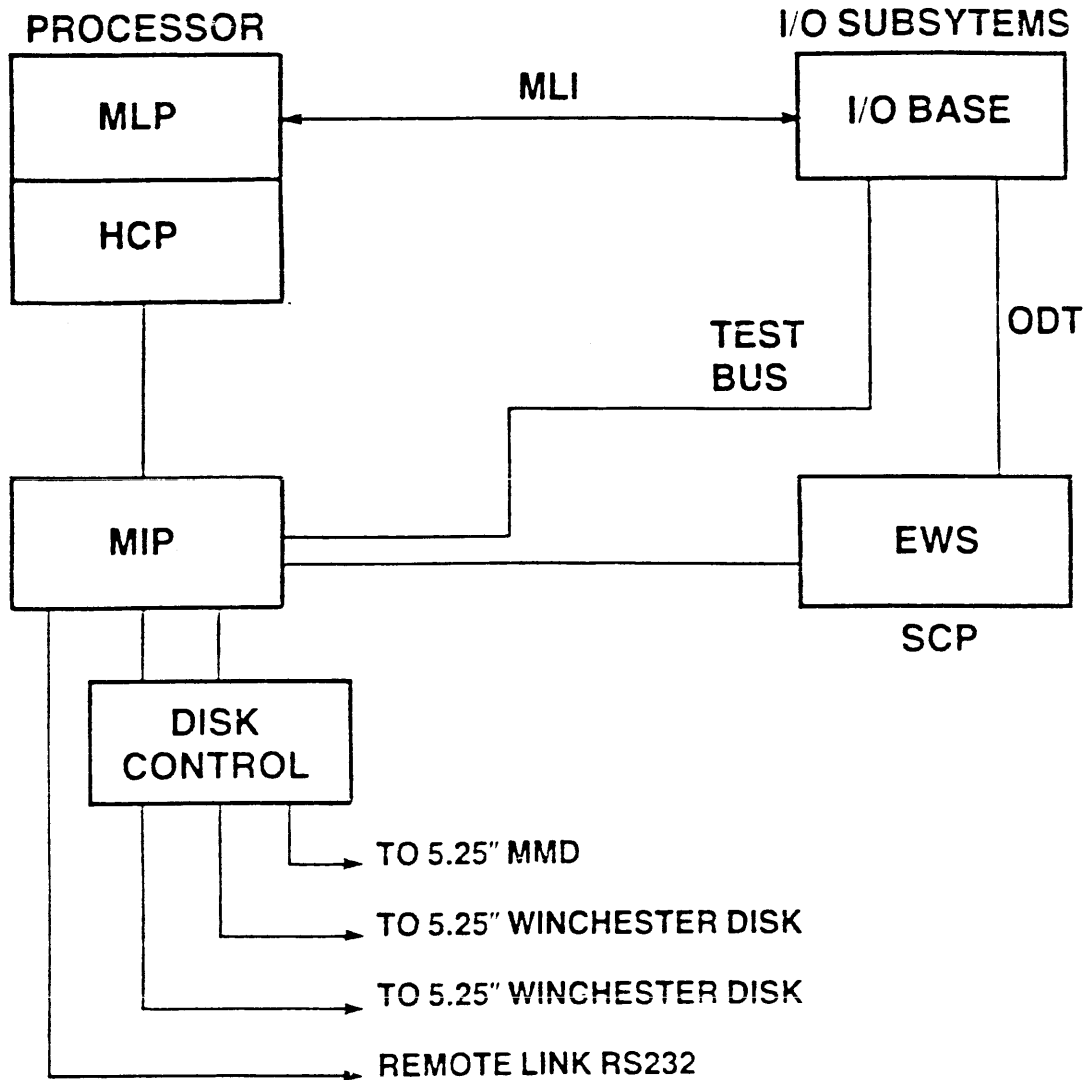
The Maintenance Interface Processor (MIP) contains built-in maintenance subsystem program firmware, and a microprocessor. The MIP can execute maintenance programs that test and control all the other A 9 hardware resources.

The disks store maintenance and diagnostic programs.

An RS232 communications link via the MIP allows testing and diagnostic programs to be executed by Burroughs support personnel located at remote sites. The remote link can be disabled with a key switch on the A 9 cabinet.

The maintenance subsystem interfaces to the A 9 central system through the Host Control Port (HCP).

**A SERIES AND B 5000/B 6000/B 7000 CONCEPTS  
A 9 HARDWARE OVERVIEW**



**MLP = MULTIPLE LOGICAL PROCESSOR**  
**HCP = HOST CONSOLE PORT**  
**MIP = MAINTENANCE INTERFACE PROCESSOR**  
**MLI = MESSAGE LEVEL INTERFACE**  
**SCP = SYSTEM CONTROL PROCESSOR**  
**EWS = ERGONOMIC WORK STATION (ET 2000)**

Figure A9-3 System Control Processor and Maintenance Subsystem

# A SERIES AND B 5000/B 6000/B 7000 CONCEPTS A 9 HARDWARE OVERVIEW

## Input/Output and Data Comm Subsystem

All A Series systems, as well as B 5900s, B 6900s, and B 7900s use Burroughs Universal I/O for both peripheral and data communications management.

Each peripheral subsystem is connected to a Data Link Processor (DLP) which is designed to control that specific type of peripheral. Some peripherals (e.g. image printers) can have only 1 device attached to each DLP, but other peripherals (e.g. disk packs, tapes) can have multiple devices attached to a single DLP through an exchange and/or controller. The data communications subsystem includes two special types of DLPs: Network Support Processors (NSPs), and Line Support Processors (LSPs).

The DLPs are grouped into I/O-Data Comm (IODC) base modules. Each IODC base can hold up to 8 DLPs, depending on the number of circuit boards in the various types of DLPs used. The A 9 system may include 1 to 3 IODC cabinets, depending on the number of IODC base modules required in the configuration.

IODC base modules are connected by Message Level Interface (MLI) cables to the Message Level Interface Processor (MLIP).

The MLIP is an I/O processor providing the system interface to the Universal I/O subsystem. When an I/O operation is to be performed, the Program Controller generates an MLIP operator to be executed. Other processor operators continue to be executed concurrently with the MLIP operator, so that instruction execution continues while an I/O is being performed.

Data to be output to a device is transferred from the central system to the MLIP, then through the MLI to the IODC base, to the DLP, and finally to the designated device. Data input from peripherals follows the reverse route.

A SERIES AND B 5000/B 6000/B 7000 CONCEPTS  
A 9 HARDWARE OVERVIEW

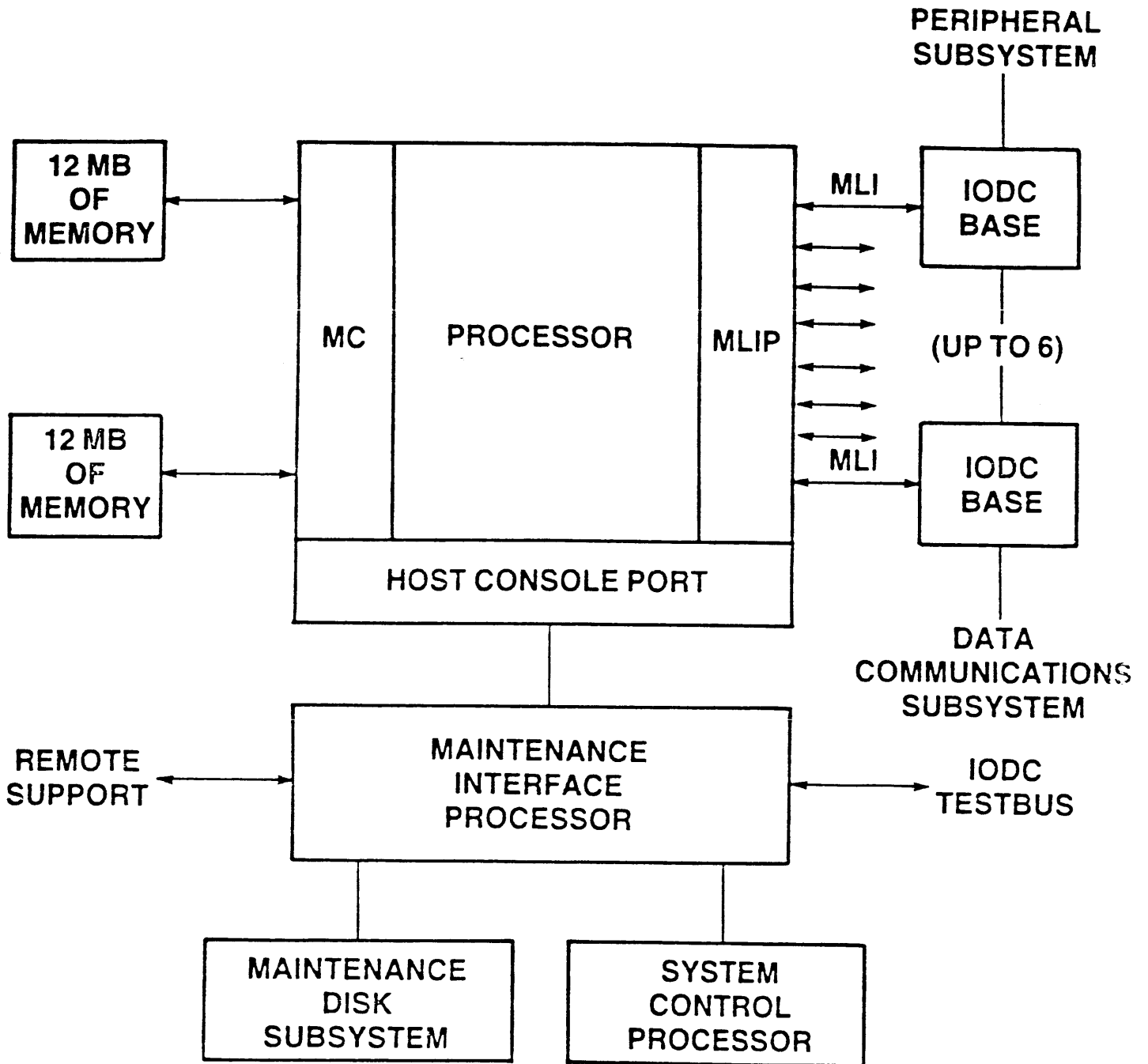


Figure A9-4 System Block Diagram

**A SERIES AND B 5000/B 6000/B 7000 CONCEPTS  
A 9 HARDWARE OVERVIEW**

**Practice**

Match the A 9 system elements on the left with the descriptions on the right.

- |                                            |                                                                           |
|--------------------------------------------|---------------------------------------------------------------------------|
| _____ 1. Multiple Logical Processor        | a. Contains the Top-of-Stack register pairs.                              |
| _____ 2. Data Link Processor               | b. Functions as an ODT, SCP, or maintenance display.                      |
| _____ 3. Maintenance Interface Processor   | c. Is hardware designed to control a specific type of peripheral device.  |
| _____ 4. Message Level Interface Processor | d. Stores the operator microcode.                                         |
| _____ 5. Data Path                         | e. Contains 3 logical processors, to pipeline hardware operator tasks.    |
| _____ 6. Stored Logic Control              | f. Determines the tasks required to execute the object code.              |
| _____ 7. Ergonomic Work Station            | g. Executes maintenance programs initiated on-site or from a remote site. |
| _____ 8. Program Controller                | h. Interfaces the A 9 processor to the I/O subsystem.                     |

**SECTION A 10**  
**A 10 HARDWARE OVERVIEW**

# **A SERIES AND B 5000/B 6000/B 7000 CONCEPTS A 10 HARDWARE OVERVIEW**

## **Objective**

Identify the major hardware elements that constitute the A 10 system.

## **Purpose**

The A 10 system consists of a series of logical units housed in several cabinets, and 2 Operator Display Terminals. This unit will introduce you to these elements.

## **Resources**

A 10 System Reference Manual Volume 1



# A SERIES AND B 5000/B 6000/B 7000 CONCEPTS

## A 10 HARDWARE OVERVIEW

### Central System Components

The A 10 central system consists of 1 or 2 Multiple Logical Processors, Memory Subsystems, Maintenance Subsystems, and Input/Output and Data Communications Subsystems. Modular power supplies are distributed throughout the A 10 cabinets, with externally accessible power control switches located on the cabinets.

### *Multiple Logical Processor (MLP)*

The A 10 central processor is called the Processor Element (PE) or the Multiple Logical Processor (MLP). The MLP considers each hardware operator to be a micro-program or task, and pipelines these tasks with a hardware operating system based on the Master Control Program (MCP) logic used in all Burroughs B 5000/B 6000/B 7000 and A Series systems. The pipeline contains 3 logical processors, which can execute 3 tasks concurrently. The MLP consists of the major modules described below.

1. Program Controller (PC)

The PC examines the object code stream, determines the tasks necessary to execute the object code, and establishes the requirements (for example, parameters) and priorities of the tasks. The PC then forwards the tasks to the Task Controller.

2. Task Controller (TC)

The TC allocates and controls system resources to accomplish tasks forwarded by the PC. The TC maintains the status of all ready and waiting tasks, selects the next task to be executed, suspends tasks when necessary, and synchronizes the execution of tasks. The TC also assigns Top-of-Stack register pairs to tasks which perform operations on data.

3. Data Path (DP)

The DP stores the primary data items for all the operators in progress, and performs the logical and arithmetic operations on these data items. The DP contains an array of 32 1-word registers to store data (the Top-of-Stack register pairs), and a register mapping device to record the register assignments made by the TC. Utility and Special Purpose registers are located in the DP as well. The DP also contains the arithmetic/logic unit (ALU), which performs arithmetic and logical operations on data in the registers.

4. Address and State Unit (ASU)

The ASU converts the relative memory addresses calculated at compilation time to absolute memory addresses assigned at execution time.

5. Stored Logic Controller (SLC)

The SLC is programmable microcode memory which contains the operator algorithms used to control the execution of the other parts of the processor. When the SLC receives an operator and step number from the TC, it locates the algorithm for that operator, and controls the execution of the requested step.

A SERIES AND B 5000/B 6000/B 7000 CONCEPTS  
A 10 HARDWARE OVERVIEW

E-MODE OPERATOR  
FROM MS (MEMORY)

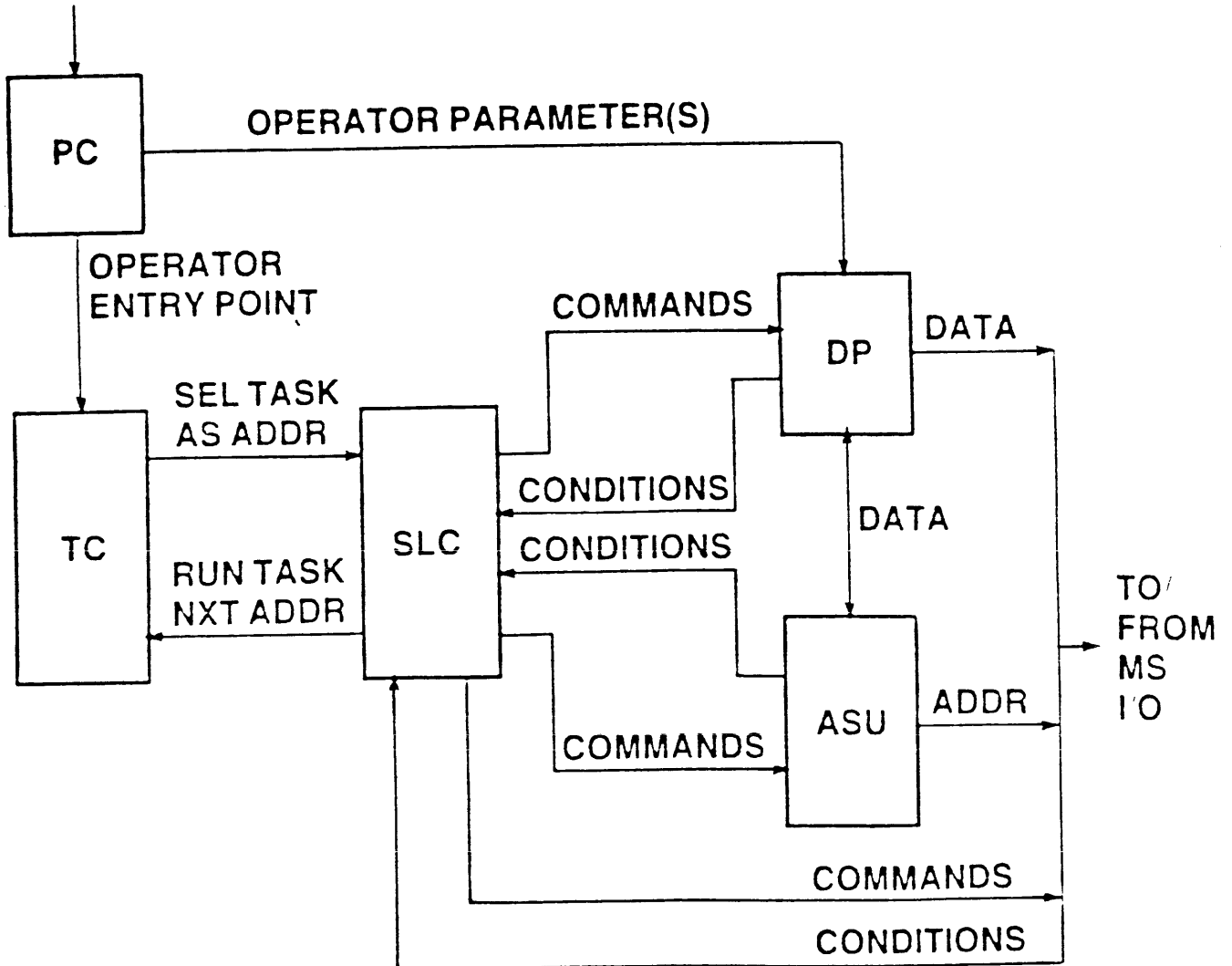


Figure A10-1 Processor Block Diagram

# A SERIES AND B 5000/B 6000/B 7000 CONCEPTS A 10 HARDWARE OVERVIEW

## *A 10 Dual Processor Systems*

A 10H Dual Processor Systems can be operated as a single, joined (monolithic) system, or as 2 independent partitions. A partition is a set of resources capable of running as a system under an MCP. A configuration file controls the division of memory and other resources for a partitioned system.

The Dual Processor Link (DPL) allows the 2 CPUs to communicate when operating in monolithic mode.

## *Memory Subsystem*

The A 10 memory subsystem is capable of addressing 2 to 16 million words (12 to 96 million bytes) of memory in 1 or 2 cabinets.

Under MCP, the A 10 can address 1 million words of memory at a time. A 10 systems with more than 1 million words of memory are configured via ODT commands into Address Spaces (ASN = Address Space Number) and Environment Components (EC). Each Address Space consists of a shared component, which is common to all the Address Spaces, and a local component, which is specific to that Address Space. The shared plus the local components of an Address Space may occupy a maximum total of 1 million words. Each program in the mix will be assigned to an Address Space, which will be accessed when that program has control of the processor.

Under MCP/AS, the A 10 can access all of its memory at any time. Address Spaces are not used with MCP/AS.

The memory subsystem interfaces to the A 10 processor through Dual Port Memory (DPM) Modules, and through the Memory Controller (MC). Configurations with 10 million words (60 million bytes) of memory or more have Dual Port Increments (DPI) to store the additional memory.

The MC also controls 6000 bytes of high speed purgeless cache memory, which contains copies of recently addressed blocks in memory. If the required block is present in the cache, memory read requests are satisfied by reading from the cache, rather than from main memory.

A SERIES AND B 5000/B 6000/B 7000 CONCEPTS  
A 10 HARDWARE OVERVIEW

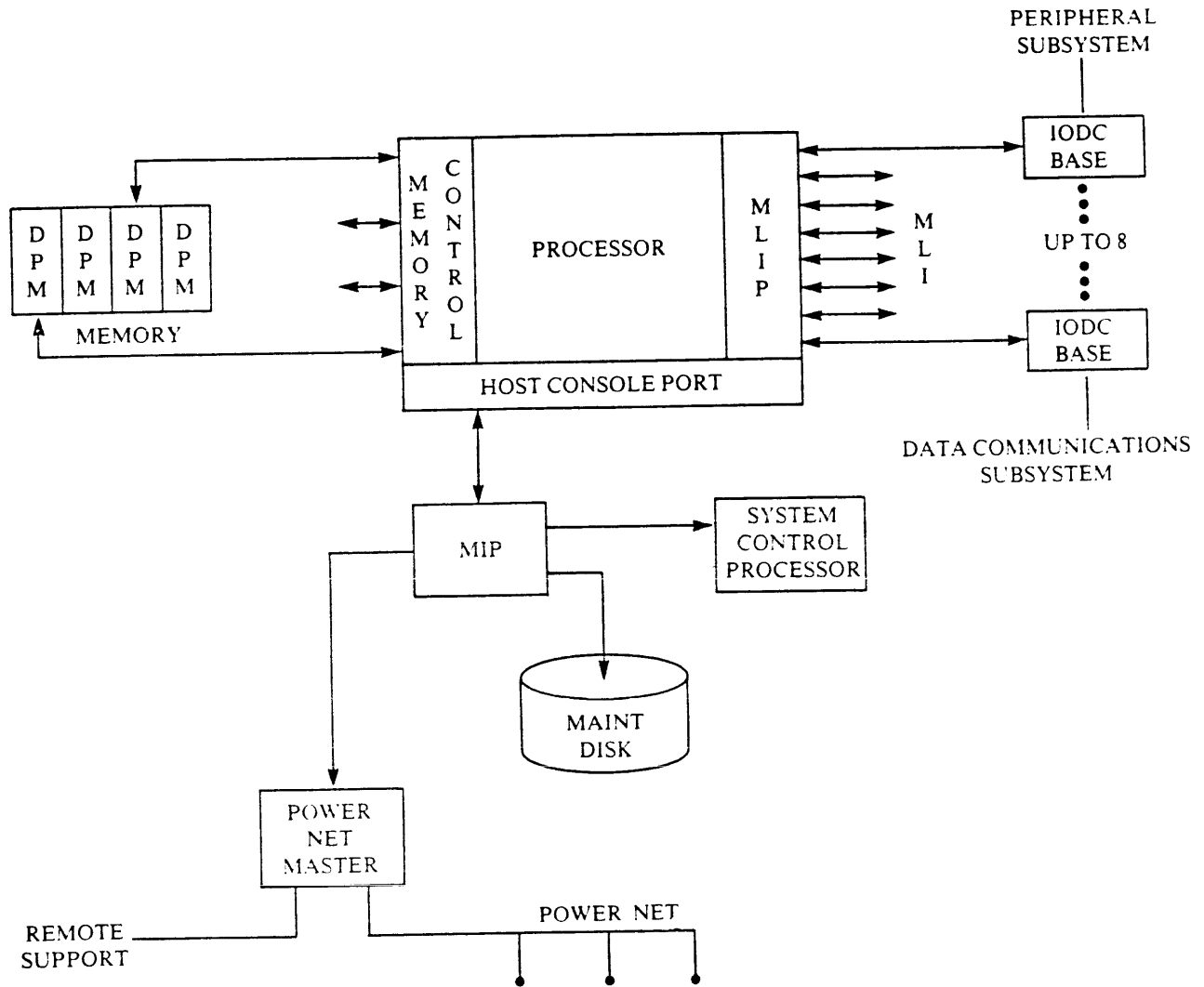


Figure A10-2 A 10 Single Processor Block Diagram

## **A SERIES AND B 5000/B 6000/B 7000 CONCEPTS A 10 HARDWARE OVERVIEW**

### ***Maintenance Subsystem***

The maintenance subsystem includes:

- 2 - Ergonomic Work Stations (EWS)
- 1 - 5 1/4" removable diskette
- 2 - 5 1/4" Winchester fixed disks
- 1 - Maintenance Interface Processor (MIP)

The Ergonomic Work Stations are ET 2000s which function as both the system ODTs, and as the System Control Processor (SCP). The SCP can initialize the system, display the A 10 system state, or cause execution of maintenance and diagnostic programs.

The Maintenance Interface Processor (MIP) contains built-in maintenance subsystem program firmware, and a microprocessor. The MIP can execute maintenance programs that test and control all the other A 10 hardware resources.

The disks store maintenance and diagnostic programs.

An RS232 communications link via the MIP allows testing and diagnostic programs to be executed by Burroughs support personnel located at remote sites. The remote link can be disabled with a key switch on the A 10 cabinet.

The maintenance subsystem interfaces to the A 10 central system through the Host Control Port (HCP).

A SERIES AND B 5000/B 6000/B 7000 CONCEPTS  
A 10 HARDWARE OVERVIEW

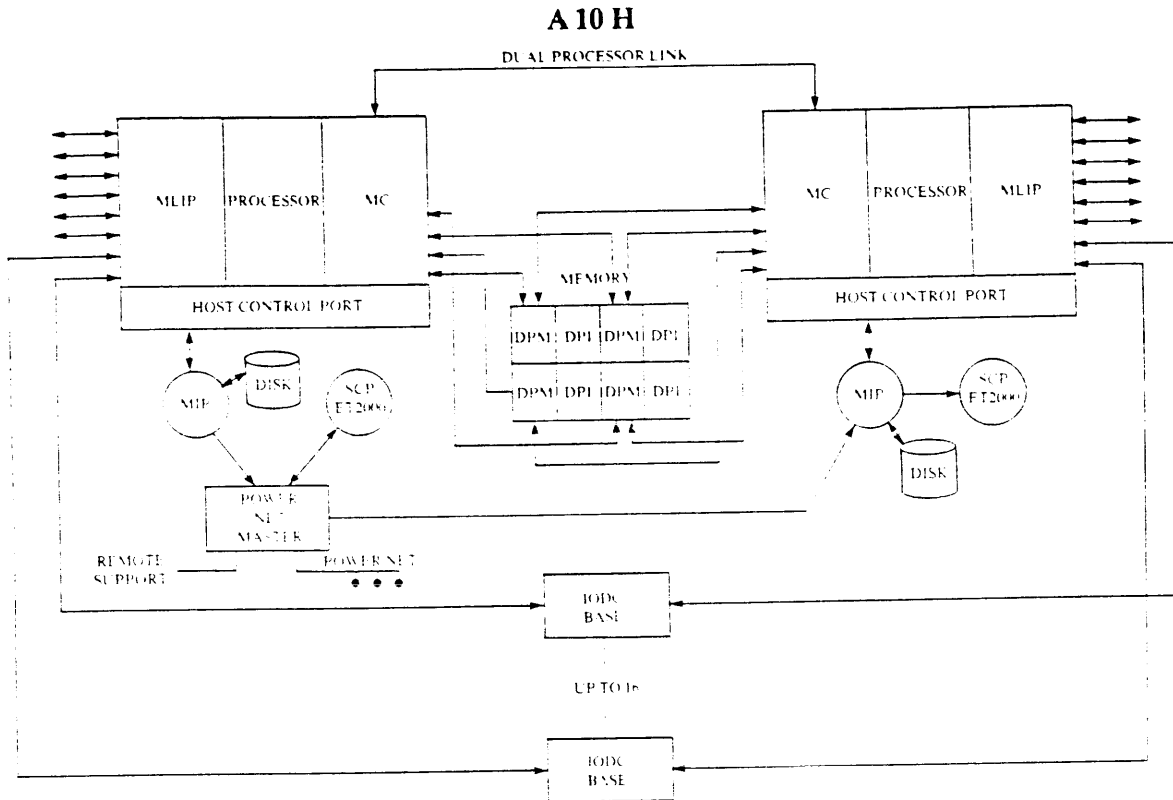


Figure A10-3 A 10 Dual Processor Block Diagram

# A SERIES AND B 5000/B 6000/B 7000 CONCEPTS

## A 10 HARDWARE OVERVIEW

### Input/Output and Data Comm Subsystem

All A Series systems, as well as B 5900s, B 6900s, and B 7900s use Burroughs Universal I/O for both peripheral and data communications management.

Each peripheral subsystem is connected to a Data Link Processor (DLP) which is designed to control that specific type of peripheral. Some peripherals (e.g. image printers) can have only 1 device attached to each DLP, but other peripherals (e.g. disk packs, tapes) can have multiple devices attached to a single DLP through an exchange and/or controller. The data communications subsystem includes two special types of DLPs: Network Support Processors (NSPs), and Line Support Processors (LSPs).

The DLPs are grouped into I/O-Data Comm (IODC) base modules. Each IODC base can hold up to 8 DLPs, depending on the number of circuit boards in the various types of DLPs used. The A 10 system may include 1 to 3 IODC cabinets, depending on the number of IODC base modules required in the configuration.

IODC base modules are connected by Message Level Interface (MLI) cables to the Message Level Interface Processor (MLIP).

The MLIP is an I/O processor providing the system interface to the Universal I/O subsystem. When an I/O operation is to be performed, the Program Controller generates an MLIP operator to be executed. Other processor operators continue to be executed concurrently with the MLIP operator, so that instruction execution continues while an I/O is being performed.

Data to be output to a device is transferred from the central system to the MLIP, then through the MLI to the IODC base, to the DLP, and finally to the designated device. Data input from peripherals follows the reverse route.

**A SERIES AND B 5000/B 6000/B 7000 CONCEPTS  
A 10 HARDWARE OVERVIEW**

**Practice**

Match the A 10 system elements on the left with the descriptions on the right.

- |                                            |                                                                           |
|--------------------------------------------|---------------------------------------------------------------------------|
| _____ 1. Multiple Logical Processor        | a. Contains the Top-of-Stack register pairs.                              |
| _____ 2. Data Link Processor               | b. Functions as an ODT, SCP, or maintenance display.                      |
| _____ 3. Maintenance Interface Processor   | c. Is hardware designed to control a specific type of peripheral device.  |
| _____ 4. Message Level Interface Processor | d. Stores the operator microcode.                                         |
| _____ 5. Data Path                         | e. Contains 3 logical processors, to pipeline hardware operator tasks.    |
| _____ 6. Stored Logic Control              | f. Determines the tasks required to execute the object code.              |
| _____ 7. Ergonomic Work Station            | g. Executes maintenance programs initiated on-site or from a remote site. |
| _____ 8. Program Controller                | h. Interfaces the A 10 processor to the I/O subsystem.                    |



**SECTION A 15**

**A 15 HARDWARE OVERVIEW**

# **A SERIES AND B 5000/B 6000/B 7000 CONCEPTS A 15 HARDWARE OVERVIEW**

## **Objective**

Identify the major hardware elements that constitute the A 15 system.

## **Purpose**

The A 15 system consists of a series of logical units housed in several cabinets, and the system console. This unit will introduce you to these elements.

## **Resources**

A 15 System Capabilities and Features Manual

A 15 Hardware Operational Guide

A 15 Operating Guide

# A SERIES AND B 5000/B 6000/B 7000 CONCEPTS

## A 15 HARDWARE OVERVIEW

### Central System Components

The A 15 central system includes 1 or more Central Processing Modules, I/O Subsystem Modules, Memory Subsystem Modules, System Control Cabinets, and System Maintenance Stations. Additional Central Processing Modules and I/O Subsystem Modules can be incorporated to build the required configuration.

### Central Processor Module (CPM)

The A 15 system includes 1 to 4 CPMs, each of which includes the hardware modules described below.

1. Program Control Unit (PCU)

The PCU examines the object code stream, extracts the operators, and builds the execution string or "pipeline." The PCU also prepares the required data by assigning stack locations, and by requesting the Data Reference Unit (DRU) to read data from memory.

2. Data Reference Unit (DRU)

The DRU, upon command from the PCU, fetches data from memory, and places it in the Central Data Buffer or stack.

3. Memory Access Unit (MAU)

The MAU interfaces the CPM to main memory. It receives the memory address from the PCU for code fetches, from the DRU for data fetches, and from the Write Unit for writes to memory.

4. Write Unit

The Write Unit reduces traffic to main memory by buffering data before it is sent to the MAU. Repeated stores to the same address will be performed as one store to main memory; stores to adjacent addresses will be grouped into one multi-word store.

5. Execution Unit (EU)

The EU actually performs all the arithmetic and logical operations in the system, by executing the pipelines of operators created by the PCU, using the data supplied by the DRU. The PCU and DRU prepare the operators and data, so that the EU can continue executing without being interrupted to do memory accesses.

A SERIES AND B 5000/B 6000/B 7000 CONCEPTS  
 A 15 HARDWARE OVERVIEW

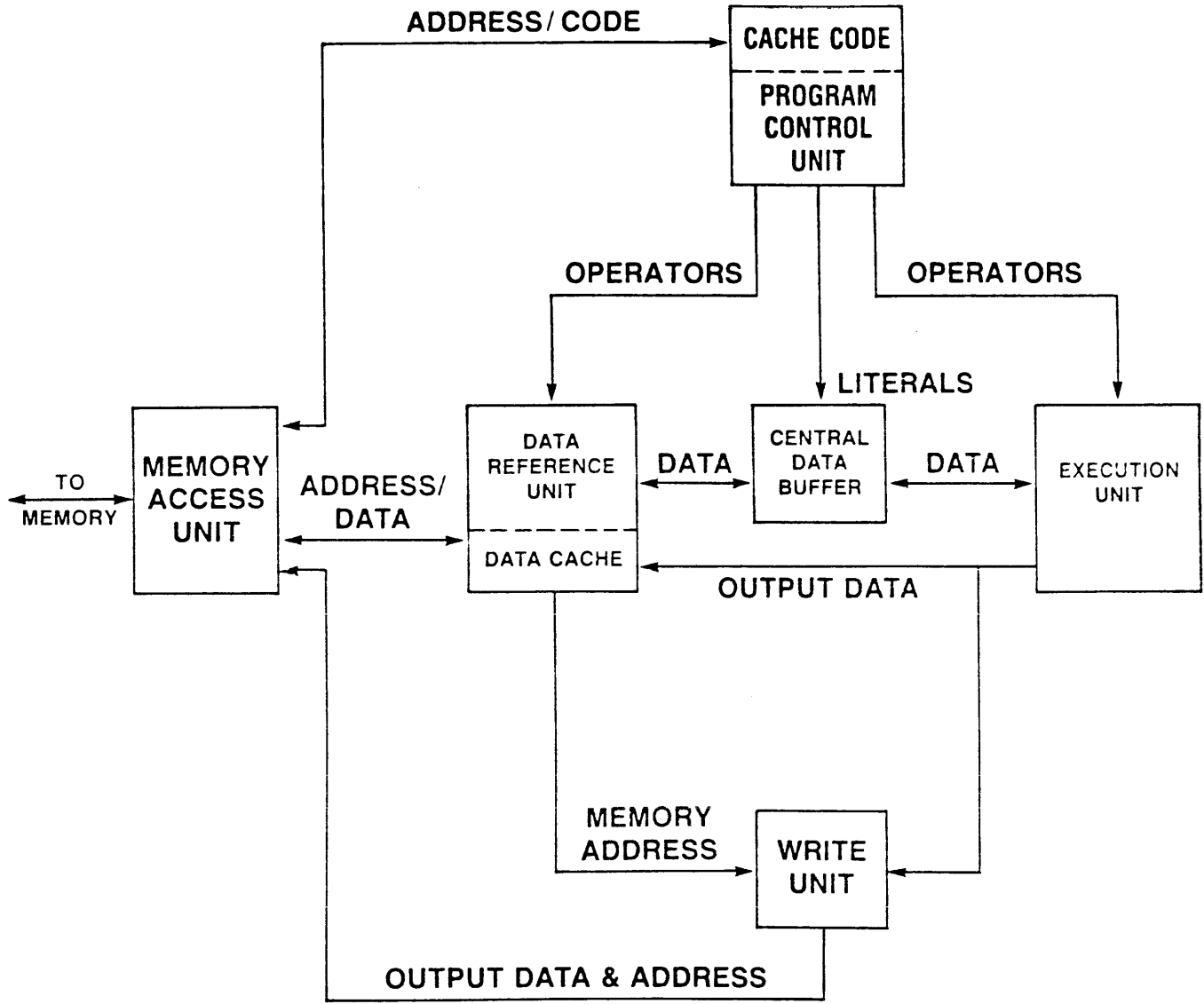


Figure A15-1 CPM Block Diagram

## A SERIES AND B 5000/B 6000/B 7000 CONCEPTS A 15 HARDWARE OVERVIEW

### *Input/Output Subsystem Module (IOSM)*

All A 15 systems have at least 1 IOSM cabinet, to house the Host Data Unit, the System Maintenance Processor, and the IO DC base modules (described under I/O-Data Comm Subsystem below).

1. Host Data Unit (HDU)

Each IOSM cabinet has an HDU, to handle all I/O data transfers between A 15 main memory and the I/O subsystem. Refer to the I/O-Data Comm Subsystem section for further information about the HDU.

2. System Maintenance Processor (SMP)

The System Maintenance Processor is a processor housed in the IOSM. The SMP is intended to operate at all times, although it is not critical for operation or for Halt/Loading of the A 15. While the A 15 is processing other programs, the SMP can analyze memory dumps that were transferred from the mainframe, or send dumps to a Remote Diagnostics Center

### *Memory Subsystem Module (MSM)*

The A 15 memory subsystem can contain a maximum of 32 million words (192 million bytes) of memory. The MSM cabinet holds a Memory Control, and a maximum of 4 Memory Storage Units (MSU), with up to 8 million words of memory each. The Memory Control allows a maximum of 8 requestors (CPMs, APs and HDUs) to access memory.

High speed purgeless cache memory keeps recently used information in memory, so that it can be accessed quickly if needed again. The A 15 has 24,000 bytes of cache memory for object code storage, and 24,000 bytes for data. The A 15 cache does not purge the contents of cache to main memory regularly, as was necessary in previous cache implementations (for example, on the B 7900). This reduces the traffic between memory and the processor, and results in finding the desired data in cache more often.

Under MCP, the A 15 can address 1 million words of memory at a time. A 15 systems with more than 1 million words of memory are configured via ODT commands into Address Spaces (ASN = Address Space Number) and Environment Components (EC). Each Address Space consists of a shared component, which is common to all the Address Spaces, and a local component, which is specific to that Address Space. The shared plus the local components of an Address Space may occupy a maximum total of 1 million words. Each program in the mix will be assigned to an Address Space and its object code to another. When a program has control of the processor, the MCP will access both the shared component and the local component of the appropriate Address Spaces for that program.

Under MCP/AS, the A 15 can access all of its memory at any time. Address Spaces are not used with MCP/AS.

**A SERIES AND B 5000/B 6000/B 7000 CONCEPTS  
A 15 HARDWARE OVERVIEW**

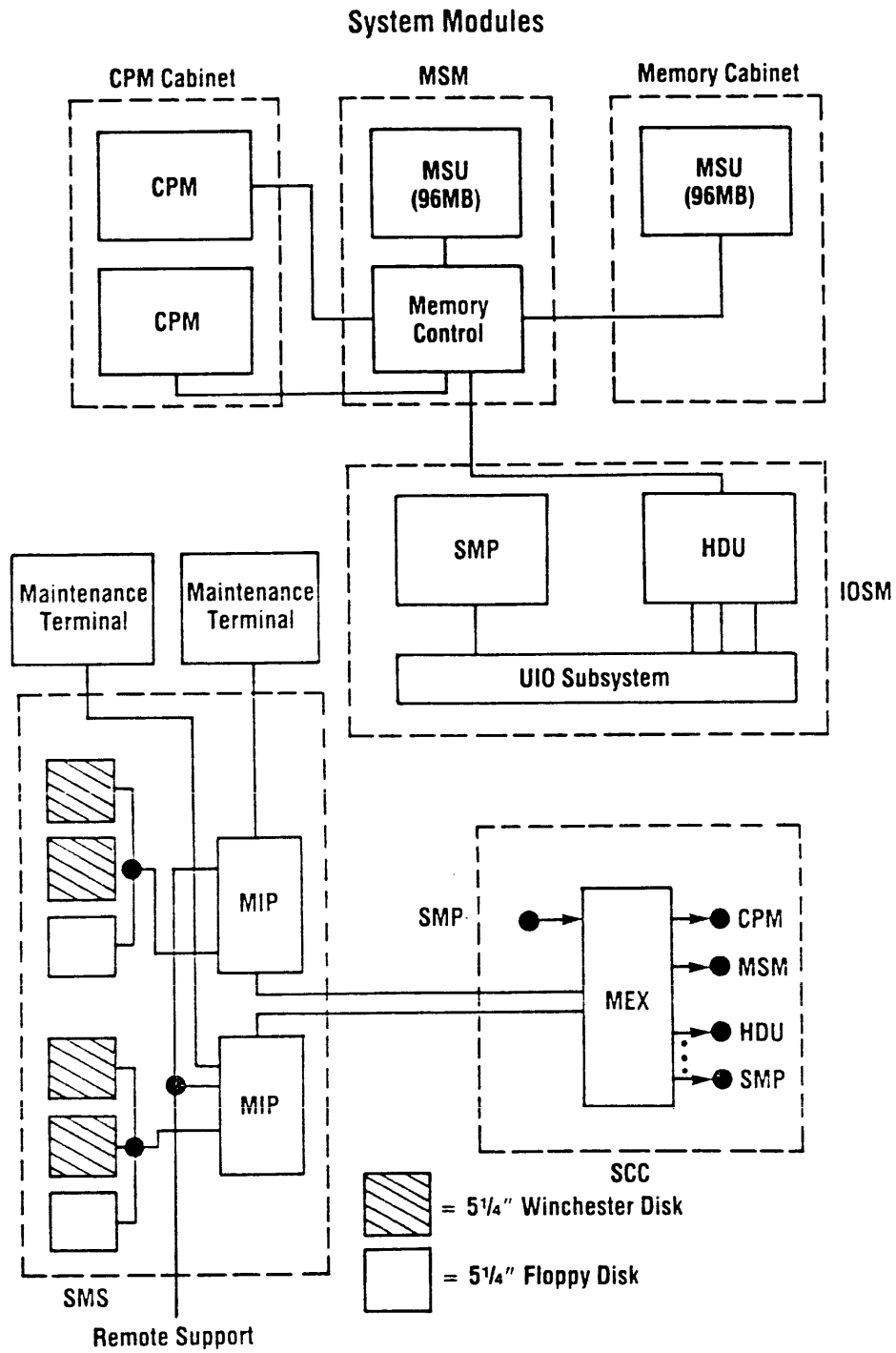


Figure A15-2 System Modules

## **A SERIES AND B 5000/B 6000/B 7000 CONCEPTS A 15 HARDWARE OVERVIEW**

### ***System Control Cabinet (SCC)***

The SCC is a separate cabinet that contains the Central Power Control, the Master System Clock, and the Maintenance Exchange. The Central Power Control allows all mainframe components to be powered up or down from a central location. The Master System Clock supplies clock signals to the main system. The Maintenance Exchange provides the interface between all possible system maintenance processors, and all possible mainframe units under test.

### ***System Maintenance Station (SMS)***

The System Maintenance Station is a separate cabinet that contains the electronics and storage for the ET 2000-based soft console. There are 2 ET 2000s, 1 for use as the System Maintenance Processor (SMP) ODT, and 1 for use as a system ODT. Each ET 2000 has its own Maintenance Interface Processor (MIP), 2 Winchester disk drives, a disk controller, and a floppy disk drive. At least 1 of the ET 2000s must be available during the Halt/Load process.

SYCON (SYstem CONsole) is a software program which executes on an ET 2000/MIP pair to allow configuration, initialization, and status checking for the A 15 and its partitions. The operator loads SYCON into the MIP from the mini-disk drives, and then communicates with SYCON through menu screens on the ET 2000.

In addition, an RS232 communications link can be used to allow A 15 testing and diagnostic programs to be executed by Burroughs personnel located at remote sites. The installation may choose to allow remote access to the SMP, but prevent access to the A 15 mainframe.

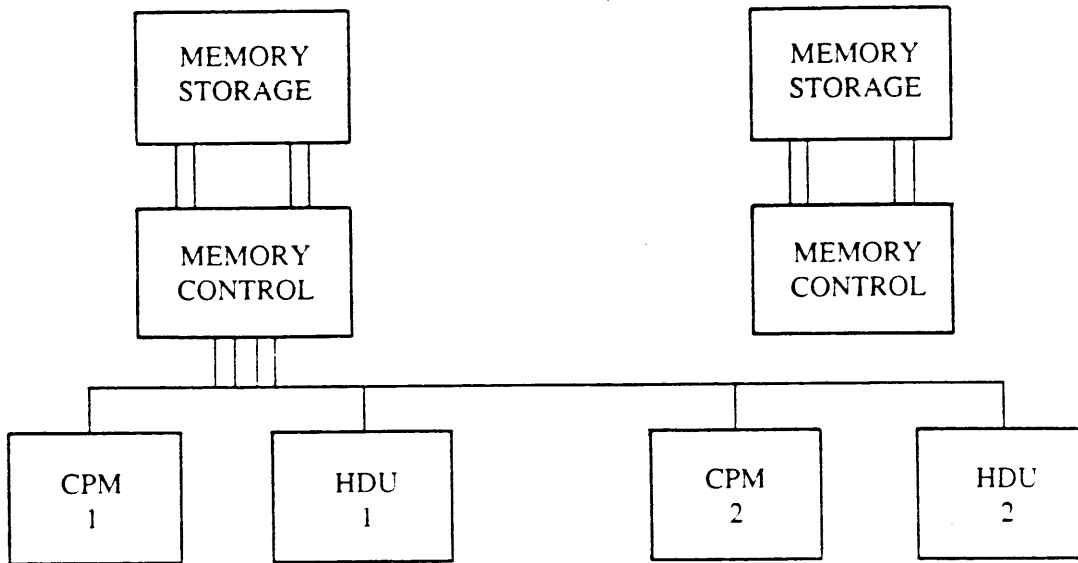
### **Partitions**

Multiple processor A 15s may be configured as a single system, or as separate partitions with at least 1 CPM, HDU, and MSM in each partition. The terms "box" and "component" refer to an individual CPM, HDU, AP, or MSM, so a partition can be defined as a subset of boxes running under a separate MCP. The boxes within a partition communicate with each other through the memory subsystem.

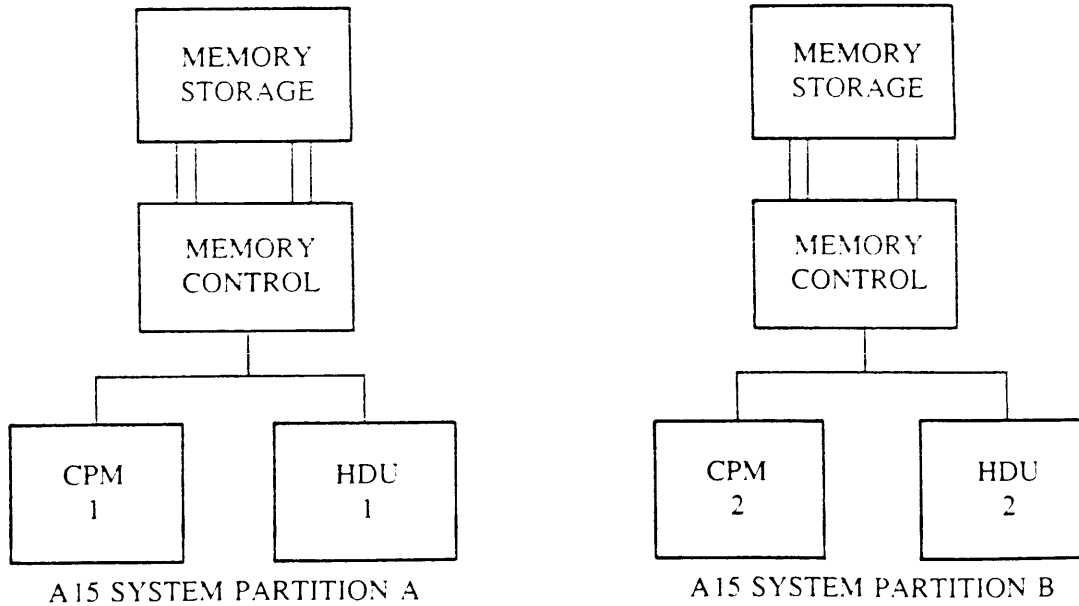
Partitioning creates logically separate systems running under separate MCPs, which allows different workload environments, or both production and test environments, to exist concurrently. Partitions give additional flexibility by enabling a running partition to act as the maintenance processor for hardware modules not in that partition.

The SYCON program (see the Maintenance Hardware and Software section) can be used to establish partitions at Halt/Load time, or a configuration file can be built so that RECONFIGURE commands can be entered at the ODT while the system is running. The ODT commands FREE and ACQUIRE are used to add or delete resources for an existing partition. A PC diagram, which is a file that defines the hardware configuration of the A 15, is also required for partitioning.

### A15 SYSTEM PARTITIONING



A15 SINGLE IMAGE SYSTEM



A15 SYSTEM PARTITION A

A15 SYSTEM PARTITION B

A15 PARTITIONED INTO TWO SYSTEMS

Figure A15-3 Partitions



# **A SERIES AND B 5000/B 6000/B 7000 CONCEPTS A 15 HARDWARE OVERVIEW**

## **Input/Output and Data Comm Subsystem**

All A Series systems, as well as the B 5900, B 6900, and B 7900, use Burroughs Universal I/O for both peripheral and data communications management.

Each peripheral subsystem is connected to a Data Link Processor (DLP) which is designed to control that specific type of peripheral. Some peripherals (e.g. line printers) can have only 1 device attached to each DLP, but other peripherals (e.g. disk packs, tapes) can have multiple devices attached to a single DLP through an exchange and/or controller. The data communications subsystem includes two special types of DLPs: Network Support Processors (NSPs), and Line Support Processors (LSPs).

The DLPs are grouped into I/O-Data Comm (IODC) base modules. Each IODC base module can hold up to 8 DLPs, depending on the number of circuit boards in the various DLPs used. The DLPs and IODC base modules are located in the IOSM cabinet.

IODC base modules are connected by Message Level Interface (MLI) cables to a Host Dependent Port (HDP). The A 15 can have a maximum of 3 HDPs, each controlling 2 MLIs.

HDPs are components of the Host Data Unit (HDU), which handles all I/O data transfers between main memory and the I/O subsystem. HDPs provide the HDU interface to the I/O subsystem; the Memory Bus Control (MBC) provides the HDU interface to main memory. The Queue Manager (QM), which is the third major component of the HDU, maintains the I/O queues, and schedules activity for the HDPs.

Data to be output to a device is transferred from main memory through the MBC to a HDP, then through a MLI to the IODC base, to the DLP, and finally to the designated device. Data input from peripherals follows the reverse route.

**A SERIES AND B 5000/B 6000/B 7000 CONCEPTS  
A 15 HARDWARE OVERVIEW**

**Practice**

Match the A 15 system elements on the left with the descriptions on the right.

- |                               |                                                                                 |
|-------------------------------|---------------------------------------------------------------------------------|
| _____ 1. Data Link Processor  | a. High speed memory that contains the most recently used data and object code. |
| _____ 2. Cache                | b. Buffers data to reduce the number of writes to memory.                       |
| _____ 3. SYCON                | c. Performs the logical and arithmetic operators in the processor.              |
| _____ 4. Execution Unit       | d. Is hardware designed to control a specific type of peripheral device.        |
| _____ 5. Program Control Unit | e. Functions as a maintenance processor.                                        |
| _____ 6. Write Unit           | f. Manages all data transfers between main memory and the I/O subsystem.        |
| _____ 7. Host Data Unit       | g. Is a subset of A 15 modules which has been configured as a separate system.  |
| _____ 8. Partition            | h. Is a program to initialize and configure the A 15 from the console.          |
| _____ 9. SMP                  | i. Builds the operator pipeline.                                                |

**SECTION B 5900**

**B 5900 HARDWARE OVERVIEW**

# **A SERIES AND B 5000/B 6000/B 7000 CONCEPTS B 5900 HARDWARE OVERVIEW**

## **Objective**

Identify the major hardware elements that constitute the B 5900 system.

## **Purpose**

The B 5900 system consists of a series of micro-processors housed in 1 or 2 cabinets. In addition, there is an operator's console that houses the Operator Display Terminal and mini-disk drives. This unit will introduce you to these elements.

## **Resources**

B 5900 System Reference Manual

# A SERIES AND B 5000/B 6000/B 7000 CONCEPTS B 5900 HARDWARE OVERVIEW

## Central System Components

The B 5900 central system consists of the Data Processor, the Memory Subsystem, the Maintenance Subsystem, and the Input/Output and Data Communications Subsystem. The power supply for the processor and local memory is located in the mainframe cabinet. Power for the console is located in the console.

### *Data Processor (DP)*

The Data Processor module has the logic to perform all arithmetic and logical functions, as well as logic to shift data words and extract bits from them. The Arithmetic Logic Units (ALUs) are located in this module and contain the 16 processor registers.

The DP has several components.

1. Program Controller (PC) Module

The Program Controller has the main function of fetching code words from local memory and decoding these words into the various program operators and their parameters. For each program operator, the PC addresses a location in a microcode memory contained on another processor module. The microcode word read from the memory then controls the hardware to cause execution of the program operator.

2. Stored Logic Control (SLC) Module

The Stored Logic Control is the module containing the microcode memory. This memory is addressed by the program operator from the PC module. The addressed micro-words are used to control the other modules in the processor. The SLC also looks at conditions being returned by the other modules, and uses these conditions to determine which location of the microcode memory to address for the next micro-word. This continues until the program operator is executed completely. Then the next program operator from the PC module addresses the microcode memory and the process is repeated.

The SLC is also called the Micro Master Control Processor (MMCP).

3. Command (C) Bus

Microcode words are placed on the Command Bus by the SLC module. These microcode words are the instructions to the other modules in the B 5900 processor.

4. Memory (M) Bus

Data flows between the memory modules and the B 5900 processor on the bi-directional Memory Bus.

## **A SERIES AND B 5000/B 6000/B 7000 CONCEPTS B 5900 HARDWARE OVERVIEW**

### **5. Interrupt/Timer Module**

The Interrupt/Timer Module has the timers required by the processor, as well as logic which detects interrupts that might occur during processor operations. These interrupts can then be handled by the software or hardware as necessary. If interrupts that cause the processor to halt occur, logic called snakes may be displayed on the maintenance ODT for the FE to use when isolating faults.

### ***Memory Subsystem***

The Memory Control Module contains the timing logic and control logic required to perform reads and writes to local memory. It receives its commands from the C Bus, and uses the M Bus both to receive data to be written to memory and to send data to the other processor modules.

The local memory is contained in the processor cabinet. It is driven by the Memory Control Module.

Global Memory can be used to connect a maximum of 4 B 5900 processors together.

### ***Maintenance Subsystem***

The Maintenance Subsystem is used to perform maintenance functions on the processor modules. It receives its commands via the C Bus or the Host Console Port (HCP), which connects the Maintenance processor and the console.

The B 5900 has 1 or 2 MTS-2 terminals that are used as both ODTs and maintenance displays. The console also houses 1 or 2 Industry Compatible Minidisk (ICMD) drives, to provide storage for system initialization and diagnostic programs.

The Maintenance Interface Processor (MIP) interfaces the MTS-2 maintenance terminals with the other modules of the maintenance subsystem.

**A SERIES AND B 5000/B 6000/B 7000 CONCEPTS  
B 5900 HARDWARE OVERVIEW**

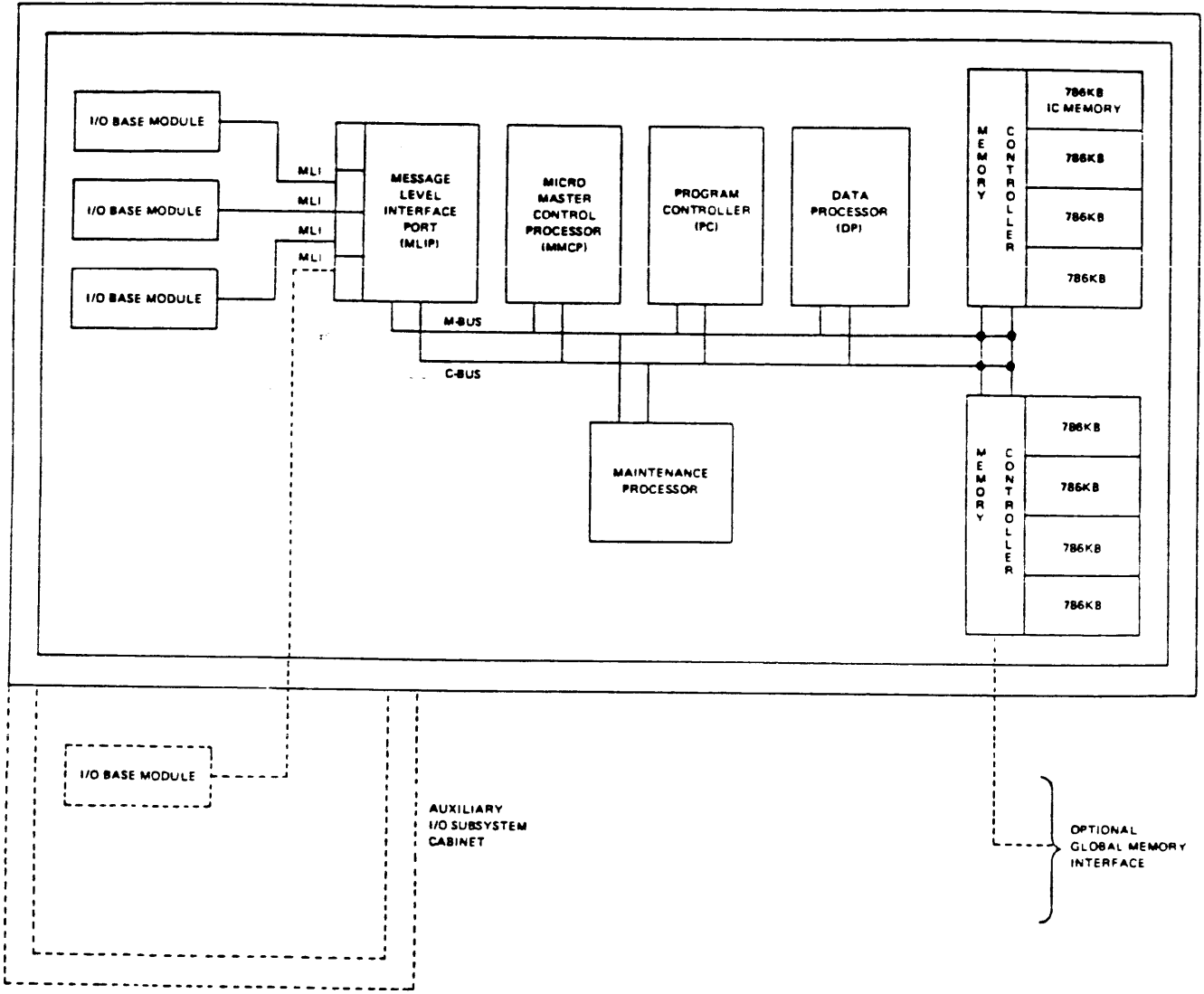


Figure B5900-1 B 5900 Functional Block Diagram

# A SERIES AND B 5000/B 6000/B 7000 CONCEPTS B 5900 HARDWARE OVERVIEW

## Input/Output and Data Comm Subsystem

B 5900s, as well as B 6900s, and B 7900s, and all A Series systems, use Burroughs Universal I/O for both peripheral and data communications management.

Each peripheral subsystem is connected to a Data Link Processor (DLP) which is designed to control that specific type of peripheral. Some peripherals (e.g. image printers) can have only 1 device attached to each DLP, but other peripherals (e.g. disk packs, tapes) can have multiple devices attached to a single DLP through an exchange and/or controller. The data communications subsystem includes two special types of DLPs: Network Support Processors (NSPs), and Line Support Processors (LSPs).

The DLPs are grouped into I/O-Data Comm (IODC) base modules. Each IODC base can hold up to 8 DLPs, depending on the number of circuit boards in the various types of DLPs used.

IODC base modules are connected by Message Level Interface (MLI) cables to the Message Level Interface Processor (MLIP).

The MLIP is an I/O processor providing the system interface to the Universal I/O subsystem. When an I/O operation is to be performed, the Program Controller generates an MLIP operator to be executed. Other processor operators continue to be executed concurrently with the MLIP operator, so that instruction execution continues while an I/O is being performed.

Data to be output to a device is transferred from the central system to the MLIP, then through the MLI to the IODC base, to the DLP, and finally to the designated device. Data input from peripherals follows the reverse route.



**A SERIES AND B 5000/B 6000/B 7000 CONCEPTS  
B 5900 HARDWARE OVERVIEW**

**Practice**

Match the B 5900 system elements on the left with the descriptions on the right.

- |                                            |                                                                                            |
|--------------------------------------------|--------------------------------------------------------------------------------------------|
| _____ 1. Program Controller                | a. Develops the timing required to read and write memory.                                  |
| _____ 2. Stored Logic Control              | b. Extracts individual operators and their parameters from code words fetched from memory. |
| _____ 3. Data Processor                    | c. Is hardware designed to control a specific type of peripheral device.                   |
| _____ 4. Message Level Interface Processor | d. Stores the operator microcode.                                                          |
| _____ 5. Memory Control Module             | e. Interfaces the B 5900 processor to the I/O subsystem.                                   |
| _____ 6. Maintenance Processor             | f. Serves as an interface between the console and the Maintenance Processor.               |
| _____ 7. Maintenance Interface Processor   | g. Contains the logic necessary to perform arithmetic functions and shift data.            |
| _____ 8. Data Link Processor               | h. Performs maintenance functions against processor modules.                               |

**A SERIES AND B 5000/B 6000/B 7000 CONCEPTS  
B 5900 HARDWARE OVERVIEW**

This page left blank for formatting.

***SECTION B 6900***

***B 6900 HARDWARE OVERVIEW***

# **A SERIES AND B 5000/B 6000/B 7000 CONCEPTS B 6900 HARDWARE OVERVIEW**

## **Objective**

Identify the major hardware elements that constitute the B 6900 system.

## **Purpose**

The B 6900 system consists of a series of logical units housed in 5 to 7 cabinets. In addition, there is an operator's console that houses the 2 Operator Display Terminals. This unit will introduce you to these elements.

## **Resources**

B 6900 System Reference Manual

# A SERIES AND B 5000/B 6000/B 7000 CONCEPTS B 6900 HARDWARE OVERVIEW

## Central System Components

The B 6900 central system consists of the Data Processor, the Memory Subsystem, the Maintenance Subsystem, and the Input/Output and Data Communications Subsystem. The power supply for the processor and local memory is located in the mainframe cabinet.

### *Data Processor (DP)*

The Data Processor module has the logic to perform all arithmetic and logical functions, as well as logic to shift data words and extract bits from them. The DP contains logic circuits to sense interrupts from other modules, and to notify the MCP to handle the interrupt.

The B 6900 system also uses look-ahead logic in the DP. This feature fetches words of program code before the DP is ready to execute the code. This virtually eliminates the need for halting a program to fetch words of program code. The memory accesses that are performed by the look-ahead logic are independent of other memory cycles performed for the DP, and do not cause delays in obtaining data for normal DP functions.

The B 6900 DP makes extensive use of Random Access Memory (RAM) and Programmable Read Only Memory (PROM) integrated circuit components.

### *Memory Subsystem*

The local memory is stored in modules of 128K or 256K words (1K = 1024 words). The B 6900 may have a maximum of 1 million words of local memory.

Global Memory can be used to connect a maximum of 4 B 6900 processors together. The B 6900 can access a total of 1 million words local and global memory at any time

The Memory Control Module operates a memory interface exchange that allows 2 system requestors to access the memory modules. The first requestor is the look-ahead logic for object code fetches. The second is the DP or MLIP, which share a requestor path to the memory control.

In addition to controlling the interface paths through the memory exchange, the memory control module also performs memory retries, and memory read data error corrections.

A SERIES AND B 5000/B 6000/B 7000 CONCEPTS  
B 6900 HARDWARE OVERVIEW

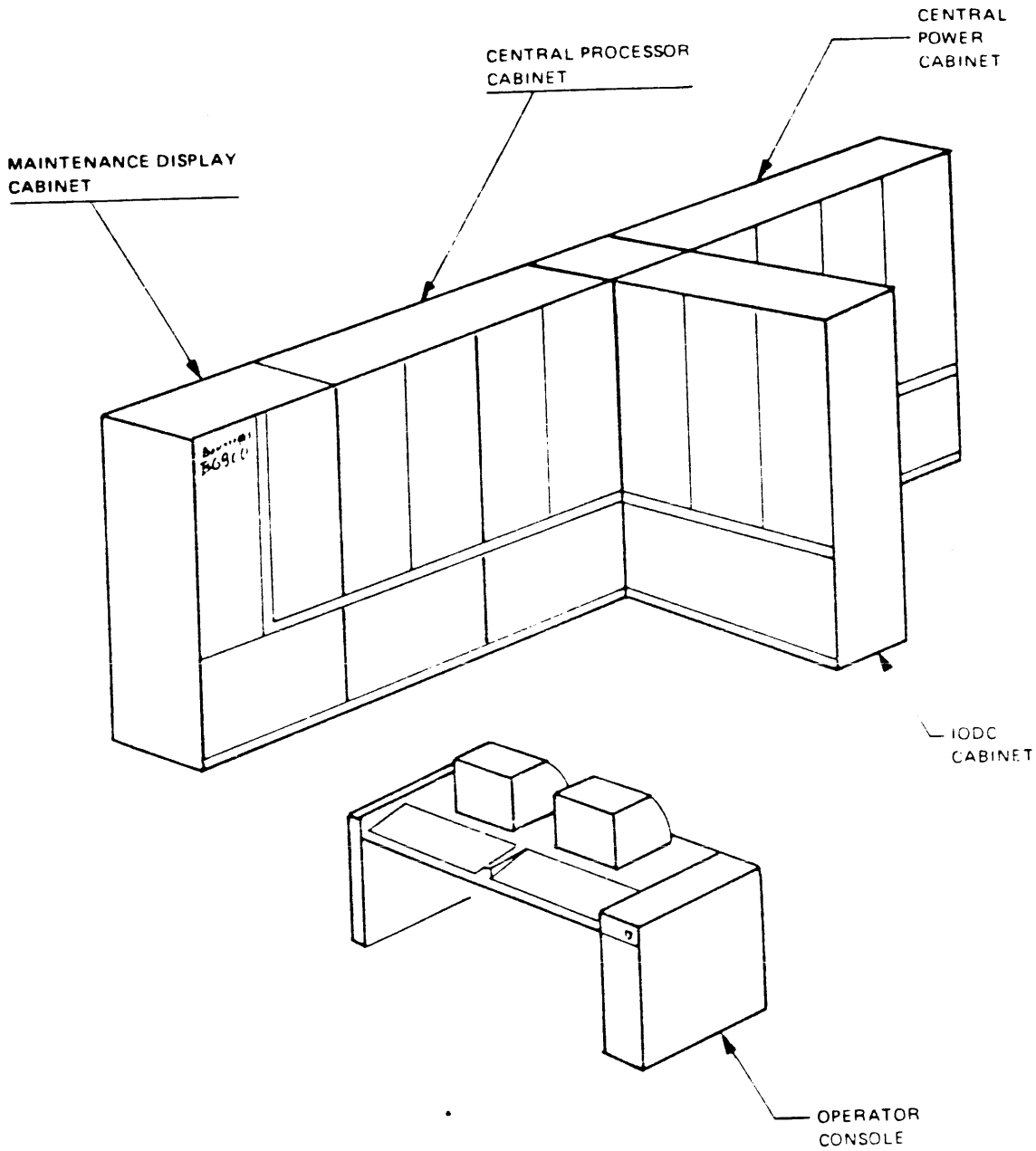


Figure B6900-1 B 6900 with Maintenance Display Cabinet

## **A SERIES AND B 5000/B 6000/B 7000 CONCEPTS B 6900 HARDWARE OVERVIEW**

### *Maintenance Subsystem*

The Maintenance Subsystem is used to perform maintenance functions on the processor modules, and to initialize the B 6900.

The Maintenance Diagnostic Processor (MDP) is sometimes called the Burroughs Diagnostic Unit (BDU). It is actually a B 81 built into the B 6900 cabinets.

The (MDP) is contained in a separate cabinet for B 6900s with low serial numbers. B 6900s with higher serial numbers do not contain an MDP cabinet, but the functions of the MDP cabinet logic circuits are distributed to other cabinets and modules.

The MDP includes a series of switches and buttons used for maintenance purposes, and a diskette drive which loads the firmware into the maintenance processor during system initialization.

The Operator Display Terminals (ODT) allow the operator to enter commands during system initialization and testing.

### **Input/Output and Data Comm Subsystem**

B 6900s, as well as B 5900s, and B 7900s, and all A Series systems, use Burroughs Universal I/O for both peripheral and data communications management.

Each peripheral subsystem is connected to a Data Link Processor (DLP) which is designed to control that specific type of peripheral. Some peripherals (e.g. image printers) can have only 1 device attached to each DLP, but other peripherals (e.g. disk packs, tapes) can have multiple devices attached to a single DLP through an exchange and/or controller. The data communications subsystem includes two special types of DLPs: Network Support Processors (NSPs), and Line Support Processors (LSPs).

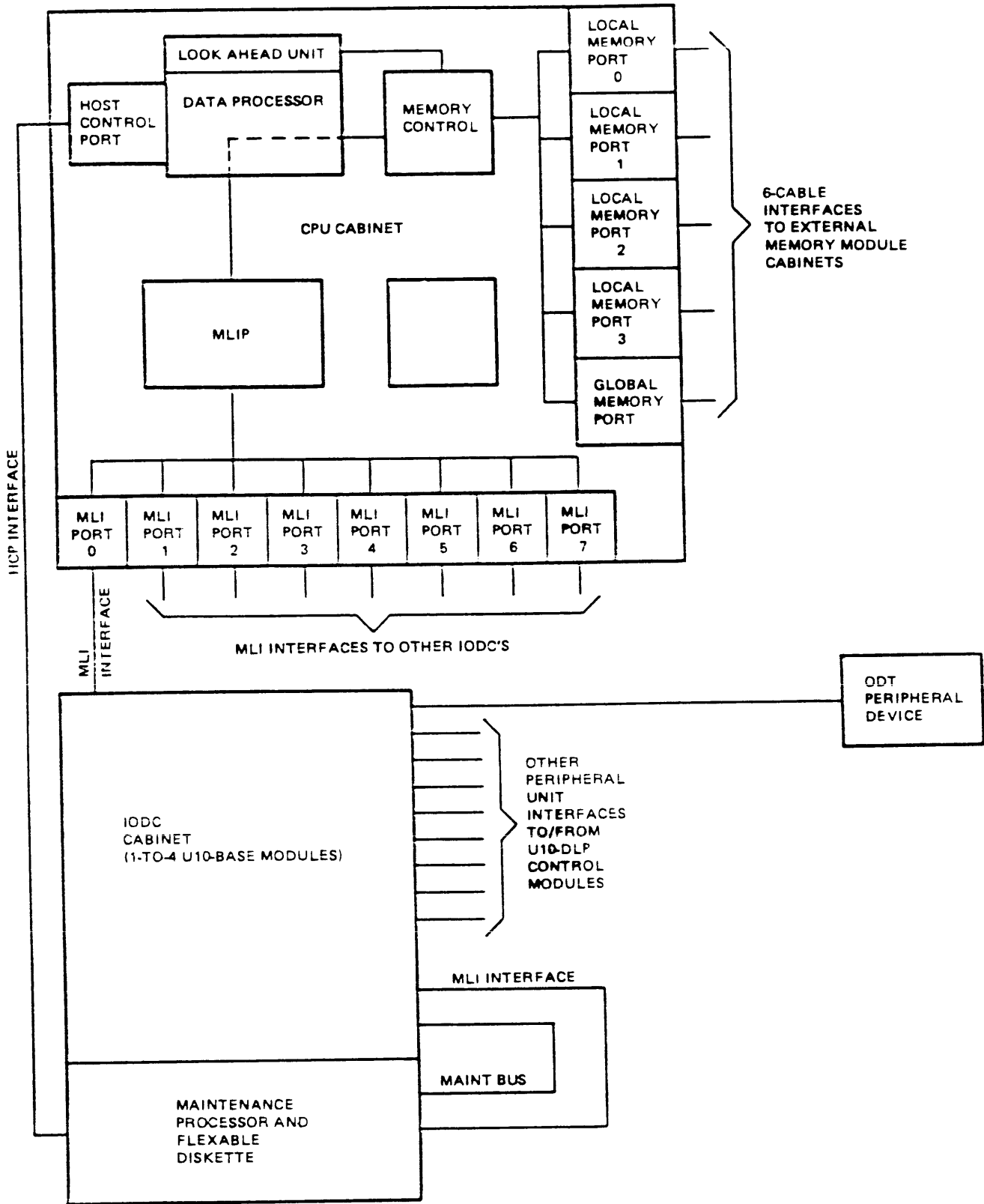
The DLPs are grouped into I/O-Data Comm (IODC) base modules. Each IODC base can hold up to 8 DLPs, depending on the number of circuit boards in the various types of DLPs used.

IODC base modules are connected by Message Level Interface (MLI) cables to the Message Level Interface Processor (MLIP).

The MLIP is an I/O processor providing the system interface to the Universal I/O subsystem. When an I/O operation is to be performed, the Program Controller generates an MLIP operator to be executed. Other processor operators continue to be executed concurrently with the MLIP operator, so that instruction execution continues while an I/O is being performed.

Data to be output to a device is transferred from the central system to the MLIP, then through the MLI to the IODC base, to the DLP, and finally to the designated device. Data input from peripherals follows the reverse route.

A SERIES AND B 5000/B 6000/B 7000 CONCEPTS  
 B 6900 HARDWARE OVERVIEW



MV4503

Figure B6900-2 B 6900 Block Diagram



**A SERIES AND B 5000/B 6000/B 7000 CONCEPTS  
B 6900 HARDWARE OVERVIEW**

**Practice**

Match the B 6900 system elements on the left with the descriptions on the right.

- |                                            |                                                                                                |
|--------------------------------------------|------------------------------------------------------------------------------------------------|
| _____ 1. Host Control Port                 | a. Controls the paths to memory.                                                               |
| _____ 2. Data Link Processor               | b. Performs system initialization and testing.                                                 |
| _____ 3. Data Processor                    | c. Is hardware designed to control a specific type of peripheral device.                       |
| _____ 4. Message Level Interface Processor | d. Connects an IODC base module to the Message Level Interface Processor.                      |
| _____ 5. Memory Control Module             | e. Interfaces the B 6900 processor to the I/O subsystem.                                       |
| _____ 6. Maintenance Diagnostic Processor  | f. Serves as an interface between the Data Processor and the Maintenance Diagnostic Processor. |
| _____ 7. Message Level Interface           | g. Contains the logic necessary to perform arithmetic functions and shift data.                |

**A SERIES AND B 5000/B 6000/B 7000 CONCEPTS  
B 6900 HARDWARE OVERVIEW**

This page left blank for formatting.

***SECTION B 7900***

***B 7900 HARDWARE OVERVIEW***

# **A SERIES AND B 5000/B 6000/B 7000 CONCEPTS B 7900 HARDWARE OVERVIEW**

## **Objective**

Identify the major hardware elements that constitute the B 7900 system.

## **Purpose**

The B 7900 system consists of a series of logical units housed in several cabinets, and the system console. This unit will introduce you to these elements.

## **Resources**

B 7900 System Operators Guide

B 7900 System Hardware Operational Guide

B 7900 System Capabilities and Features Guide

# A SERIES AND B 5000/B 6000/B 7000 CONCEPTS B 7900 HARDWARE OVERVIEW

## Central System Components

The B 7900 central system includes 1 or more Central Processing Modules, I/O Subsystem Modules, Memory Subsystem Modules, System Control Modules, and System Consoles. Additional Central Processing Modules, I/O Subsystem Modules, Memory Storage Modules, and System Consoles can be incorporated to build the required configuration.

## Central Processor Module (CPM)

The B 7900 system includes 1 or 2 CPMs, each of which includes the hardware modules described below.

1. Program Control Unit (PCU)

The PCU examines the object code stream, extracts the operators, and builds the execution string or "pipeline." The PCU also prepares the required data by assigning stack locations, and by requesting the Data Reference Unit (DRU) to read data from memory.

2. Data Reference Unit (DRU)

The DRU, upon command from the PCU, fetches data from memory, and places it in the Central Data Buffer or stack.

3. Memory Access Unit (MAU)

The MAU interfaces the CPM to main memory. It receives the memory address from the PCU for code fetches, from the DRU for data fetches, and from the SQ for writes to memory.

4. Store Queue (SQ)

The SQ reduces traffic to main memory by buffering data before it is sent to the MAU. Repeated stores to the same address will be performed as one store to main memory; stores to adjacent addresses will be grouped into one multi-word store.

5. Execution Unit (EU)

The EU actually performs all the arithmetic and logical operations in the system, by executing the pipelines of operators created by the PCU, using the data supplied by the DRU. The PCU and DRU prepare the operators and data, so that the EU can continue executing without being interrupted to do memory accesses.

A SERIES AND B 5000/B 6000/B 7000 CONCEPTS  
 B 7900 HARDWARE OVERVIEW

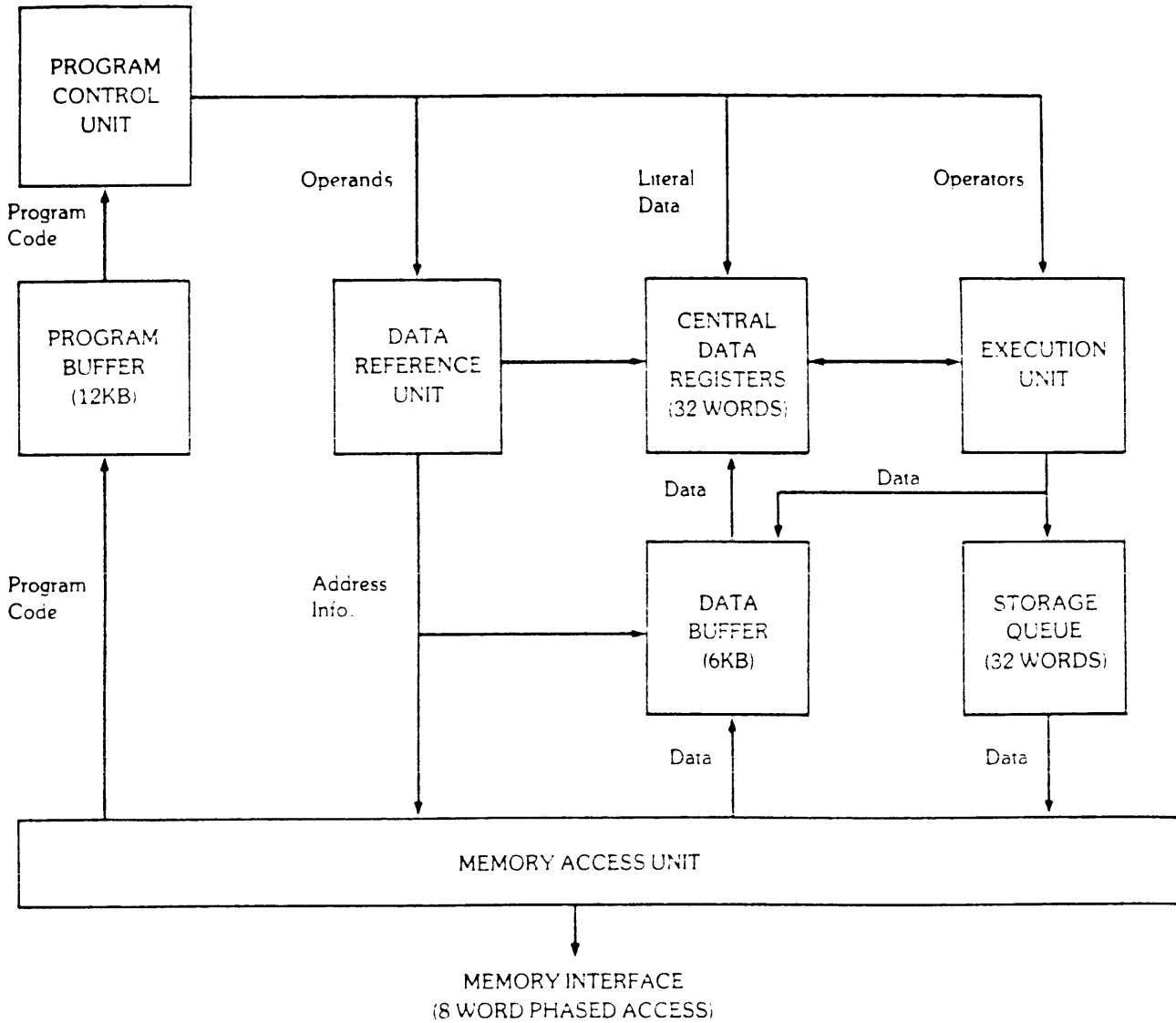


Figure B7900-1 CPM Block Diagram

# A SERIES AND B 5000/B 6000/B 7000 CONCEPTS B 7900 HARDWARE OVERVIEW

## *Input/Output Subsystem Module (IOSM)*

All B 7900 systems have at least 1 IOSM cabinet, to house the Host Data Unit, the Auxiliary Processor, and the IODC base modules (described in the I/O-Data Comm Subsystem section below).

### 1. Host Data Unit (HDU)

Each IOSM cabinet has an HDU, to handle all I/O data transfers between B 7900 main memory and the I/O subsystem. Refer to the I/O-Data Comm Subsystem section for further information about the HDU.

### 2. Auxiliary Processor/Auxiliary Maintenance Processor (AP/AMP)

The AP/AMP is actually a B 5900 processor, which operates in two distinct modes: Auxiliary and Maintenance.

In the Auxiliary Processor (AP) mode, the AP handles I/O finishes and other routine functions, which increases throughput by allowing the CPM to continue executing without those interruptions. If the CPM fails, the AP can either perform the CPM functions (at a slower rate than the CPM), or execute a user-written shutdown program to close files and data bases in an orderly fashion, so that recovery is not necessary. If a failure occurs, the ODT operator must select any specific jobs that are to continue running on the AP only in degraded mode.

In the Auxiliary Maintenance Processor (AMP) mode, the AMP can execute programs to do testing and diagnostics on hardware in the system.

## *Memory Subsystem Module (MSM)*

The B 7900 memory subsystem can contain a maximum of 16 million words (96 millions bytes) of memory, housed in 1 or 2 cabinets. A MSM cabinet holds a Memory Control, and a maximum of 4 Memory Storage Units (MSU), with 1 or 2 million words of memory each. The Memory Control allows a maximum of 8 requestors (CPMs, APs and HDUs) to access memory.

The techniques of interleaving and phasing enhance the performance of B 7900 memory. Interleaving is a method in which consecutive words are not written serially to consecutive memory locations, but are written in parallel to the 8 memory areas of the MSU. Phasing speeds memory access by reducing the number of processor cycles needed to retrieve words from memory. The B 7900 Memory Controller also has separate read and write buses, so that one requestor can read, and another can write, simultaneously in the same MSU.

The MCP can address 1 million words of memory at a time. B 7900 systems with more than 1 million words of memory are configured via ODT commands into Address Spaces (ASN = Address Space Number) and Environment Components (EC). Each Address Space consists of a shared component, which is common to all the Address Spaces, and a local component, which is specific to that Address Space. The shared plus the local components of an Address Space may occupy a maximum total of 1 million words. Each program in the mix will be assigned to an Address Space, or the program's data may be assigned to 1 Address Space and its object code to another. When a program has control of the processor, the MCP will access both the shared component and the local components of the appropriate Address Spaces for that program.

A SERIES AND B 5000/B 6000/B 7000 CONCEPTS  
 B 7900 HARDWARE OVERVIEW

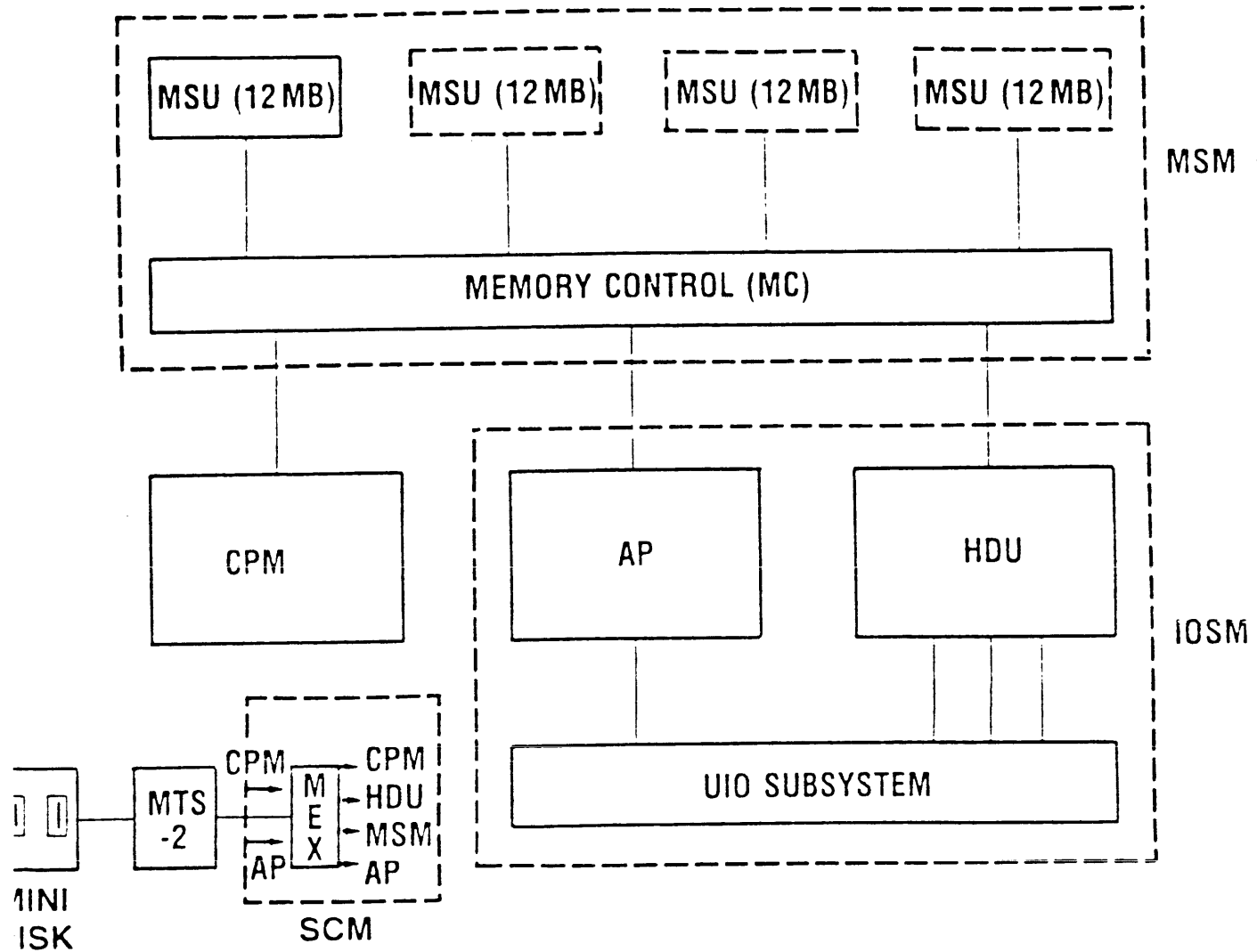


Figure B7900-2 Processor Modules



A SERIES AND B 5000/B 6000/B 7000 CONCEPTS  
B 7900 HARDWARE OVERVIEW

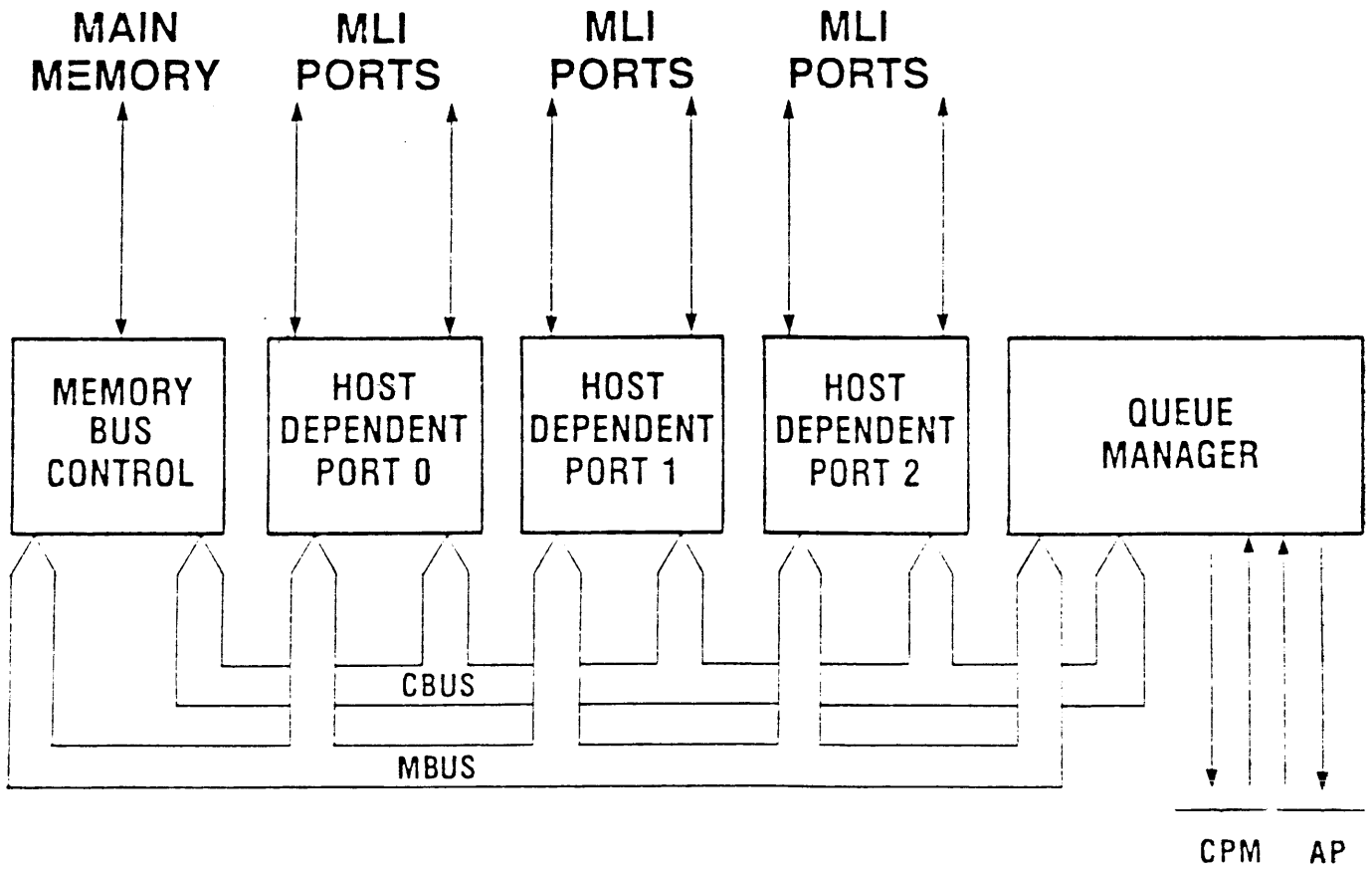


Figure B7900-3 Host Data Unit

A SERIES AND B 5000/B 6000/B 7000 CONCEPTS  
B 7900 HARDWARE OVERVIEW

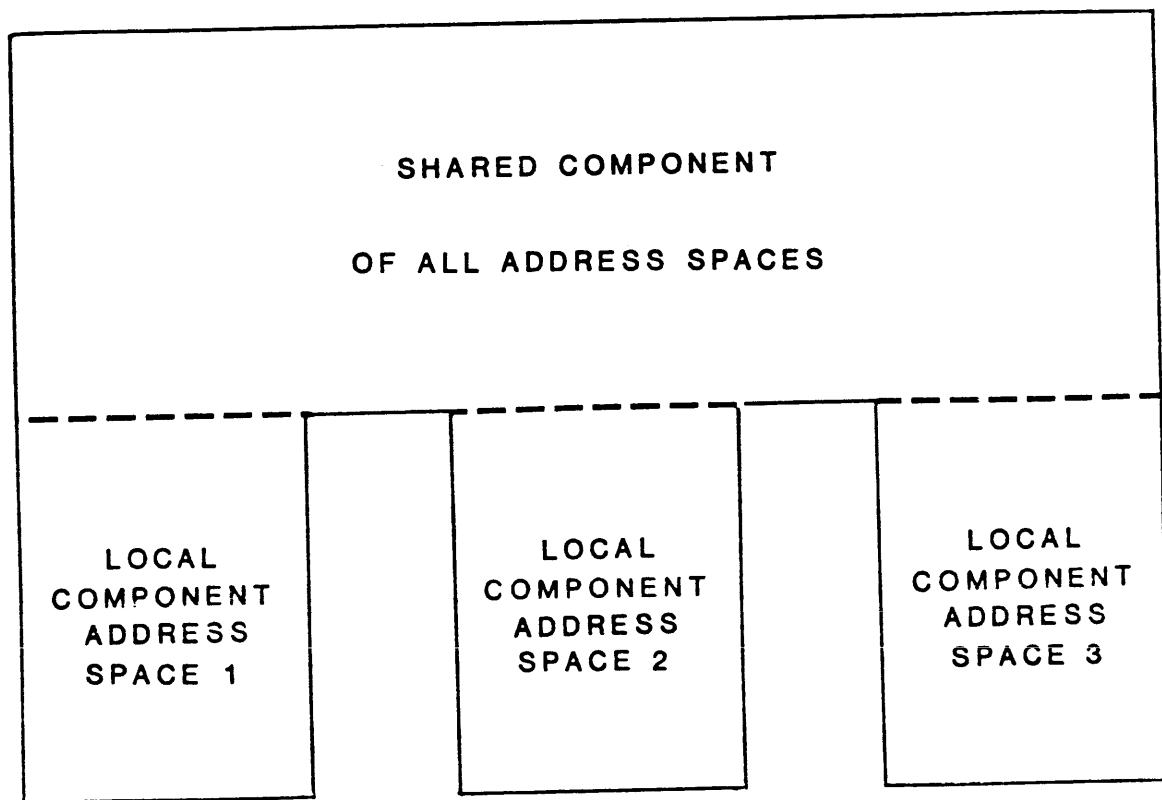


Figure B7900-4 Address Spaces

## A SERIES AND B 5000/B 6000/B 7000 CONCEPTS B 7900 HARDWARE OVERVIEW

### *System Control Module (SCM)*

The SCM is housed in a separate cabinet, and is comprised of the Card Test Station, the Master System Clock, and the Maintenance Exchange. The Card Test Station allows field engineers to test suspect circuit boards from various B 7900 modules. The Master System Clock supplies clock signals to the main system. The Maintenance Exchange provides the interface between all possible system maintenance processors, and all possible mainframe units under test.

### *System Console*

The System Console houses 2 Operator Display Terminals (ODTs), 1 maintenance terminal, 2 mini-disk drives, and the Maintenance Interface Processor. The ODTs are ET 1100s that communicate with operations personnel by displaying system status messages and receiving input commands. Refer to the section on maintenance hardware for further information on the maintenance terminal, the mini-disks, and the Maintenance Interface Processor.

### *Summary of Maintenance Hardware and Software*

B 7900 maintenance hardware includes the Auxiliary Maintenance Processor (AMP), the Maintenance Exchange (MEX), and the Card Test Station, which were described earlier. Additional maintenance hardware devices are the Maintenance Interface Processor (MIP), and the maintenance terminal. The MIP is located in the leg of the B 7900 console, and uses 2 mini-disk drives also located in the console leg. The maintenance terminal is an ET 2000 or a Modified MTS2 terminal that provides the operator interface for the MIP.

SYCON (SYstem CONsole) is a software program which executes on a maintenance terminal/MIP pair to allow configuration, initialization, and status checking for the B 7900 and its partitions. The operator loads SYCON into the MIP from the mini-disk drives, and then communicates with SYCON through menu screens on the maintenance terminal.

IDA (Interactive Diagnostic Access) is a program which runs under MCP control, and receives commands from an ODT or data comm terminal. IDA can perform maintenance tests on a running system, and execute canned procedures in various situations (e.g. generate a state dump in case of a failure).

BEAM and APCON are software packages for testing the AP/AMP from the MIP or from the AP/AMP respectively.

In addition, an RS232 communications link can be used to allow B 7900 testing and diagnostic programs to be executed by Burroughs personnel located at remote sites.

# A SERIES AND B 5000/B 6000/B 7000 CONCEPTS B 7900 HARDWARE OVERVIEW

## Partitions

Multiple processor B 7900s may be configured as a single system, or as separate partitions with at least 1 CPM, HDU, and MSM in each partition. The terms "box" and "component" refer to an individual CPM, HDU, AP, or MSM, so a partition can be defined as a subset of boxes running under a separate MCP. The boxes within a partition communicate with each other through the memory subsystem.

Partitioning creates logically separate systems running under separate MCPs, which allows different workload environments, or both production and test environments, to exist concurrently. Partitions give additional flexibility by enabling a running partition to act as the maintenance processor for hardware modules not in that partition.

The SYCON program (see the Maintenance Hardware and Software section) can be used to establish partitions at Halt/Load time, or a configuration file can be built so that RECONFIGURE commands can be entered at the ODT while the system is running. The ODT commands FREE and ACQUIRE are used to add or delete resources for an existing partition. A PC diagram, which is a file that defines the hardware configuration of the B 7900, is also required for partitioning.

## Input/Output and Data Comm Subsystem

The B 7900, like the B 5900, B 6900, and all A Series systems, uses Burroughs Universal I/O for both peripheral and data communications management.

Each peripheral subsystem is connected to a Data Link Processor (DLP) which is designed to control that specific type of peripheral. Some peripherals (e.g. line printers) can have only 1 device attached to each DLP, but other peripherals (e.g. disk packs, tapes) can have multiple devices attached to a single DLP through an exchange and/or controller. The data communications subsystem includes two special types of DLPs: Network Support Processors (NSPs), and Line Support Processors (LSPs).

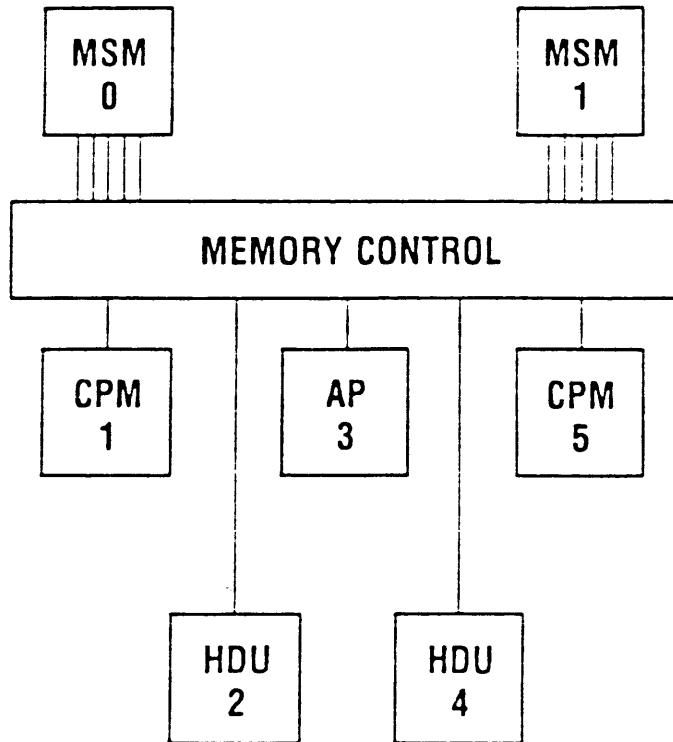
The DLPs are grouped into I/O-Data Comm (IODC) base modules. Each IODC base module can hold up to 8 DLPs, depending on the number of circuit boards in the various DLPs used. The DLPs and IODC base modules are located in the IOSM cabinet.

IODC base modules are connected by Message Level Interface (MLI) cables to a Host Dependent Port (HDP). The B 7900 can have a maximum of 3 HDPs, each controlling 2 MLIs.

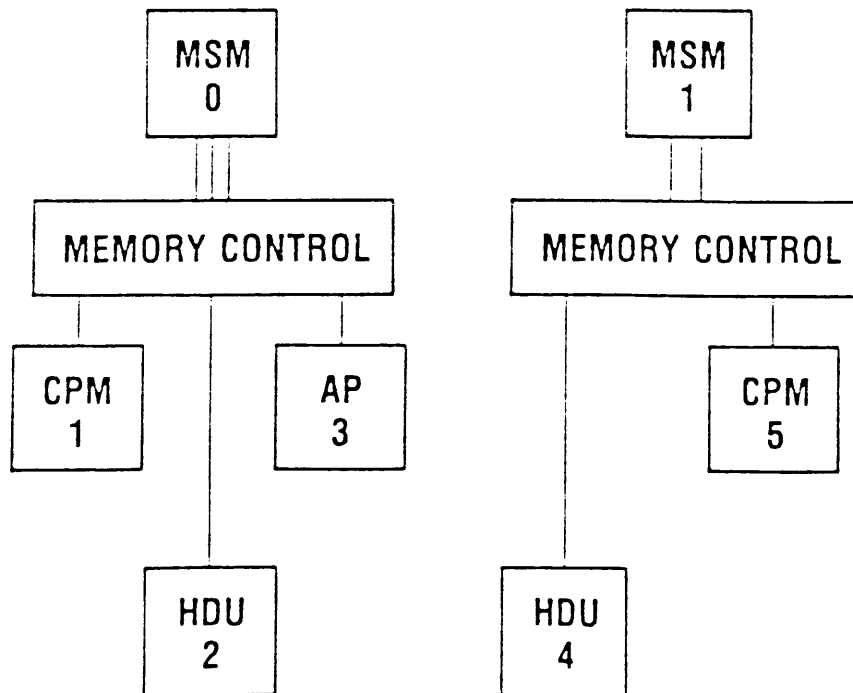
HDPs are components of the Host Data Unit (HDU), which handles all I/O data transfers between main memory and the I/O subsystem. HDPs provide the HDU interface to the I/O subsystem; the Memory Bus Control (MBC) provides the HDU interface to main memory. The Queue Manager (QM), which is the third major component of the HDU, maintains the I/O queues, and schedules activity for the HDPs.

Data to be output to a device is transferred from main memory through the MBC to a HDP, then through a MLI to the IODC base, to the DLP, and finally to the designated device. Data input from peripherals follows the reverse route.

A SERIES AND B 5000/B 6000/B 7000 CONCEPTS  
B 7900 HARDWARE OVERVIEW



B7900 SYSTEM PARTITION, SINGLE SYSTEM IMAGE



B7900 SYSTEM PARTITION A

B7900 SYSTEM PARTITION B

TWO B7900 SYSTEM PARTITIONS

**A SERIES AND B 5000/B 6000/B 7000 CONCEPTS  
B 7900 HARDWARE OVERVIEW**

**Practice**

Match the B 7900 system elements on the left with the descriptions on the right.

- |                               |                                                                                  |
|-------------------------------|----------------------------------------------------------------------------------|
| _____ 1. Data Link Processor  | a. Includes a local and a shared memory component.                               |
| _____ 2. Address Space        | b. Buffers data to reduce the number of writes to memory.                        |
| _____ 3. SYCON                | c. Performs the logical and arithmetic operators in the processor.               |
| _____ 4. Execution Unit       | d. Is hardware designed to control a specific type of peripheral device.         |
| _____ 5. Program Control Unit | e. Functions as a maintenance processor, or as an auxiliary processor.           |
| _____ 6. Store Queue          | f. Manages all data transfers between main memory and the I/O subsystem.         |
| _____ 7. Host Data Unit       | g. Is a subset of B 7900 modules which has been configured as a separate system. |
| _____ 8. Partition            | h. Is a program to initialize and configure the B 7900 from the console.         |
| _____ 9. AP/AMP               | i. Builds the operator pipeline.                                                 |

**SECTION C**  
**COMPILE LISTINGS AND PROGRAM**  
**DUMPS**

```

%%%%% SAMPLE ALGOL PAYROLL PROGRAM USED IN SECTION 9 %%%%%
$SET LIST STACK CODE $
BEGIN
REAL HOURS_WORK, HOURLY_RATE, GROSS_PAY,
  NET_PAY, TOTAL_TAXES;
FILE EMPLOYEES (KIND=DISK, TITLE="EMPLOYEES/ACTIVE.");
ARRAY TAX_TABLE [0:9];
PROCEDURE CALC_TAXES;
  BEGIN
    REAL FICA_TAX, FED_TAX;
    % CALCULATE FICA TAX AND FED TAX
    TOTAL_TAXES := FICA_TAX + FED_TAX;
    PROGRAMDUMP (FILES, ARRAYS, BASE, CODE);
  END OF CALC_TAXES;
PROCEDURE CALC_CHECK;
  BEGIN
    REAL REG_PAY, OT_PAY;
    PROCEDURE CALC_OT;
      BEGIN
        OT_PAY := HOURLY_RATE * .5 * (HOURS_WORK - 40);
        PROGRAMDUMP (FILES, ARRAYS);
      END OF CALC_OT;
    REG_PAY := HOURS_WORK * HOURLY_RATE;
    IF HOURS_WORK GT 0 THEN CALC_OT;
    GROSS_PAY := REG_PAY + OT_PAY;
    CALC_TAXES;
    NET_PAY := GROSS_PAY - TOTAL_TAXES;
  END OF CALC_CHECK;
%%%% MAIN LOGIC %%%
% READ EMPLOYEE, MOVE VALUES INTO VARIABLES
HOURS_WORK := 45;
CALC_CHECK;
% PRINT CHECK
END OF PROGRAM.
0000050
00000100
00001000
00001100
00001200
00001300
00001400
00001500
00001600
00001700
00001800
00001900
00001950
00002000
00002100
00002200
00002300
00002400
00002500
00002600
00002650
00002700
00002800
00002900
00003000
00003100
00003200
00003300
00003400
00003500
00003550
00003600
00003700
00003800

```

C-2



OBJECT / EP4195 / PAYROLL / ALGOL / OT ON DISK  
 =====

```

    %%%% SAMPLE ALGOL PAYROLL PROGRAM USED IN SECTION 9 %%%%
    $SET LIST STACK CODE $
    BEGIN
(01,0002) = BLOCK#1
(01,0003) = SEGMENT DESCRIPTOR
                                003:0000:0 NVLD
                                FF
                                BLOCK#1 IS SEGMENT 0003
                                1 00001100 003:0000:1
    REAL HOURS_WORK, HOURLY_RATE, GROSS_PAY,
(02,0002) = HOURS_WORK
(02,0003) = HOURLY_RATE
(02,0004) = GROSS_PAY
                                NET_PAY, TOTAL_TAXES;
                                00001200 003:0000:1
(02,0005) = NET_PAY
(02,0006) = TOTAL_TAXES
                                FILE EMPLOYEES (KIND=DISK,TITLE="EMPLOYEES/ACTIVE.");
                                00001300 003:0000:1
(02,0007) = FUNNY SIRW
(02,0008) = EMPLOYEES
                                DATA POOL AT (02,0008):
                                0000 010000000000
                                0001 0D010109C5D4
                                0002 D7D3D6E8C5C5
                                0003 E20000000000
                                0004 031D04030801
                                0005 020011C5D4D7
                                0006 D3D6E8C5C5E2
                                0007 61C1C3E3C9E5
                                0008 C54B00000000
                                DATA LENGTH IN WORDS IS 0009
                                00001400 003:0000:1
                                ARRAY TAX_TABLE [0:9];
(02,0009) = TAX_TABLE
                                PROCEDURE CALC_TAXES;
                                00001500 003:0000:1
(02,000A) = CALC_TAXES
                                BEGIN
                                REAL FICA_TAX, FED_TAX;
                                00001600 003:0000:1
(01,0004) = SEGMENT DESCRIPTOR
                                00001700 003:0000:1
                                CALC_TAXES IS SEGMENT 0004
                                004:0000:0 NVLD
                                FF
(03,0002) = FICA_TAX
(03,0003) = FED_TAX
                                % CALCULATE FICA TAX AND FED TAX
                                TOTAL_TAXES := FICA_TAX + FED_TAX;
                                2 00001800 004:0000:1
                                00001900 004:0000:1
                                004:0000:1 VALC (03,0002) 3002
                                004:0000:3 VALC (03,0003) 3003
                                004:0000:5 ADD 80
                                004:0001:0 NAMC (02,0006) 5006
                                004:0001:2 STOD 88
                                PROGRAMDUMP (FILES, ARRAYS, BASE, CODE);
                                00001950 004:0001:3
(01,0005) = PROGRAMDUMP
                                004:0001:3 MKST AE
                                004:0001:4 NAMC (01,0005) 6005
                                004:0002:0 LT16 1920 B30780
                                004:0002:3 CHSN 8E
                                004:0002:4 ENTR AB
    
```

C-3

A SERIES AND B 5000/B 6000/B 7000 CONCEPTS  
 COMPILE LISTINGS AND PROGRAM DUMPS

END OF CALC\_TAXES;

00002000 004:0002:5

```

004:0002:5 EXIT A3
***** STACK BUILDING CODE FOR LEVEL 03 *****
004:0003:0 ZERO B0
004:0003:1 ZERO B0
004:0003:2 PUSH B4
004:0003:3 BRUN 0000:1 A22000
004:0004:0 NVLD FF
004:0004:1 NVLD FF
004:0004:2 NVLD FF
004:0004:3 NVLD FF
004:0004:4 NVLD FF
004:0004:5 NVLD FF

```

CALC\_TAXES(004) LENGTH IN WORDS IS 0005  
2 00002100 003:0000:1

PROCEDURE CALC\_CHECK;

(02,000B) = CALC\_CHECK

BEGIN

REAL REG\_PAY, OT\_PAY;

(01,0006) = SEGMENT DESCRIPTOR

CALC\_CHECK IS SEGMENT 0006

006:0000:0 NVLD FF

(03,0002) = REG\_PAY

(03,0003) = OT\_PAY

PROCEDURE CALC\_OT;

2 00002400 006:0000:1

(03,0004) = CALC\_OT

BEGIN

OT\_PAY := HOURLY\_RATE \* .5 \* (HOURS\_WORK - 40);

00002500 006:0000:1

00002600 006:0000:1

```

006:0000:1 VALC (02,0003) 1003
006:0000:3 LT8 155 B29B
006:0000:5 ISOL 9:48 9A0930
006:0001:2 MULT 82
006:0001:3 VALC (02,0002) 1002
006:0001:5 LT8 40 B228
006:0002:1 SUBT 81
006:0002:2 MULT 82
006:0002:3 NAMC (03,0003) 7003
006:0002:5 STOD B8
PROGRAMDUMP (FILES, ARRAYS);
006:0003:0 MKST AE
006:0003:1 NAMC (01,0005) 6005
006:0003:3 LT16 1280 B30500
006:0004:0 CHSN 8E
006:0004:1 ENTR AB

```

3 00002650 006:0003:0

END OF CALC\_OT;

006:0004:2 EXIT A3

```

***** STACK BUILDING CODE FOR LEVEL 04 *****

```

REG\_PAY := HOURS\_WORK \* HOURLY\_RATE;

3 00002800 006:0004:3

```

006:0004:3 VALC (02,0002) 1002
006:0004:5 VALC (02,0003) 1003
006:0005:1 MULT 82
006:0005:2 NAMC (03,0002) 7002
006:0005:4 STOD B8

```

00002900 006:0005:5

IF HOURS\_WORK GTR 0 THEN CALC\_OT;

```

006:0005:5 VALC (02,0002) 1002
006:0006:1 ZERO B0
006:0006:2 GRTR 8A
006:0007:0 MKST AE
006:0007:1 NAMC (03,0004) 7004
006:0007:3 ENTR AB
006:0006:3 BRFL-LINK 0000:0 A00000

```

GROSS\_PAY := REG\_PAY + OT\_PAY;

00003000 006:0007:4

C-4

A SERIES AND B 5000/B 6000/B 7000 CONCEPTS  
COMPILE LISTINGS AND PROGRAM DUMPS

```

                                006:0007:4 VALC (03,0002) 3002
                                006:0008:0 VALC (03,0003) 3003
                                006:0008:2 ADD 80
                                006:0008:3 NAMC (02,0004) 5004
                                006:0008:5 STOD 88
CALC_TAXES;
                                006:0009:0 MKST AE
                                006:0009:1 NAMC (02,000A) 500A
                                006:0009:3 ENTR AB
NET_PAY := GROSS_PAY - TOTAL TAXES;
                                006:0009:4 VALC (02,0004) 1004
                                006:000A:0 VALC (02,0006) 1006
                                006:000A:2 SUBT 81
                                006:000A:3 NAMC (02,0005) 5005
                                006:000A:5 STOD 88
END OF CALC_CHECK;
                                006:000B:0 EXIT A3
                                006:0006:3 BRFL 0007:4 A08007
***** STACK BUILDING CODE FOR LEVEL 03 *****
                                006:000B:1 ZERO B0
                                006:000B:2 ZERO B0
(03,0004) = = = = =
                                006:000B:3 MPCW BF
                                006:000C 006:0000:1 000200012006
                                006:000D:0 PUSH B4
                                006:000D:1 BRUN 0004:3 A26004
                                006:000D:4 NVLD FF
                                006:000D:5 NVLD FF
                                CALC_CHECK(006) LENGTH IN WORDS IS 000E
                                2 00003400 003:0000:1
                                00003500 003:0000:1
                                00003550 003:0000:1
                                003:0000:1 LT8 45 B22D
                                003:0000:3 NAMC (02,0002) 5002
                                003:0000:5 STOD 88
CALC_CHECK;
                                003:0001:0 MKST AE
                                003:0001:1 NAMC (02,000B) 500B
                                003:0001:3 ENTR AB
                                00003700 003:0001:4
                                00003800 003:0001:4
% PRINT CHECK
END OF PROGRAM.
(01,0007) = BLOCKEXIT
                                003:0001:4 MKST AE
                                003:0001:5 NAMC (01,0007) 6007
                                003:0002:1 ENTR AB
                                003:0002:2 EXIT A3
***** STACK BUILDING CODE FOR LEVEL 02 *****
                                003:0002:3 ZERO B0
                                003:0002:4 ZERO B0
                                003:0002:5 ZERO B0
                                003:0003:0 ZERO B0
                                003:0003:1 ZERO B0
(02,0007) = = = = =
                                003:0003:2 NAMC (01,0000) 6000
                                003:0003:4 STFF AF
                                003:0003:5 BSET 13 960D
(02,0008) = = = = =
                                003:0004:1 LT48 BE
                                003:0005 270000940001 (3"1160000045000001")
                                003:0006:0 LT8 5 B205
                                003:0006:2 STAG 95B4

```



B6900 PROGRAMDUMP FOR STACK 2F8 (MIX 3014/3020) BOSR=45932  
NAME: (CONCEPTS)OBJECT/EP4195/PAYROLL/ALGOL/OT ON SYSTEMSED.

TUESDAY, FEBRUARY 11, 1986 14:59:30

MCP 36.140.3025: \*SYSTEM/MCP36140. INTRINSICS: SYSTEM/INTRINSICS ON DISK. (LOADED)  
SYSTEM SERIAL: #2372 HOSTNAME: SYSEDB6900. GROUP ID: DEFAULT.

CAUSE OF DUMP: PROGRAM REQUESTED @ 006:0004:2, 006:0007:4, 003:0001:4.

PIB HISTORY WORD AT ENTRY TO PROGRAM DUMP: 0 000000 000000.

PROGRAMDUMP OPTIONS: ARRAY(S), FILE(S)

0328 = LOSR (00045C5A)

002A (01,0002) 0 400000 000501 OP: OCT:20000000 00002401 , EBC: ?????, DEC:-1281  
0029 3 000400 412006 RCW: LL=04, NORML STATE [USER SEGMENT @ 0006:0004:2]  
SEG DESC: 3 800000 EB30E4  
CODE: 3 093082 1002B2 3 288182 7003B8 3 AE6005 B30500 >3 8EABA3 100210< 3 038270 02B810  
0028 ----D[01]=>3 C12000 804002 \*MSCW: PREVIOUS MSCW @ 0026; D[00]=0008 IN STACK 012  
0027 3 000800 70E006 RCW: LL=03, NORML STATE [USER SEGMENT @ 0006:0007:4]  
SEG DESC: 3 800000 EB30E4  
CODE: 3 8EABA3 100210 3 038270 02B810 3 02B08A A08007 >3 AE7004 AB3002< 3 300380 5004B8  
0026 ----D[04]=>3 6F8002 110005 \*MSCW: PREVIOUS MSCW @ 0021; D[03]=0021  
0025 (03,0004) 7 2F8200 012006 PCW: LL=04, D[1] SEGMENT @ 0006:0000:1, NORML STATE  
0024 (03,0003) 0 000000 000000  
0023 (03,0002) 0 000000 000000  
0022 3 000800 10A003 RCW: LL=02, NORML STATE [USER SEGMENT @ 0003:0001:4]  
SEG DESC: 3 800000 FC0786  
CODE: 3 FFB22D 5002B8 >3 AE500B ABAE60< 3 07ABA3 B0B0B0  
0021 ----D[03]=>3 6F8001 40C00D \*MSCW: PREVIOUS MSCW @ 0014; D[02]=0014  
0020 (02,000C) 6 800000 002800 SCW: (BLOCK BELOW DECLARED FILES, SNGL-DIM ARRAYS)  
001F (02,0008) 7 2F8200 B0E006 PCW: LL=03, D[1] SEGMENT @ 0006:0008:1, NORML STATE  
001E (02,000A) 7 2F8000 30E004 PCW: LL=03, D[1] SEGMENT @ 0004:0003:0, NORML STATE  
001D (02,0009) 5 000000 A00000 DESC [ABSENT-MOM]: DATA, LENGTH=10 (UNREFERENCED OLAY SPACE)  
001C (02,0008) 5 270000 940001 DESC [ABSENT-MOM]: FILE DESCRIPTION, LENGTH=9 (CODEFILE ADRS=1)  
001B (02,0007) 1 6F9000 402000 SIRW: OFFSET=0004 (0004+0000) IN STACK 2F9  
001A (02,0006) 0 000000 000000  
0019 (02,0005) 0 000000 000000  
0018 (02,0004) 0 000000 000000  
0017 (02,0003) 0 000000 000000  
0016 (02,0002) 0 000000 00002D OP: OCT:00000000 00000055 , EBC:??????, DEC:45  
0015 3 00024E E88FCD RCW: LL=02, CNTRL STATE [MCP SEGMENT @ 0FCD:04EE:1 (16562000)]  
SEG DESC: 3 8800B6 ABCAC1  
CODE: 3 BEFFFF FFFFFF 3 4A6870 C30C30 3 ABA3AE 400AAB >3 A3B234 B22695< 3 B8B180 95B9A2  
0014 ----D[02]=>3 EF9000 408002 \*MSCW: PREVIOUS MSCW @ 0012; D[01]=0004 IN STACK 2F9  
0013 3 000000 002000 RCW: DUMMY (RUN)  
0012 ----D[02]=>3 F822D0 808011 \*MSCW: PREVIOUS MSCW @ 0001; D[01]=2D08 IN STACK 382  
0000 = BOSR (00045932)

C-7

A SERIES AND B 5000/B 6000/B 7000 CONCEPTS  
COMPILE LISTINGS AND PROGRAM DUMPS

MCP 36.140.3025: \*SYSTEM/MCP36140. INTRINSICS: SYSTEM/INTRINSICS ON DISK. (LOADED)  
SYSTEM SERIAL: #2372 HOSTNAME: SYSEDB6900. GROUP ID: DEFAULT.

CAUSE OF DUMP: PROGRAM REQUESTED @ 004:0002:5, 006:0009:4, 003:0001:4.

PIB HISTORY WORD AT ENTRY TO PROGRAM DUMP: 0 000000 000000.

PROGRAMDUMP OPTIONS: ARRAY(S), BASE, CODE, FILE(S)

0328 = LOSR (00045C5A)

002C (01,0002) 0 400000 000781 OP: OCT:20000000 00003601 , EBC:????a, DEC:-1921  
002B 3 000A00 20E004 RCW: LL=03, NORML STATE [USER SEGMENT @ 0004:0002:5]  
SEG DESC: 3 800000 5B30D3

002A ----D[01]=>3 C12000 804004 \*MSCW: PREVIOUS MSCW @ 0026; D[00]=0008 IN STACK 012  
CODE: 3 FF3002 300380 3 5006B8 AE6005 >3 B30780 8EABA3< 3 BOB0B4 A22000

0029 (03,0003) 0 000000 000000  
0028 (03,0002) 0 000000 000000  
0027 3 000800 90E006 RCW: LL=03, NORML STATE [USER SEGMENT @ 0006:0009:4]  
SEG DESC: 3 800000 EB30E4

0026 ----D[03]=>3 6F8001 40C005 \*MSCW: PREVIOUS MSCW @ 0021; D[02]=0014  
CODE: 3 02B08A A08007 3 AE7004 AB3002 3 300380 5004B8 >3 AE500A AB1004< 3 100681 5005B8

0025 (03,0004) 7 2F8200 012006 PCW: LL=04, D[1] SEGMENT @ 0006:0000:1, NORML STATE

0024 (03,0003) 0 000000 000000  
0023 (03,0002) 0 000000 000000  
0022 3 000800 10A003 RCW: LL=02, NORML STATE [USER SEGMENT @ 0003:0001:4]  
SEG DESC: 3 800000 FC0786

0021 ----D[03]=>3 6F8001 40C00D \*MSCW: PREVIOUS MSCW @ 0014; D[02]=0014  
CODE: 3 FFB22D 5002B8 >3 AE500B ABAE60< 3 07ABA3 BOB0B0

0020 (02,000C) 6 800000 002800 SCW: (BLOCK BELOW DECLARED FILES, SNGL-DIM ARRAYS)  
001F (02,000B) 7 2F8200 80E006 PCW: LL=03, D[1] SEGMENT @ 0006:000B:1, NORML STATE  
001E (02,000A) 7 2F8000 30E004 PCW: LL=03, D[1] SEGMENT @ 0004:0003:0, NORML STATE  
001D (02,0009) 5 000000 A00000 DESC [ABSENT-MOM]: DATA, LENGTH=10 (UNREFERENCED OLAY SPACE)  
001C (02,0008) 5 270000 940001 DESC [ABSENT-MOM]: FILE DESCRIPTION, LENGTH=9 (CODEFILE ADRS=1)  
001B (02,0007) 1 6F9000 402000 SIRW: OFFSET=0004 (0004+0000) IN STACK 2F9

001A (02,0006) 0 000000 000000  
0019 (02,0005) 0 000000 000000  
0018 (02,0004) 0 000000 000000  
0017 (02,0003) 0 000000 000000  
0016 (02,0002) 0 000000 00002D OP: OCT:00000000 00000055 , EBC:??????, DEC:45  
0015 3 00024E E88FCD RCW: LL=02, CNTRL STATE [MCP SEGMENT @ 0FCD:04EE:1 (16562000)]  
SEG DESC: 3 8800B6 ABCAC1

0014 ----D[02]=>3 EF9000 408002 \*MSCW: PREVIOUS MSCW @ 0012; D[01]=0004 IN STACK 2F9  
CODE: 3 BEFFFF FFFFFFFF 3 4A6870 C30C30 3 ABA3AE 400AAB >3 A3B234 B22695< 3 88B180 95B9A2

0013 3 000000 002000 RCW: DUMMY (RUN)  
0012 ----D[02]=>3 F822D0 808011 \*MSCW: PREVIOUS MSCW @ 0001; D[01]=2D08 IN STACK 382

0011 5 800004 422D90 DESC [PRESENT-MOM]: DATA, LENGTH=68

0010 0 000000 000000  
000F 0 000000 000000  
000E 0 000000 000000  
000D 0 000300 014000 OP: OCT:00001400 00240000 , EBC:???? ? , DEC:12884983808  
000C 0 000000 0011F0 OP: OCT:00000000 00010760 , EBC:?????0 , DEC:4592

C-8

A SERIES AND B 5000/B 6000/B 7000 CONCEPTS  
COMPILE LISTINGS AND PROGRAM DUMPS

```

0008      0 000000 000000
000A      5 000000 400000  DESC [ABSENT-MOM]: DATA, LENGTH=4 (UNREFERENCED OLA SPACE)
0009      0 000000 000000
0008      0 000000 000000
0007      0 000000 000000
0006      7 2F8000 000000  PCW: LL=00, D[0] SEGMENT @ 0000:0000:0, NORML STATE
0005      5 800000 655A26  DESC [PRESENT-MOM]: DATA, LENGTH=6
0004      6 800000 000000  SCW:
0003      0 000000 000000
0002      0 00FFFF F020D2  OP:   OCT:00177777 74020322 , EBC:?????K, DEC:4398038189710.0
0001      3 6F8000 000000  *MSCW: DUMMY FOR STACK 2F8
0000      0 000000 000001  PROCESSOR ID

```

0000 = BOSR (00045932)

DUMP OF PROCESS INFORMATION BLOCK (TASK VARIABLE)

```

0(0000) 3 C12000 804001 0 000000 000001 0 000000 0002F9 0 000000 240082 0 400002 575CFB 0 000000 00099B
6(0006) 0 000000 0008B6 0 000000 000000 0 000000 000003 0 000000 000000 0 000000 0C0001 0 000000 000000
12(000C) 0 000000 000012 0 000000 000000 0 600A03 9FCCE2 0 000000 000000 0 000000 0C0000 0 000000 000000
18(0012) 0 000000 000000 THRU 21(0015)
22(0016) 5 800000 45C5EB
0(0000) 0 170212 04C4C9 0 E2D209 E2E8E2 0 E3C5D4 E2C5C4 0 04C4C9 E2D2A1
23(0017) 0 000000 000000 THRU 25(0019)
26(001A) 5 800000 9266E7
0(0000) 0 350707 08C3D6 0 D5C3C5 D7E3E2 0 06D6C2 D1C5C3 0 E306C5 D7F4F1 0 F9FE07 D7C1E8 0 D9D6D3 D305C1
6(0006) 0 D3C7D6 D302D6 0 E309E2 E8E2E3 0 C5D4E2 C5C400
27(001B) 0 000000 000000 THRU 30(001E)
31(001F) 5 800000 2BA132
0(0000) 0 09E2E8 E2E3C5 0 D4E2C5 C40000
32(0020) 0 002EB0 000000 5 800000 64A114
0(0000) 0 000000 000000 THRU 5(0005)
34(0022) 0 000000 000000 0 6EB001 40000A 0 000000 000000 0 000000 000000 0 3EAA53 C54898 0 000000 000000
40(0028) 0 000000 000000 1 6F9000 400002 0 000000 000000 0 000000 000000 0 000000 000000 0 000000 000000
46(002E) 0 000000 710075 0 000000 000000 0 000000 000000 0 000000 000000 0 000000 000000 0 000000 000000
52(0034) 0 2EB010 00C802 0 000000 000000 0 000030 000000 0 000000 000000 5 800001 A32F49
0(0000) 0 3F3F03 400000 0 000000 012000 0 000000 000000 0 000000 000000 0 000000 000000 0 600005 000640
6(0006) 0 000000 000000 0 000200 000000 0 000000 003000 0 000000 000000 0 000000 000000 0 000000 000000
12(000C) 0 000000 000000 THRU 17(0011)
18(0012) 0 080600 03911F 0 000000 000000 0 000000 000000 0 000000 000000 0 000000 000000 0 800000 000000
24(0018) 0 800000 000000 0 000000 000000
57(0039) 0 000000 000000 0 4002EB 000000 0 000000 000000 0 000000 000000 0 000000 000000 2 000000 000000
63(003F) 2 000000 000000 0 000000 000000 0 000000 000000 0 000000 000000 0 000000 000001 0 800200 000000
69(0045) 0 000000 000000 5 800001 E1C189
0(0000) 0 10C800 000C38 0 000000 100005 5 C00000 80F78E 5 C00001 E1C189 5 E00001 91C189 5 E00001 81C189
6(0006) 0 000000 0A000C 0 000000 000000 5 E00000 10F4E2 0 000000 000000 5 E00001 5AB0C4 0 000000 000000
12(000C) 0 000000 000000 0 00053C 6E4C31 0 000000 005814 0 0122F8 300003 0 000030 011100 0 000000 000000
18(0012) 0 120000 053725 5 C00001 5AB0C4 1 6F8000 C0000C 0 000000 000000 0 000000 000000 0 000150 000000
24(0018) 0 C00001 500000 0 802000 053725 0 000000 000000 0 000000 000000 0 000000 000000 0 120000 00002E
71(0047) 0 000000 000001 0 000000 000000 0 000000 000000 0 000000 000000 0 000000 000000 0 000000 000000
77(004D) 0 200000 03C235 7 38205F F88EEB 1 7822D0 80004E 0 000000 080000 0 000000 000000 0 200A03 B9EAD
83(0053) 0 000000 000001 0 200A03 B9F52A 0 000000 000000 0 200000 01BB10 0 200000 03C235 0 000000 0C6007
89(0059) 0 000000 00003E 0 200000 00BD83 0 000000 000003 0 200000 000F24 0 000000 000001 0 000000 000000
95(005F) 0 000000 000000 THRU 99(0063)
100(0064) 0 000008 200000 0 000000 000000 0 007FFF FFFFFFF 0 000008 000000 0 000000 000108 0 000000 000269
106(006A) 0 000000 000000 THRU 107(006B)
108(006C) 0 007FFF FFFFFFF 0 000000 000000 0 000000 000000 0 000000 000000 5 E0002E B0B51E 0 000000 000000
114(0072) 5 E0007E B0B51E 0 000000 428072 0 004000 000000 0 000000 000032 0 000000 000000 0 000000 000000
120(0078) 0 2F800B C50BCC 0 000000 001194 0 800001 300200 0 000000 000000 0 000000 000000 0 000000 000000
126(007E) 0 000000 000000 0 2EB000 000000 0 0C0101 08C3D6 0 D5C3C5 D7E3E2 0 000000 000000 0 000000 000000

```

C-9

A SERIES AND B 5000/B 6000/B 7000 CONCEPTS  
 COMPILE LISTINGS AND PROGRAM DUMPS

DUMP OF SEGMENT DICTIONARY (STACK 2F9)

000D = LOSR (0003EE0D)

```

000D (01,0009) 1 432001 400032 SIRW: OFFSET=0046 (0014+0032) IN STACK 032
000C (01,0008) 5 070000 00008C DESC [ABSENT-MOM]: INTRINSIC #0,140
000B (01,0007) 1 412000 80000A SIRW: OFFSET=0012 (0008+000A) IN STACK 012
000A (01,0006) 3 800000 EB30E4 SEG DESC [PRESENT-MOM]: LENGTH=14
    0(0000) 3 FF1003 B29B9A 3 093082 1002B2 3 288182 7003B8 3 AE6005 B30500 3 8EABA3 100210 3 038270 02B810
    6(0006) 3 02B08A A08007 3 AE7004 AB3002 3 300380 5004B8 3 AE500A AB1004 3 100681 5005B8 3 A3B0B0 BFFFFFF
    12(000C) 3 000200 012006 3 B4A260 04FFFF
0009 (01,0005) 1 412000 800017 SIRW: OFFSET=001F (0008+0017) IN STACK 012
0008 (01,0004) 3 800000 5B30D3 SEG DESC [PRESENT-MOM]: LENGTH=5
    0(0000) 3 FF3002 300380 3 5006B8 AE6005 3 B30780 8EABA3 3 B0B0B4 A22000 3 FFFFFFF FFFFFFF
0007 (01,0003) 3 800000 FC0786 SEG DESC [PRESENT-MOM]: LENGTH=15
    0(0000) 3 FFB22D 5002B8 3 AE500B ABAE60 3 07ABA3 B0B0B0 3 B0B060 00AF96 3 0DBEFF FFFFFFF 3 270000 940001
    6(0006) 3 B20595 B4BEFF 3 000000 A00000 3 B20595 B4BFFF 3 000000 30E004 3 BFFFFFF FFFFFFF 3 000200 B0E006
    12(000C) 3 B4B328 00962F 3 B20695 B4A220 3 00FFFF FFFFFFF
0006 (01,0002) 7 2F9600 20A003 PCW: LL=02, D[1] SEGMENT @ 0003:0002:3, NORML STATE
0005 (01,0001) 0 000000 000000
0004 ----D[01]=>3 C12000 804003 *MSCW: PREVIOUS MSCW @ 0001; D[00]=0008 IN STACK 012

0003      0 000032 000000 OP:   OCT:00000062 00000000 , EBC:?????, DEC:838860800
0002      0 00FFFF F020D2 OP:   OCT:00177777 74020322 , EBC:???0?K, DEC:4398038189710.0
0001      3 6F9000 000000 *MSCW: DUMMY FOR STACK 2F9
0000      3 000000 D84009 TSCW: S @ 000D, F @ 0004, LL=01

```

0000 = BOSR (0003EE00)

DUMP OF PROCESS INFORMATION BLOCK (TASK VARIABLE)

```

0(0000) 0 000000 000075 0 000000 000001 0 000000 0002F9 0 000000 240082 0 400000 93951D 0 000000 000040
6(0006) 0 000000 000012 0 000000 000000 0 000000 0000AC 0 000000 000000 0 000000 000001 0 000000 000000

```

C-10

A SERIES AND B 5000/B 6000/B 7000 CONCEPTS  
COMPILE LISTINGS AND PROGRAM DUMPS



```

*** SAMPLE PAYROLL PROGRAM FROM SECTION 9, CONVERTED TO COBOL74 **
$ SET LIST MAP CODE
IDENTIFICATION DIVISION.
ENVIRONMENT DIVISION.
INPUT-OUTPUT SECTION.
FILE-CONTROL.
    SELECT EMPLOYEES ASSIGN TO DISK.
DATA DIVISION.
FILE SECTION.
FD EMPLOYEES
    VALUE OF TITLE IS "EMPLOYEES/ACTIVE".
01 EMPLOYEE-RECORD.
    03 FILLER          PIC X(180).
WORKING-STORAGE SECTION.
77 HOURS-WORK        PIC 9(5)V99      COMP.
77 HOURLY-RATE       PIC 9(5)V99      COMP.
77 GROSS-PAY        PIC 9(5)V99      COMP.
77 NET-PAY          PIC 9(5)V99      COMP.
77 TOTAL-TAXES     PIC 9(5)V99      COMP.
77 REG-PAY         PIC 9(5)V99      COMP.
77 OT-PAY          PIC 9(5)V99      COMP.
77 FICA-TAX        PIC 9(5)V99      COMP.
77 FED-TAX         PIC 9(5)V99      COMP.
01 TAX-TABLE.
    03 TAX-RATE      PIC 99V99      OCCURS 10 TIMES.
PROCEDURE DIVISION.
100-MAIN-LOGIC.
* OPEN EMPLOYEE FILE.
* READ AN EMPLOYEE, AND MOVE VALUES INTO VARIABLES.
  MOVE 45 TO HOURS-WORK.
  PERFORM 200-CALC-CHECK.
* PRINT CHECK.
* CLOSE EMPLOYEE FILE.
  STOP RUN.
200-CALC-CHECK.
  COMPUTE REG-PAY = HOURS-WORK * HOURLY-RATE.
  IF HOURS-WORK GREATER THAN 40 THEN PERFORM 300-CALC-OT.
  COMPUTE GROSS-PAY = REG-PAY + OT-PAY.
  PERFORM 400-CALC-TAXES.
  COMPUTE NET-PAY = GROSS-PAY - TOTAL-TAXES.
300-CALC-OT.
  COMPUTE OT-PAY = HOURLY-RATE * .5 * (HOURS-WORK - 40).
  CALL SYSTEM DUMP.
400-CALC-TAXES.
* CALCULATE FICA-TAX AND FED-TAX.
  COMPUTE TOTAL-TAXES = FICA-TAX + FED-TAX.
  CALL SYSTEM DUMP.

```

000020  
000050  
000100  
000300  
000400  
000500  
000600  
000700  
000800  
000900  
001000  
001100  
001200  
001500  
001520  
001540  
001560  
001580  
001600  
001620  
001640  
001660  
001680  
001850  
001860  
001900  
002000  
002100  
002150  
002170  
002200  
002300  
002350  
002400  
002500  
002600  
002700  
002800  
002820  
002840  
002900  
003000  
003100  
003400  
003500  
003600  
003700

(CONCEPTS)OBJECT/EP4195/PAYROLL/C74/OT ON SYSTEMSED

000020\*\*\* SAMPLE PAYROLL PROGRAM FROM SECTION 9, CONVERTED TO COBOL74 \*\*  
000100 IDENTIFICATION DIVISION.

MACRO 0001:06:KBLSM 00AA 0000

000300 ENVIRONMENT DIVISION.  
000400 INPUT-OUTPUT SECTION.  
000500 FILE-CONTROL.  
000600 SELECT EMPLOYEES ASSIGN TO DISK.  
000700 DATA DIVISION.

MACRO 0001:11:SWMCO 0057 0001

000800 FILE SECTION.  
000900 FD EMPLOYEES  
001000 VALUE OF TITLE IS "EMPLOYEES/ACTIVE".

MACRO 0001:18:FLDEC 0001 0002

001100 01 EMPLOYEE-RECORD.  
001200 03 FILLER

PIC X(180).

MACRO 0001:1D:LOCRA 0003 0003

001500 WORKING-STORAGE SECTION.

MACRO 0001:20:FLDEC 0073 0002

001520 77 HOURS-WORK

PIC 9(5)V99

MACRO 0001:21:SWMCO 0001 0000  
COMP.  
MACRO 0001:24:DCL77 0004 0000

001540 77 HOURLY-RATE

PIC 9(5)V99

MACRO 0001:27:DCL77 0005 0000  
COMP.

001560 77 GROSS-PAY

PIC 9(5)V99

MACRO 0001:2A:DCL77 0006 0000  
COMP.

001580 77 NET-PAY

PIC 9(5)V99

MACRO 0001:2D:DCL77 0007 0000  
COMP.

001600 77 TOTAL-TAXES

PIC 9(5)V99

MACRO 0001:30:DCL77 0008 0000  
COMP.

001620 77 REG-PAY

PIC 9(5)V99

MACRO 0001:33:DCL77 0009 0000  
COMP.

001640 77 OT-PAY

PIC 9(5)V99

MACRO 0001:36:DCL77 000A 0000  
COMP.

001660 77 FICA-TAX

PIC 9(5)V99

MACRO 0001:39:DCL77 000B 0000  
COMP.

001680 77 FED-TAX

PIC 9(5)V99

MACRO 0001:3C:DCL77 000C 0000  
COMP.

001850 01 TAX-TABLE.  
001860 03 TAX-RATE

PIC 99V99

MACRO 0001:41:LOCRA 000D 000D  
OCCURS 10 TIMES.

0000:0000:0  
0000:0000:0  
START OF SEGMENT AT (01,002)  
0002:0000:0  
0002:0000:0  
0002:0000:0  
0002:0000:0  
0002:0000:0  
0002:0000:0  
0002:0000:0  
0002:0000:0  
SIRW TO D[01] = (02,003)  
MYUSE VALUE = (02,005)  
EMPLOYEES = (02,004)  
0002:0000:0  
0002:0000:0

EMPLOYEE-RECORD = (02,006)  
0002:0000:0

EMPLOYEES(MAXRECSIZE) = 180  
EMPLOYEES(INTMODE) = EBCDIC

0002:0000:0

HOURS-WORK = (02,013)  
0002:0000:0

0002:0000:0

GROSS-PAY = (02,00B)  
0002:0000:0

NET-PAY = (02,00C)  
0002:0000:0

TOTAL-TAXES = (02,00D)  
0002:0000:0

REG-PAY = (02,00E)  
0002:0000:0

OT-PAY = (02,00F)  
0002:0000:0

0002:0000:0

0002:0000:0

0000(0000:0)

G-12

A SERIES AND B 5000/B 6000/B 7000 CONCEPTS  
COMPILE LISTINGS AND PROGRAM DUMPS

```

001900 PROCEDURE DIVISION.
MACRO 0001:44:SWMCO 0057 0000
MACRO 0001:45:KBLSM 0109 0000
002000 100-MAIN-LOGIC.
002100* OPEN EMPLOYEE FILE.
002150* READ AN EMPLOYEE, AND MOVE VALUES INTO VARIABLES.
MACRO 0001:4C:DEFLM 0000 0000
0002:0000:0 NOOP FE
002170 MOVE 45 TO HOURS-WORK.
MACRO 0001:4D:DEFLM 000F 0000
MACRO 0001:50:SVNLT 0000 0000
MACRO 0001:51:LITCM 002D 0800
MACRO 0001:52:FIXMM 0000 0000
MACRO 0001:53:MVNLT 0000 0004
0002:0000:1 LT8 B205 5.0
0002:0000:1 NAMC (02,0009) 5009
0002:0000:3 LT16 B34500 17664.0
0002:0001:0 STOD B8
MACRO 0001:54:SNTNC 040E 0000
002200 PERFORM 200-CALC-CHECK.
002300* PRINT CHECK.
002350* CLOSE EMPLOYEE FILE.
MACRO 0001:5B:MHPWM 000F 0000
MACRO 0001:5C:PFTMM 000F 0000
MACRO 0001:5D:PFRMM 0000 0000
MACRO 0001:5E:XXXXM 0000 0000
MACRO 0001:5F:XXXXM 000F 0000
0002:0001:1 MPCW BF00000040E002
0002:0003:0 LT16 B30002 2.0
0002:0003:3 BRUN 0000:0 *LINK*
MACRO 0001:60:XXXXM 00A3 0000
MACRO 0001:61:SNTNC 0321 0000
002400 STOP RUN.
MACRO 0001:64:STOPM 0134 0000
0002:0004:0 MKST AE
MCP PROCEDURE: GOTOSOLVER = (01,003)
0002:0004:1 NAMC (01,0003) 6003 MCP.GOTOSOLVER
0002:0004:3 ZERO B0
0002:0004:4 ENTR AB
MACRO 0001:65:SNTNC 000F 0000
002500 200-CALC-CHECK.
MACRO 0001:68:ENDLM 000F 0000
MACRO 0001:69:DEFLM 0010 0000
* 0002:0003:3 BRUN 0004:5 A2A004
002600 COMPUTE REG-PAY = HOURS-WORK * HOURLY-RATE.
MACRO 0001:6C:OPERM 0009 0000
0002:0004:5 NAMC (02,000E) 500E REG-PAY
MACRO 0001:6D:FIXMM 0000 0000
MACRO 0001:6E:SWMCO 0047 0001
MACRO 0001:6F:EXPRM 0000 0000
MACRO 0001:70:TEIDM 0004 0000
MACRO 0001:71:TEIDM 0005 0000
MACRO 0001:72:TEOPM 0082 0000
MACRO 0001:73:ENDXM 0000 0000
MACRO 0001:74:TEVAL 0000 0000
0002:0004:5 ZERO B0

```

TAX-TABLE = (02,014)  
0002:0000:0

0002:0000:0  
0002:0000:0  
0002:0000:0

PCW = (02,015)  
PCW = (02,016)  
LIBRARY DIRECTORY = (02,017)  
PCW(002:000:1) = (02,018)

0002:0000:1

0002:0001:1  
0002:0001:1  
0002:0001:1

0002:0004:0

0002:0004:5

0002:0004:5

C-14

```

0002:0004:5 ZERO B0
0002:0005:0 NAMC (02,000E) 500E REG-PAY
0002:0005:2 STOD B8
002700 IF HOURS-WORK GREATER THAN 40 THEN PERFORM 300-CALC-OT. 0002:0005:3
MACRO 0001:75:SNTNC 043F 0000
MACRO 0001:78:EXPRM 0001 0000
MACRO 0001:79:SWMCO 00AB 0001
MACRO 0001:7A:OPERM 0004 0001
0002:0005:3 LT8 B205 5.0
0002:0005:5 NAMC (02,0013) 5013 HOURS-WORK
0002:0006:1 INDX A6
0002:0006:2 LT8 B207 7.0
0002:0006:4 ICVD CA
MACRO 0001:7B:LITCM 0004 0401
0002:0006:5 LT8 B228 40.0
MACRO 0001:7C:RELAM 008A 0001
0002:0007:1 LT8 B264 100.0
0002:0006:5 LT16 B30FAO 4000.0
0002:0007:2 GRTR 8A
MACRO 0001:7D:BFIXM 0000 0000
0002:0007:3 FIXUP
MACRO 0001:7E:ENDXM 0001 0000
WARNING 435 : BURROUGHS EXTENSION EXCEEDS U.S. HIGH LEVEL *** THEN <<0001>>
MACRO 0001:82:DEFPM 0001 0000
MACRO 0001:83:MHPWM 0010 0000
MACRO 0001:84:PFTMM 0010 0000
MACRO 0001:85:PFRMM 0000 0000
MACRO 0001:86:XXXXM 0000 0000
MACRO 0001:87:XXXXM 0010 0000
0002:0008:0 MPCW BF000000B0E002
0002:000A:0 LT16 B30003 3.0
0002:000A:3 BRUN 0000:0 *LINK*
MACRO 0001:88:XXXXM 00A3 0000
MACRO 0001:89:BRFPM 0001 0000
* 0002:0007:3 BRFL 000B:0 A0000B
MACRO 0001:8A:SNTNC 0415 0000
002800 COMPUTE GROSS-PAY = REG-PAY + OT-PAY. 0002:0008:0
MACRO 0001:8D:OPERM 0006 0000
0002:000B:0 NAMC (02,000B) 500B GROSS-PAY
MACRO 0001:8E:FIXMM 0000 0000
MACRO 0001:8F:SWMCO 0047 0001
MACRO 0001:90:EXPRM 0000 0000
MACRO 0001:91:TEIDM 0009 0000
MACRO 0001:92:TEIDM 000A 0000
MACRO 0001:93:TEOPM 0080 0000
MACRO 0001:94:ENDXM 0000 0000
MACRO 0001:95:TEVAL 0000 0000
0002:0008:0 VALC (02,000E) 100E REG-PAY
0002:0008:2 VALC (02,000F) 100F OT-PAY
0002:0008:4 ADD 80
0002:0008:5 BRST 9E2E
0002:000C:1 LT48 BE000000989680 1.0E+7
0002:000E:0 RDIV 85
0002:000E:1 NTIA 86
0002:000E:2 NAMC (02,000B) 500B GROSS-PAY
0002:000E:4 STOD B8
MACRO 0001:96:SNTNC 040E 0000
002820 PERFORM 400-CALC-TAXES. 0002:000E:5
MACRO 0001:99:MHPWM 0011 0000
MACRO 0001:9A:PFTMM 0011 0000
MACRO 0001:9B:PFRMM 0000 0000

```

```

MACRO 0001:9C:XXXXM 0000 0000
MACRO 0001:9D:XXXXM 0011 0000
0002:000E:5 MPCW BF00000110E002
0002:0010:0 LT16 B30004 4.0
0002:0010:3 BRUN 0000:0 *LINK*
MACRO 0001:9E:XXXXM 00A3 0000
MACRO 0001:9F:SNTNC 0415 0000
002840 COMPUTE NET-PAY = GROSS-PAY - TOTAL-TAXES. 0002:0011:0
MACRO 0001:A2:OPERM 0007 0000
0002:0011:0 NAMC (02,000C) 500C NET-PAY
MACRO 0001:A3:FIXMM 0000 0000
MACRO 0001:A4:SWMCO 0047 0001
MACRO 0001:A5:EXPRM 0000 0000
MACRO 0001:A6:TEIDM 0006 0000
MACRO 0001:A7:TEIDM 0008 0000
MACRO 0001:A8:TEOPM 0081 0000
MACRO 0001:A9:ENDXM 0000 0000
MACRO 0001:AA:TEVAL 0000 0000
0002:0011:0 VALC (02,000B) 100B GROSS-PAY
0002:0011:2 VALC (02,000D) 100D TOTAL-TAXES
0002:0011:4 SUBT 81
0002:0011:5 BRST 9E2E
0002:0012:1 LT48 BE000000989680 1.0E+7
0002:0014:0 RDIV 85
0002:0014:1 NTIA 86
0002:0014:2 NAMC (02,000C) 500C NET-PAY
0002:0014:4 STOD B8
MACRO 0001:AB:SNTNC 0010 0000
002900 300-CALC-OT. 0002:0014:5
MACRO 0001:AE:ENDLM 0010 0000
0002:0014:5 DUPL B7
0002:0015:0 LT8 B202 2.0
0002:0015:2 SAME 94
0002:0015:3 BRFL 0016:2 A04016
0002:0016:0 DLET B5
0002:0016:1 DBUN AA
MACRO 0001:AF:DEFLM 0011 0000
* 0002:000A:3 BRUN 0016:2 A24016
003000 COMPUTE OT-PAY = HOURLY-RATE * .5 * (HOURS-WORK - 40). 0002:0016:2
MACRO 0001:B2:OPERM 000A 0000
0002:0016:2 NAMC (02,000F) 500F OT-PAY
MACRO 0001:B3:FIXMM 0000 0000
MACRO 0001:B4:SWMCO 0047 0001
MACRO 0001:B5:EXPRM 0000 0000
MACRO 0001:B6:TEIDM 0005 0000
MACRO 0001:B7:TECON 0000 0000
MACRO 0001:B8:LITCM 0005 0420
MACRO 0001:B9:TEOPM 0082 0000
MACRO 0001:BA:EXPRM 0000 0000
MACRO 0001:BB:TEIDM 0004 0000
MACRO 0001:BC:TECON 0000 0000
MACRO 0001:BD:LITCM 0004 0401
MACRO 0001:BE:TEOPM 0081 0000
MACRO 0001:BF:ENDXM 0000 0000
MACRO 0001:CO:TEOPM 0082 0000
MACRO 0001:C1:ENDXM 0000 0000
MACRO 0001:C2:TEVAL 0000 0000
0002:0016:2 LT8 B205 5.0
0002:0016:4 NAMC (02,0013) 5013 HOURS-WORK
0002:0017:0 INDX A6
0002:0017:1 LT8 B207 7.0

```

```
0002:0017:3 ICVD CA
0002:0017:4 ZERO B0
0002:0017:5 MULT 82
0002:0018:0 ZERO B0
0002:0018:1 ADD 80
0002:0018:2 LT8 B20A 10.0
0002:0018:4 IDIV 84
0002:0018:5 BRST 9E2E
0002:0019:1 LT48 BE000000989680 1.0E+7
0002:001B:0 RDIV 85
0002:001B:1 NTIA 86
0002:001B:2 NAMC (02,000F) 500F OT-PAY
0002:001B:4 STOD B8
MACRO 0001:C3:SNTNC 0047 0000
003100 CALL SYSTEM DUMP. 0002:001B:5
WARNING 435 : BURROUGHS EXTENSION EXCEEDS U.S. HIGH LEVEL *** SYSTEM 002700 <<0002>>
MACRO 0001:C9:PROCM 0000 010C
0002:001B:5 MKST AE MCP PROCEDURE: PROGRAMDUMP = (01,004)
0002:001C:0 NAMC (01,0004) 6004 MCP.PROGRAMDUMP
0002:001C:2 LT8 B202 2.0
0002:001C:4 ENTR AB
0002:001C:5 NOOP FE
MACRO 0001:CA:SNTNC 0011 0000
003400 400-CALC-TAXES.
003500* CALCULATE FICA-TAX AND FED-TAX.
MACRO 0001:CF:ENDLM 0011 0000
0002:001D:0 DUPL B7
0002:001D:1 LT8 B203 3.0
0002:001D:3 SAME 94
0002:001D:4 BRFL 001E:3 A0601E
0002:001E:1 DLET B5
0002:001E:2 DBUN AA
MACRO 0001:D0:DEFLM 0012 0000
* 0002:0010:3 BRUN 001E:3 A2601E
003600 COMPUTE TOTAL-TAXES = FICA-TAX + FED-TAX. 0002:001E:3
MACRO 0001:D3:OPERM 0008 0000
0002:001E:3 NAMC (02,000D) 500D TOTAL-TAXES
MACRO 0001:D4:FIXMM 0000 0000
MACRO 0001:D5:SWMCO 0047 0001
MACRO 0001:D6:EXPRM 0000 0000
MACRO 0001:D7:TEIDM 000B 0000
MACRO 0001:D8:TEIDM 000C 0000
MACRO 0001:D9:TEOPM 0080 0000
MACRO 0001:DA:ENDXM 0000 0000
MACRO 0001:DB:TEVAL 0000 0000
0002:001E:3 ZERO B0
0002:001E:3 ZERO B0
0002:001E:4 NAMC (02,000D) 500D TOTAL-TAXES
0002:001F:0 STOD B8
MACRO 0001:DC:SNTNC 0047 0000
003700 CALL SYSTEM DUMP.
WARNING 435 : BURROUGHS EXTENSION EXCEEDS U.S. HIGH LEVEL *** SYSTEM
MACRO 0001:E2:PROCM 0000 010C
0002:001F:1 MKST AE
0002:001F:2 NAMC (01,0004) 6004 MCP.PROGRAMDUMP
0002:001F:4 LT8 B202 2.0
0002:0020:0 ENTR AB
0002:0020:1 NOOP FE
MACRO 0001:E3:SNTNC 0000 0000
MACRO 0001:E4:ENDLM 0012 0000
0002:001F:1
003100 <<0003>>
```

```

0002:0020:2 DUPL          B7
0002:0020:3 LT8          B204          4.0
0002:0020:5 SAME          94
0002:0021:0 BRFL      0021:5 A0A021
0002:0021:3 DLET          B5
0002:0021:4 DBUN          AA
                MACRO 0001:E5:ENDLM 0000 0000
0002:0021:5 NVLD          FF
                                SEGMENT 0002 IS 0022 LONG
                                MACRO 0001:E6:THEEND 0000 0000
                                START OF SEGMENT AT (01,005)
                                PCW(005:000:0) = (02,015)

0005:0000:0 LT8          B235          53.0
0005:0000:2 RPRR          9588
                                LIBRARY USER = (02,019)

0005:0000:4 NAMC (02,0019) 5019
0005:0001:0 LOAD          BD
0005:0001:1 EQU          8C
0005:0001:2 FIXUP
0005:0001:5 NVLD          FF
* 0005:0001:2 BRFL      0002:0 A00002
0005:0002:0 MKST          AE
                                MCP PROCEDURE: MUTATE = (01,006)

0005:0002:1 NAMC (01,0006) 6006 MCP.MUTATE
0005:0002:3 NAMC (02,0002) 5002
0005:0002:5 LOAD          BD
0005:0003:0 LT8          B20C          12.0
0005:0003:2 LT8          B206          6.0
0005:0003:4 ENTR          AB
0005:0003:5 MKST          AE
                                MCP PROCEDURE: BLOCKEXIT = (01,007)

0005:0004:0 NAMC (01,0007) 6007 MCP.BLOCKEXIT
0005:0004:2 ENTR          AB
0005:0004:3 EXIT          A3
                                PCW(005:004:4) = (02,016)
                                LIBRARY LOCK = (02,01A)
                                LIBRARY EVENT = (02,01B)

0005:0004:4 LT8          B235          53.0
0005:0005:0 RPRR          9588
0005:0005:2 NAMC (02,001A) 501A
0005:0005:4 RDLK          958A
0005:0006:0 ZERO          B0
0005:0006:1 EQU          8C
0005:0006:2 FIXUP
0005:0006:5 MKST          AE
                                MCP PROCEDURE: WAIT = (01,008)

0005:0007:0 NAMC (01,0008) 6008 MCP.WAIT
0005:0007:2 NAMC (02,001B) 501B
0005:0007:4 STFF          AF
0005:0007:5 ONE          B1
0005:0008:0 ENTR          AB
0005:0008:1 ONE          B1
0005:0008:2 NAMC (02,001A) 501A
0005:0008:4 RDLK          958A
0005:0009:0 ZERO          B0
0005:0009:1 EQU          8C
0005:0009:2 BRFL      0006:5 A0A006
* 0005:0006:2 BRTR      0009:5 A1A009
0005:0009:5 LT8          B235          53.0
0005:000A:1 RPRR          9588
0005:000A:3 NAMC (02,0019) 5019

```

C-18

```

0005:000A:5 STOD B8
0005:000B:0 FIXUP
0005:000B:3 MKST AE
0005:000B:4 NAMC (01,0007) 6007 MCP.BLOCKEXIT
0005:000C:0 ENTR AB
0005:000C:1 EXIT A3
0005:000C:2 ZERO B0
0005:000C:3 NAMC (02,0019) 5019
0005:000C:5 STOD B8
0005:000D:0 ZERO B0
0005:000D:1 NAMC (02,001A) 501A
0005:000D:3 RDLK 95BA
0005:000D:5 LT8 B235 53.0
0005:000E:1 RPRR 95B8
0005:000E:3 EQU L 8C
0005:000E:4 FIXUP
0005:000F:1 MKST AE
MCP PROCEDURE: CAUSEP = (01,009)
0005:000F:2 NAMC (01,0009) 6009 MCP.CAUSEP
0005:000F:4 NAMC (02,001B) 501B
0005:0010:0 STFF AF
0005:0010:1 ONE B1
0005:0010:2 ENTR AB
* 0005:000E:4 BRTR 0010:3 A16010
0005:0010:3 ZERO B0
0005:0010:4 RETN A7
* 0005:0008:0 BRUN 0010:5 A2A010
0005:0010:5 MPCW BF000600B0E005
0005:0012:0 NAMC (03,0002) 7002
0005:0012:2 STFF AF
LIBRARY EXIT PCW = (02,01D)
0005:0012:3 NAMC (02,001D) 501D
0005:0012:5 OVRD BA
0005:0013:0 MPCW BF000400C12005
0005:0015:0 LT48 BE800000100000 1048576.0
0005:0017:0 LT8 B206 6.0
0005:0017:2 STAG 95B4
0005:0017:4 ZERO B0
0005:0017:5 NAMC (02,0018) 5018
0005:0018:1 DBUN AA
SEGMENT 0005 IS 0019 LONG
START OF SEGMENT AT (01,00A)
LIBRARY FIRST EXECUTABLE PCW(00A:008:2) = (01,00B)
000A:0008:2 MKST AE
MCP PROCEDURE: MYSELF = (01,00C)
000A:0008:3 NAMC (01,000C) 600C MCP.MYSELF
000A:0008:5 ENTR AB
000A:0009:0 FIXUP
NORMAL FIRST EXECUTABLE PCW(00A:009:3) = (01,00D)
000A:0009:3 ZERO B0
* 000A:0009:0 BRUN 0009:4 A28009
STKPARAM = (02,002)
000A:0009:4 NAMC (01,0000) 6000
000A:000A:0 STFF AF
000A:000A:1 BSET 960D
STKNAME = (02,003)
000A:000A:3 LT48 BE270000B40004 2.68220992439E-6
000A:000C:0 LT8 B205 5.0
000A:000C:2 STAG 95B4
STKFILE = (02,004)
000A:000C:4 ZERO B0

```



000A:001D:4	MPCW	BF00020000E002		STKPCW = (02,018)
000A:001F:0	ZERO	B0		TEMPORARY = (02,019)
000A:001F:1	ZERO	B0		STKOP = (02,01A)
000A:001F:2	ZERO	B0		STKOP = (02,01B)
000A:001F:3	ZERO	B0		
000A:001F:4	JOIN	9542		STKOP = (02,01C)
000A:0020:0	ZERO	B0		TEMPORARY = (02,01D)
000A:0020:1	LT16	B32800	10240.0	
000A:0020:4	BSET	962F		
000A:0021:0	LT8	B206	6.0	
000A:0021:2	STAG	95B4		
000A:0021:4	PUSH	B4		
000A:0021:5	BRUN	0000:0 A20000		
000A:0000:0	NAMC (02,0002)	5002		LIBRARY VARIABLE = (02,002)
000A:0000:2	LOAD	BD		
000A:0000:3	ZERO	B0		
000A:0000:4	SAME	94		
000A:0000:5	FIXUP			
000A:0001:2	NAMC (02,0015)	5015		
000A:0001:4	STFF	AF		
000A:0001:5	LT8	B202	2.0	
000A:0002:1	NAMC (02,0017)	5017		
000A:0002:3	INDX	A6		
000A:0002:4	OVRD	BA		
000A:0002:5	NAMC (02,0016)	5016		
000A:0003:1	STFF	AF		
000A:0003:2	LT8	B204	4.0	
000A:0003:4	NAMC (02,0017)	5017		
000A:0004:0	INDX	A6		
000A:0004:1	OVRD	BA		
000A:0004:2	MKST	AE		
				MCP PROCEDURE: FREEZELIB = (01,00F)
000A:0004:3	NAMC (01,000F)	600F	MCP.FREEZELIB	
000A:0004:5	ONE	B1		
000A:0005:0	NAMC (02,0017)	5017		
000A:0005:2	LOAD	BD		
000A:0005:3	NAMC (02,0000)	5000		
000A:0005:5	STFF	AF		
000A:0006:0	ENTR	AB		
000A:0006:1	FIXUP			
* 000A:0000:5	BRTR	0006:4 A18006		
000A:0006:4	MKST	AE		
000A:0006:5	NAMC (02,0018)	5018		
000A:0007:1	ZERO	B0		
000A:0007:2	ENTR	AB		
* 000A:0006:1	BRUN	0007:3 A26007		
000A:0007:3	MKST	AE		
000A:0007:4	NAMC (01,0007)	6007	MCP.BLOCKEXIT	
000A:0008:0	ENTR	AB		
000A:0008:1	EXIT	A3		

SEGMENT 000A IS 0023 LONG

COMPILER O.K.  
 NUMBER OF WARNINGS DETECTED = 0003  
 LAST WARNING AT 003700

000A:000C:5	LT48	BE04000B400000	3.166593488E+15	TEMPORARY = (02,005)
000A:000E:0	LT8	B205	5.0	
000A:000E:2	STAG	95B4		STKARRAY = (02,006)
000A:000E:4	ZERO	B0		
000A:000E:5	LT8	B209	9.0	
000A:000F:1	INSR	9C2714		
000A:000F:4	LT8	B205	5.0	
000A:0010:0	STAG	95B4		STKARRAY = (02,007)
000A:0010:2	ZERO	B0		TEMPORARY = (02,008)
000A:0010:3	ZERO	B0		TEMPORARY = (02,009)
000A:0010:4	ZERO	B0		STKOP = (02,00A)
000A:0010:5	ZERO	B0		STKOP = (02,00B)
000A:0011:0	ZERO	B0		STKOP = (02,00C)
000A:0011:1	ZERO	B0		STKOP = (02,00D)
000A:0011:2	ZERO	B0		STKOP = (02,00E)
000A:0011:3	ZERO	B0		STKOP = (02,00F)
000A:0011:4	ZERO	B0		STKOP = (02,010)
000A:0011:5	ZERO	B0		STKOP = (02,011)
000A:0012:0	ZERO	B0		
000A:0012:1	MKST	AE		MCP PROCEDURE: INSTACKARRAYDEC = (01,00E)
000A:0012:2	NAMC (01,000E)	600E	MCP.INSTACKARRAYDEC	
000A:0012:4	NAMC (02,0007)	5007		
000A:0013:0	STFF	AF		
000A:0013:1	NAMC (02,0009)	5009		
000A:0013:3	STFF	AF		
000A:0013:4	ONE	B1		
000A:0013:5	ENTR	AB		= (02,012)
000A:0014:0	PUSH	B4		
000A:0014:1	NAMC (02,0007)	5007		
000A:0014:3	LOAD	B0		
000A:0014:4	LT8	B26C	2097260.0	
000A:0015:0	BSET	9615		
000A:0015:2	INSR	9C2A17		STKARRAY = (02,013)
000A:0015:5	LT48	BE040002A00000	7.3887181386E+14	
000A:0017:0	LT8	B205	5.0	
000A:0017:2	STAG	95B4		STKARRAY = (02,014)
000A:0017:4	MPCW	BF00C00000E005		STKPCW = (02,015)
000A:0019:0	MPCW	BF00C80040E005		STKPCW = (02,016)
000A:0018:0	LT48	BE00C001040001	17039361.0	
000A:001D:0	LT8	B205	5.0	
000A:001D:2	STAG	95B4		STKARRAY = (02,017)

TOTAL CARD COUNT: 46  
D[01] STACK SIZE: 0016(010) WORDS  
D[02] STACK SIZE: 0030(01E) WORDS  
CORE ESTIMATE: 711 WORDS  
STACK ESTIMATE: 310 WORDS  
CODE FILE SIZE: 10 RECORDS  
PROGRAM SIZE: 3 CODE SEGMENTS, 94 TOTAL WORDS  
SUBROUTINE NAME: OT, LEVEL 02  
COMPILED ON THE B6900 FOR THE LEVELO SERIES  
COMPILER COMPILED WITH THE FOLLOWING OPTIONS:  
BDMS.  
COMPILE TIMES: ELAPSED CPU I-O RPM  
0020.020 0002.072 0002.269 01332

MCP 36.140.3025: \*SYSTEM/MCP36140. INTRINSICS: SYSTEM/INTRINSICS ON DISK. (LOADED)  
SYSTEM SERIAL: #2372 HOSTNAME: SYSEDB6900. GROUP ID: DEFAULT.

CAUSE OF DUMP: PROGRAM REQUESTED @ 002:001C:5, 00A:0007:3.

PIB HISTORY WORD AT ENTRY TO PROGRAM DUMP: 0 000000 000000.

PROGRAMDUMP OPTIONS: (DEFAULT)

033A = LOSR (0005BD4A)

```
003C (01,0002) 0 000000 000001 OP: OCT:00000000 00000001 , EBC:??????, DEC:1
003B          3 200A01 COE002 RCW: LL=03, NORML STATE, TRUE [USER SEGMENT @ 0002:001C:5]
          SEG DESC: 3 800002 2AE8C3
          CODE: 3 2EBEFF FFFFFFF 3 000000 989680 3 858650 0FB8AE >3 6004B2 02ABFE< 3 B7B203 94A060
003A ----D[01]=>3 C12000 804007 *MSCW: PREVIOUS MSCW @ 0033; D[00]=0008 IN STACK 012
0039 (03,0006) 0 000000 000003 OP: OCT:00000000 00000003 , EBC:??????, DEC:3
0038 (03,0005) 7 2FE000 BOE002 PCW: LL=03, D[1] SEGMENT @ 0002:000B:0, NORML STATE
0037 (03,0004) 0 000000 000002 OP: OCT:00000000 00000002 , EBC:??????, DEC:2
0036 (03,0003) 7 2FE000 40E002 PCW: LL=03, D[1] SEGMENT @ 0002:0004:0, NORML STATE
0035 (03,0002) 0 000000 000000
0034          3 000600 70A00A RCW: LL=02, NORML STATE [USER SEGMENT @ 000A:0007:3]
          SEG DESC: 3 800002 3B3001
          CODE: 3 A6BAAE 600FB1 3 5017BD 5000AF 3 ABA260 07AE50 >3 18B0AB AE6007< 3 ABA3AE 600CAB
0033 ----D[03]=>3 6FE001 40C01F *MSCW: PREVIOUS MSCW @ 0014; D[02]=0014
0032 (02,001E) 6 800000 002800 SCW: (BLOCK BELOW DECLARED FILES, SNGL-DIM ARRAYS)
0031 (02,001D) 0 000000 000000
0030 (02,001C) 2 000000 000000 DPOP: OCT:00000000 00000000 , 2ND:0
002F (02,001B) 2 000000 000000 DPOP: OCT:00000000 00000000 , 1ST:0, DBL:0.0
002E (02,001A) 0 000000 000000
002D (02,0019) 0 000000 000000
002C (02,0018) 7 2FE200 00E002 PCW: LL=03, D[1] SEGMENT @ 0002:0000:1, NORML STATE
002B (02,0017) 5 000001 040001 DESC [ABSENT-MOM]: DATA, LENGTH=16 (CODEFILE ADRS=1)
002A (02,0016) 7 2FE800 40E005 PCW: LL=03, D[1] SEGMENT @ 0005:0004:4, NORML STATE
0029 (02,0015) 7 2FE000 00E005 PCW: LL=03, D[1] SEGMENT @ 0005:0000:0, NORML STATE
0028 (02,0014) 5 040002 A00000 DESC [ABSENT-MOM]: STRING (8-BIT), LENGTH=42 (UNREFERENCED OLAY SPACE)
0027 (02,0013) 5 C20006 C5BA2D DESC [PRESENT-COPY]: STRING (4-BIT), LENGTH=108 (POINTS @ OFFSET=001D IN THIS STACK)
0026 (02,0012) 3 000000 000000 SEG DESC [ABSENT-MOM]: LENGTH=0 (CODEFILE ADRS=0)
0025 (02,0011) 0 000000 000000
0024 (02,0010) 0 000000 000000
0023 (02,000F) 0 000000 000000
0022 (02,000E) 0 000000 000000
0021 (02,000D) 0 000000 000000
0020 (02,000C) 0 000000 000000
001F (02,000B) 0 000000 000000
001E (02,000A) 0 000000 000000
001D (02,0009) 0 000000 004500 OP: OCT:00000000 00042400 , EBC:??????, DEC:17664
001C (02,0008) 5 400000 15BA2B DESC [ABSENT-COPY]: DATA, LENGTH=1 (MOM @ OFFSET=001B IN THIS STACK)
001B (02,0007) 5 C00000 95BA2D DESC [PRESENT-COPY]: DATA, LENGTH=9 (POINTS @ OFFSET=001D IN THIS STACK)
001A (02,0006) 5 04000B 400000 DESC [ABSENT-MOM]: STRING (8-BIT), LENGTH=180 (UNREFERENCED OLAY SPACE)
0019 (02,0005) 0 000000 000000
0018 (02,0004) 5 270000 B40004 DESC [ABSENT-MOM]: FILE DESCRIPTION, LENGTH=11 (CODEFILE ADRS=4)
0017 (02,0003) 1 6FF000 402000 SIRW: OFFSET=0004 (0004+0000) IN STACK 2FF
0016 (02,0002) 0 000000 000000
0015          3 00024F E88FCD RCW: LL=02, CNTRL STATE [MCP SEGMENT @ 0FCD:04EE:1 (16562000)]
```

C-22

A SERIES AND B 5000/B 6000/B 7000 CONCEPTS  
COMPILE LISTINGS AND PROGRAM DUMPS

```
SEG DESC: 3 8800B6 ABCAC1
CODE: 3 BEFFFF FFFFFF 3 4A6870 C30C30 3 ABA3AE 400AAB >3 A3B234 B22695< 3 B8B180 95B9A2
0014 ----D[02]=>3 EFF000 408002 *MSCW: PREVIOUS MSCW @ 0012; D[01]=0004 IN STACK 2FF
0013          3 000000 002000 RCW: DUMMY (RUN)
0012 ----D[02]=>3 F82055 108011 *MSCW: PREVIOUS MSCW @ 0001; D[01]=0551 IN STACK 382
0000 = BOSR (0005BA10)
```

MCP 36.140.3025: \*SYSTEM/MCP36140. INTRINSICS: SYSTEM/INTRINSICS ON DISK. (LOADED)  
SYSTEM SERIAL: #2372 HOSTNAME: SYSEDB6900. GROUP ID: DEFAULT.

CAUSE OF DUMP: PROGRAM REQUESTED @ 002:0020:1, 00A:0007:3.

PIB HISTORY WORD AT ENTRY TO PROGRAM DUMP: 0 000000 000000.

PROGRAMDUMP OPTIONS: (DEFAULT)

033A = LOSR (0005BD4A)

003C (01,0002) 0 000000 000001 OP: OCT:00000000 00000001 , EBC:??????, DEC:1  
003B 3 200202 00E002 RCW: LL=03, NORML STATE, TRUE [USER SEGMENT @ 0002:0020:1]  
SEG DESC: 3 800002 2AE8C3  
CODE: 3 B7B203 94A060 3 1EB5AA B0500D 3 B8AE60 04B202 >3 ABFEB7 B20494< 3 A0A021 B5AAFF  
003A ---D[01]=>3 C12000 804007 \*MSCW: PREVIOUS MSCW @ 0033; D[00]=0008 IN STACK 012

0039 (03,0006) 0 000000 000004 OP: OCT:00000000 00000004 , EBC:??????, DEC:4  
0038 (03,0005) 7 2FE001 10E002 PCW: LL=03, D[1] SEGMENT @ 0002:0011:0, NORML STATE  
0037 (03,0004) 0 000000 000002 OP: OCT:00000000 00000002 , EBC:??????, DEC:2  
0036 (03,0003) 7 2FE000 40E002 PCW: LL=03, D[1] SEGMENT @ 0002:0004:0, NORML STATE  
0035 (03,0002) 0 000000 000000  
0034 3 000600 70A00A RCW: LL=02, NORML STATE [USER SEGMENT @ 000A:0007:3]  
SEG DESC: 3 800002 3B3001  
CODE: 3 A6BAAE 600FB1 3 5017BD 5000AF 3 ABA260 07AE50 >3 18B0AB AE6007< 3 ABA3AE 600CAB

0033 ---D[03]=>3 6FE001 40C01F \*MSCW: PREVIOUS MSCW @ 0014; D[02]=0014

0032 (02,001E) 6 800000 002800 SCW: (BLOCK BELOW DECLARED FILES, SNGL-DIM ARRAYS)  
0031 (02,001D) 0 000000 000000  
0030 (02,001C) 2 000000 000000 DPOP: OCT:00000000 00000000 , 2ND:0  
002F (02,001B) 2 000000 000000 DPOP: OCT:00000000 00000000 , 1ST:0, DBL:0.0  
002E (02,001A) 0 000000 000000  
002D (02,0019) 0 000000 000000  
002C (02,0018) 7 2FE200 00E002 PCW: LL=03, D[1] SEGMENT @ 0002:0000:1, NORML STATE  
002B (02,0017) 5 000001 040001 DESC [ABSENT-MOM]: DATA, LENGTH=16 (CODEFILE ADRS=1)  
002A (02,0016) 7 2FE800 40E005 PCW: LL=03, D[1] SEGMENT @ 0005:0004:4, NORML STATE  
0029 (02,0015) 7 2FE000 00E005 PCW: LL=03, D[1] SEGMENT @ 0005:0000:0, NORML STATE  
0028 (02,0014) 5 040002 A00000 DESC [ABSENT-MOM]: STRING (8-BIT), LENGTH=42 (UNREFERENCED OLAY SPACE)  
0027 (02,0013) 5 C20006 C5BA2D DESC [PRESENT-COPY]: STRING (4-BIT), LENGTH=108 (POINTS @ OFFSET=001D IN THIS STACK)  
0026 (02,0012) 3 000000 000000 SEG DESC [ABSENT-MOM]: LENGTH=0 (CODEFILE ADRS=0)  
0025 (02,0011) 0 000000 000000  
0024 (02,0010) 0 000000 000000  
0023 (02,000F) 0 000000 000000  
0022 (02,000E) 0 000000 000000  
0021 (02,000D) 0 000000 000000  
0020 (02,000C) 0 000000 000000  
001F (02,000B) 0 000000 000000  
001E (02,000A) 0 000000 000000  
001D (02,0009) 0 000000 004500 OP: OCT:00000000 00042400 , EBC:??????, DEC:17664  
001C (02,0008) 5 400000 15BA2B DESC [ABSENT-COPY]: DATA, LENGTH=1 (MOM @ OFFSET=001B IN THIS STACK)  
001B (02,0007) 5 C00000 95BA2D DESC [PRESENT-COPY]: DATA, LENGTH=9 (POINTS @ OFFSET=001D IN THIS STACK)  
001A (02,0006) 5 04000B 400000 DESC [ABSENT-MOM]: STRING (8-BIT), LENGTH=180 (UNREFERENCED OLAY SPACE)  
0019 (02,0005) 0 000000 000000  
0018 (02,0004) 5 270000 B40004 DESC [ABSENT-MOM]: FILE DESCRIPTION, LENGTH=11 (CODEFILE ADRS=4)  
0017 (02,0003) 1 6FF000 402000 SIRW: OFFSET=0004 (0004+0000) IN STACK 2FF  
0016 (02,0002) 0 000000 000000  
0015 3 00024E E88FCD RCW: LL=02, CNTRL STATE [MCP SEGMENT @ 0FCD:04EE:1 (16562000)]

C-24

A SERIES AND B 5000/B 6000/B 7000 CONCEPTS  
COMPILE LISTINGS AND PROGRAM DUMPS

```
SEG DESC: 3 8800B6 ABCAC1
CODE: 3 BEFFFF FFFFFFF 3 4A6870 C30C30 3 ABA3AE 400AAB >3 A3B234 B22695< 3 B8B180 95B9A2
0014 ----D[02]=>3 EFF000 408002 *MSCW: PREVIOUS MSCW @ 0012; D[01]=0004 IN STACK 2FF
0013          3 000000 002000 RCW: DUMMY (RUN)
0012 ----D[02]=>3 F82055 108011 *MSCW: PREVIOUS MSCW @ 0001; D[01]=0551 IN STACK 382
0000 = BOSR (0005BA10)
```

```
%%%%% SAMPLE LIBRARY PROGRAM IN ALGOL %%%%%  
$ SET LIST STACK $  
BEGIN  
  PROCEDURE SQUAREIT (X);  
    REAL X;  
    BEGIN  
      X := X * X;  
      PROGRAMDUMP (LIBRARIES);  
    END;  
  % END OF PROCEDURE SQUAREIT  
  EXPORT SQUAREIT;  
  %%% OUTER BLOCK %%%  
  FREEZE (TEMPORARY);  
END.
```

```
00001300  
00001400  
00001500  
00001600  
00001700  
00001750  
00001800  
00001850  
00001870  
00001900  
00002000  
00002100  
00002200  
00002300
```

A SERIES AND B 5000/B 6000/B 7000 CONCEPTS  
COMPILE LISTINGS AND PROGRAM DUMPS



OBJECT/EP4195/LIBRARY/ALGOL ON DISK  
 =====

```

    %%% SAMPLE LIBRARY PROGRAM IN ALGOL %%%
    $ SET LIST STACK $
    BEGIN
(01,0002) = BLOCK#1
(01,0003) = SEGMENT DESCRIPTOR
                                BLOCK#1 IS SEGMENT 0003
                                1 00001600 003:0000:1
                                PROCEDURE SQUAREIT (X);
(02,0002) = SQUAREIT
                                REAL X;
                                BEGIN
(03,0002) = X
                                X := X * X;
                                PROGRAMDUMP (LIBRARIES);
(01,0004) = PROGRAMDUMP
                                2 00001800 003:0000:1
                                END;
                                2 00001850 003:0001:4
                                % END OF PROCEDURE SQUAREIT
                                EXPORT SQUAREIT;
                                %%% OUTER BLOCK %%%
                                FREEZE (TEMPORARY);
(02,0003) = LIBRARY DIRECTORY
(01,0005) = FREEZELIB
                                00001870 003:0003:0
                                00001900 003:0003:1
                                00002000 003:0003:1
                                00002100 003:0003:1
                                00002200 003:0003:1
                                END.
(01,0006) = BLOCKEXIT
                                00002300 003:0006:3
    
```

DATA LENGTH IN WORDS IS 000A  
 BLOCK#1(003) LENGTH IN WORDS IS 000E

```

=====
NUMBER OF ERRORS DETECTED = 0.
NUMBER OF SEGMENTS = 4. TOTAL SEGMENT SIZE = 24 WORDS. CORE ESTIMATE = 31 WORDS. STACK ESTIMATE = 7
PROGRAM SIZE = 14 CARDS, 29 SYNTACTIC ITEMS, 10 DISK SECTORS.
PROGRAM FILE NAME: OBJECT/EP4195/LIBRARY/ALGOL ON DISK. B5/6000 CODE GENERATED.
COMPILATION TIME = 2.856 SECONDS ELAPSED; 0.352 SECONDS PROCESSING; 0.752 SECONDS I/O.
=====
    
```

C-27

A SERIES AND B 5000/B 6000/B 7000 CONCEPTS  
 COMPILE LISTINGS AND PROGRAM DUMPS

```

%%%%% SAMPLE LIBRARY USER PROGRAM IN ALGOL %%%%
$ SET LIST STACK $
BEGIN
  LIBRARY MYLIB (TITLE = "OBJECT/EP4195/LIBRARY/ALGOL.");
  PROCEDURE SQUAREIT (X);
    REAL X;
    LIBRARY MYLIB;
  REAL IT;
  %%% OUTER BLOCK %%%
  IT := 9;
  SQUAREIT (IT);
  DISPLAY (STRING(IT,*));
END.

```

```

00000010
00000050
00000100
00000200
00000300
00000400
00000500
00000600
00000700
00000800
00000900
00000950
00001000

```

OBJECT/EP4195/LIBRARY/ALGOL/USER ON DISK  
 =====

```

    %%%% SAMPLE LIBRARY USER PROGRAM IN ALGOL %%%%          00000010 000:0000:0
    $ SET LIST STACK $                                     00000050 000:0000:0
    BEGIN                                                  00000100 000:0000:0
(01,0002) = BLOCK#1
(01,0003) = SEGMENT DESCRIPTOR
                                                    BLOCK#1 IS SEGMENT 0003
                                                    1 00000200 003:0000:1
        LIBRARY MYLIB (TITLE = "OBJECT/EP4195/LIBRARY/ALGOL.");
(02,0002) = FUNNY STRW
(02,0003) = MYLIB
(02,0004) = LIBRARY TEMPLATE MARKER
        PROCEDURE SQUAREIT (X);
(02,0005) = SQUAREIT
        REAL X;
        LIBRARY MYLIB;
        REAL IT;
(02,0006) = IT
        %%% OUTER BLOCK %%%
        IT := 9;
        SQUAREIT (IT);
        DISPLAY (STRING(IT,*));
(01,0004) = DISPLAY
(02,0007) = STRING TEMPORARY
(02,0008) = STRING TEMPORARY.LENGTH
(01,0005) = OUTPUTCONVERT
        END.
                                                    00001000 003:0008:1
(01,0006) = BLOCKEXIT
                                                    DATA LENGTH IN WORDS IS 0020
(01,0007) = GETSTRINGAREA
                                                    BLOCK#1(003) LENGTH IN WORDS IS 0014
    
```

```

=====
NUMBER OF ERRORS DETECTED = 0.
NUMBER OF SEGMENTS = 5. TOTAL SEGMENT SIZE = 52 WORDS. CORE ESTIMATE = 61 WORDS. STACK ESTIMATE = 9
PROGRAM SIZE = 13 CARDS, 39 SYNTACTIC ITEMS, 11 DISK SECTORS.
PROGRAM FILE NAME: OBJECT/EP4195/LIBRARY/ALGOL/USER ON DISK. B5/6000 CODE GENERATED.
COMPILATION TIME = 3.763 SECONDS ELAPSED; 0.446 SECONDS PROCESSING; 0.825 SECONDS I/O.
=====
    
```

C-29

A SERIES AND B 5000/B 6000/B 7000 CONCEPTS  
 COMPILE LISTINGS AND PROGRAM DUMPS

MCP 36.140.3025: \*SYSTEM/MCP36140. INTRINSICS: SYSTEM/INTRINSICS ON DISK. (LOADED)  
SYSTEM SERIAL: #2372 HOSTNAME: SYSEDB6900. GROUP ID: DEFAULT.

CAUSE OF DUMP: PROGRAM REQUESTED @ 003:0003:0, 003:0002:1.

PIB HISTORY WORD AT ENTRY TO PROGRAM DUMP: 0 000000 000000.

PROGRAMDUMP OPTIONS: LIBRARIES

0321 = LOSR (00045C53)

0023 (01,0002) 0 400000 080001 OP: OCT:20000000 02000001 , EBC: ?????, DEC:-524289  
0022 3 000000 30E003 RCW: LL=03, NORML STATE [STACK 307 SEGMENT @ 0003:0003:0]  
SEG DESC: 3 800000 EAD37E [(CONCEPTS)OBJECT/EP4195/LIBRARY/ALGOL.]  
CODE: 3 FF7002 AC3002 3 300282 B8AE60 3 04B096 138EAB >3 A35002 AFB202< 3 5003A6 BAAE60  
0021 ----D[01]=>3 C12000 804003 \*MSCW: PREVIOUS MSCW @ 001E; D[00]=0008 IN STACK 012

0020 (03,0002) 1 704001 400006 SIRW: OFFSET=001A (0014+0006) IN THIS STACK  
001F 3 000200 20A003 RCW: LL=02, NORML STATE [USER SEGMENT @ 0003:0002:1]  
SEG DESC: 3 800001 4C05AE  
CODE: 3 FFB209 5006B8 3 AE5005 5006AF >3 ABAE60 041006< 3 5007BD B7B0A6  
001E ----D[03]=>3 F06001 40C00A \*MSCW: PREVIOUS MSCW @ 0014; D[02]=0014 IN STACK 306

001D (02,0009) 6 800000 080800 SCW: (BLOCK BELOW DECLARED SNGL-DIM ARRAYS, LIB TEMPLATES)  
001C (02,0008) 0 000000 000000  
001B (02,0007) 5 440008 4682E7 DESC [ABSENT-COPY]: STRING (8-BIT), LENGTH=132 (MOM NOT IN THIS STACK OR SEGDICT)  
001A (02,0006) 0 000000 000051 OP: OCT:00000000 00000121 , EBC:?????, DEC:81  
0019 (02,0005) 1 706001 400002 SIRW: OFFSET=0016 (0014+0002) IN STACK 306  
0018 (02,0004) 6 8C2000 000000 SCW: (LIBRARY STRUCTURE MARKER)  
0017 (02,0003) 5 800003 7AE8AE LIBRARY STRUCTURE DESC [PRESENT-MOM]: DATA, LENGTH=55

----- HEADER -----  
STATUS = 800000 030000, LEVEL = 3, LINKED.  
STACK INFORMATION: IMP AT 0017 IN STK 304 = (\*,0003), (\*,0) AT 0014.  
----- USEINFO -----  
LINKED TO STACKS: EXP AT 0017 IN STK 306.  
----- AREAS -----  
FREE 002E  
USEINFO 002C  
STACKREF 0006  
IMPORTS 000B  
EXPORTS 0000  
TYPES 000E  
NAMES 0011  
ATTRIBS 0019  
----- IMPORT OBJECTS -----  
([V] = BY VALUE, [R] = BY REFERENCE, [N] = BY NAME, [RO] = READ ONLY)  
SQUAREIT IS A PROCEDURE (1 PARAMETER);  
REAL[N];  
INDEX = 12, OBJECT = (\*,0005).  
----- ATTRIBUTES -----  
VALUE = 1 H 000 0000 00001, INTNAME = MYLIB, TITLE = OBJECT/EP4195/LIBRARY/ALGOL.

0016 (02,0002) 1 705000 402000 SIRW: OFFSET=0004 (0004+0000) IN STACK 305  
0015 3 00024E E88FCD RCW: LL=02, CNTRL STATE [MCP SEGMENT @ OFCD:04EE:1 (16562000)]  
SIG DESC: 3 8800B6 ABCAC1  
CODE: 3 BEFFFF FFFFFFFF 3 4A6870 C30C30 3 ABA3AE 400AAB >3 A3B234 B22695< 3 B8B180 95B9A2  
0014 ----D[02]=>3 105000 408002 \*MSCW: PREVIOUS MSCW @ 0012; D[01]=0004 IN STACK 305

C-30

A SERIES AND B 5000/B 6000/B 7000 CONCEPTS  
COMPILE LISTINGS AND PROGRAM DUMPS

0013           3 000000 002000 RCW: DUMMY (RUN)  
0012 ----D[02]=>3 F82055 108011 \*MSCW: PREVIOUS MSCW @ 0001; D[01]=0551 IN STACK 382  
0000 = BOSR (00045932)



***** SAMPLE LIBRARY PROGRAM IN COBOL74 *****	000020
\$ SET TEMPORARY LIST MAP	000050
IDENTIFICATION DIVISION.	000100
ENVIRONMENT DIVISION.	000200
DATA DIVISION.	000300
WORKING-STORAGE SECTION.	000400
77 X PIC 9(11) COMP.	000500
PROCEDURE DIVISION USING X.	000600
100-MAIN-PARA.	000700
COMPUTE X = X * X.	000800
CALL SYSTEM DUMP.	000900
EXIT PROGRAM.	001000

(CONCEPTS)OBJECT/EP4195/LIBRARY/C74 ON SYSTEMSED

000020\*\*\*\*\* SAMPLE LIBRARY PROGRAM IN COBOL74 \*\*\*\*\*  
000100 IDENTIFICATION DIVISION.

000200 ENVIRONMENT DIVISION.  
000300 DATA DIVISION.  
000400 WORKING-STORAGE SECTION.  
000500 77 X PIC 9(11) COMP.

000600 PROCEDURE DIVISION USING X.  
WARNING 511 : ON 34: WFL INTEGER PARAM NO LONGER CONVERTED \*\*\* X  
WARNING 512 : TO COMP ITEM; USE BINARY ITEM INSTEAD \*\*\* X  
000700 100-MAIN-PARA.

000800 COMPUTE X = X \* X.  
000900 CALL SYSTEM DUMP.  
WARNING 435 : BURROUGHS EXTENSION EXCEEDS U.S. HIGH LEVEL \*\*\* SYSTEM  
001000 EXIT PROGRAM.

0000:0000:0  
0000:0000:0  
START OF SEGMENT AT (01,002)  
0002:0000:0  
0002:0000:0  
0002:0000:0 0001(0000:1)  
0002:0000:0 0001(0000:1)  
X = (02,002)  
0002:0000:0  
000600 <<0001>>  
0002:0000:0 <<0002>>  
PCW = (02,004)  
PCW = (02,005)  
LIBRARY DIRECTORY = (02,006)  
PCW(002:000:1) = (02,007)  
0002:0000:1  
0002:0004:2  
000600 <<0003>>  
MCP PROCEDURE: PROGRAMDUMP = (01,003)  
0002:0005:3  
PCW = (02,008)  
MCP PROCEDURE: UNRAVEL = (01,004)  
MCP PROCEDURE: MYSELF = (01,005)  
MCP PROCEDURE: CONTINUE = (01,006)  
SEGMENT 0002 IS 0008 LONG  
START OF SEGMENT AT (01,007)  
PCW(007:000:0) = (02,004)  
LIBRARY USER = (02,009)  
MCP PROCEDURE: MUTATE = (01,008)  
MCP PROCEDURE: BLOCKEXIT = (01,009)  
PCW(007:004:4) = (02,005)  
LIBRARY LOCK = (02,00A)  
LIBRARY EVENT = (02,00B)  
MCP PROCEDURE: WAIT = (01,00A)  
PCW(007:00E:3) = (02,008)  
MCP PROCEDURE: CAUSEP = (01,00B)  
LIBRARY EXIT PCW = (02,00D)  
SEGMENT 0007 IS 0010 LONG  
START OF SEGMENT AT (01,00C)  
LIBRARY FIRST EXECUTABLE PCW(00C:008:2) = (01,00D)  
NORMAL FIRST EXECUTABLE PCW(00C:009:4) = (01,00E)  
LIBRARY VARIABLE = (02,003)  
MCP PROCEDURE: FREEZELIB = (01,00F)  
SEGMENT 000C IS 0017 LONG

A SERIES AND B 5000/B 6000/B 7000 CONCEPTS  
COMPILE LISTINGS AND PROGRAM DUMPS

C-34

COMPILE O.K.  
NUMBER OF WARNINGS DETECTED = 0003  
LAST WARNING AT 000900  
TOTAL CARD COUNT: 11  
D[01] STACK SIZE: 0016(010) WORDS  
D[02] STACK SIZE: 0014(00E) WORDS  
CORE ESTIMATE: 589 WORDS  
STACK ESTIMATE: 310 WORDS



CODE FILE SIZE: 7 RECORDS  
PROGRAM SIZE: 3 CODE SEGMENTS, 63 TOTAL WORDS  
SUBROUTINE NAME: C74, LEVEL 02  
COMPILED ON THE B6900 FOR THE LEVEL0 SERIES  
COMPILER COMPILED WITH THE FOLLOWING OPTIONS:  
BOMS.  
COMPILE TIMES: ELAPSED CPU I-O RPM  
0007.355 0000.911 0001.054 00724

```
***** SAMPLE LIBRARY USER PROGRAM IN COBOL74 *****  
$ SET LIST MAP  
IDENTIFICATION DIVISION.  
ENVIRONMENT DIVISION.  
DATA DIVISION.  
WORKING-STORAGE SECTION.  
77 IT PIC 9(11) COMP.  
PROCEDURE DIVISION.  
100-MAIN-PARA.  
    MOVE 9 TO IT.  
    CALL "PROCEDUREDIVISION OF OBJECT/EP4195/LIBRARY/C74"  
        USING IT.  
    DISPLAY IT.  
    STOP RUN.
```

```
000020  
000050  
000100  
000200  
000300  
000400  
000500  
000600  
000700  
000800  
000900  
000950  
000970  
001000
```

(CONCEPTS)OBJECT/EP4195/LIBRARY/C74/USER ON SYSTEMSED

000020\*\*\*\*\* SAMPLE LIBRARY USER PROGRAM IN COBOL74 \*\*\*\*\*  
000100 IDENTIFICATION DIVISION.

000200 ENVIRONMENT DIVISION.  
000300 DATA DIVISION.  
000400 WORKING-STORAGE SECTION.  
000500 77 IT PIC 9(11) COMP.

000600 PROCEDURE DIVISION.  
000700 100-MAIN PARA.

000800 MOVE 9 TO IT.

000900 CALL "PROCEDUREDIVISION OF OBJECT/EP4195/LIBRARY/C74"  
000950 USING IT.

000970 DISPLAY IT.  
001000 STOP RUN.

```

0000:0000:0
0000:0000:0
START OF SEGMENT AT (01,002)
0002:0000:0
0002:0000:0
0002:0000:0      0001(0000:1)
0002:0000:0      0001(0000:1)
IT = (02,003)
0002:0000:0
0002:0000:0
PCW = (02,004)
PCW = (02,005)
LIBRARY DIRECTORY = (02,006)
PCW(002:000:1) = (02,007)
0002:0000:1
TEMPORARY = (02,008)
0002:0002:1
0002:0002:1
SIRW TO D[01] = (02,009)
LIBRARY TEMPLATE = (02,00A)
LIBRARY TEMPLATE MARKER = (02,00B)
LIBRARY ENTRYPOINT REFERENCE = (02,00C)
0002:0003:2
0002:0003:2
TEMPORARY = (02,00D)
MCP PROCEDURE: MESSER = (01,003)
MCP PROCEDURE: GOTOSOLVER = (01,004)
SEGMENT 0002 IS 000A LONG
START OF SEGMENT AT (01,005)
PCW(005:000:0) = (02,004)
LIBRARY USER = (02,00E)
MCP PROCEDURE: MUTATE = (01,006)
MCP PROCEDURE: BLOCKEXIT = (01,007)
PCW(005:004:4) = (02,005)
LIBRARY LOCK = (02,00F)
LIBRARY EVENT = (02,010)
MCP PROCEDURE: WAIT = (01,008)
MCP PROCEDURE: CAUSEP = (01,009)
LIBRARY EXIT PCW = (02,012)
SEGMENT 0005 IS 0019 LONG
START OF SEGMENT AT (01,00A)
LIBRARY FIRST EXECUTABLE PCW(00A:008:2) = (01,00B)
MCP PROCEDURE: MYSELF = (01,00C)
NORMAL FIRST EXECUTABLE PCW(00A:009:3) = (01,00D)
LIBRARY VARIABLE = (02,002)
MCP PROCEDURE: FREEZELIB = (01,00E)
SEGMENT 000A IS 0022 LONG

```

A SERIES AND B 5000/B 6000/B 7000 CONCEPTS  
COMPILE LISTINGS AND PROGRAM DUMPS

C-37

COMPILE O.K.  
TOTAL CARD COUNT: 13  
D[01] STACK SIZE: 0015(00F) WORDS  
D[02] STACK SIZE: 0019(013) WORDS  
CORE ESTIMATE: 594 WORDS  
STACK ESTIMATE: 310 WORDS

CODE FILE SIZE: 10 RECORDS  
PROGRAM SIZE: 3 CODE SEGMENTS, 69 TOTAL WORDS  
SUBROUTINE NAME: USER, LEVEL 02  
COMPILED ON THE B6900 FOR THE LEVEL 0 SERIES  
COMPILER COMPILED WITH THE FOLLOWING OPTIONS:  
BDMS.  
COMPILE TIMES: ELAPSED CPU I-O RPM  
0007.483 0000.931 0001.075 00837

B6900 PROGRAMDUMP FOR STACK 30E (MIX 3014/3031) BOSR=50EE8  
NAME: (CONCEPTS)OBJECT/EP4195/LIBRARY/C74/USER ON SYSTEMSED.

TUESDAY, FEBRUARY 11,1986 15:01:00

MCP 36.140.3025: \*SYSTEM/MCP36140. INTRINSICS: SYSTEM/INTRINSICS ON DISK. (LOADED)  
SYSTEM SERIAL: #2372 HOSTNAME: SYSEDB6900. GROUP ID: DEFAULT.

CAUSE OF DUMP: PROGRAM REQUESTED @ 002:0005:2, 002:0003:2, 00A:0007:3.

PIB HISTORY WORD AT ENTRY TO PROGRAM DUMP: 0 000000 000000.

PROGRAMDUMP OPTIONS: (DEFAULT)

0332 = LOSR (0005121A)

```
0034 (01,0002) 0 000000 000001 OP: OCT:00000000 00000001 , EBC:??????, DEC:1
0033          3 200400 50E002 RCW: LL=03, NORML STATE, TRUE [STACK 311 SEGMENT @ 0002:0005:2]
                                SEG DESC: 3 800000 BB2714 [(CONCEPTS)OBJECT/EP4195/LIBRARY/C74.]
                                CODE: 3 B05002 A6B20B 3 CA8FC6 0BB20B 3 95D1AE 6003B2 >3 02ABFE 5003B0< 3 B094A1 400750
0032 ----D[01]=>3 C12000 804007 *MSCW: PREVIOUS MSCW @ 002B; D[00]=000B IN STACK 012

0031 (03,0006) 0 000000 000000
0030 (03,0005) 6 800000 100000 SCW:
002F (03,0004) 7 30E600 D12007 PCW: LL=04, D[1] SEGMENT @ 0007:000D:3, NORML STATE
002E (03,0003) 7 30E800 C0E007 PCW: LL=03, D[1] SEGMENT @ 0007:000C:4, NORML STATE
002D (03,0002) 5 C20000 C976AE DESC [PRESENT-COPY]: STRING (4-BIT), LENGTH=12 (MOM @ OFFSET=0017 IN THIS STACK)
002C          3 000400 30E002 RCW: LL=03, NORML STATE [USER SEGMENT @ 0002:0003:2]
                                SEG DESC: 3 800000 AAB09C
                                CODE: 3 FEB050 08A6B7 3 BDB209 9C2F2C 3 B8AE50 0C5003 >3 BDAB80 500DA6< 3 B05003 A6B20B
002B ----D[03]=>3 F10001 40C003 *MSCW: PREVIOUS MSCW @ 002B; D[02]=0014 IN STACK 310

002A (03,0002) 0 000000 000000
0029          3 000600 70A00A RCW: LL=02, NORML STATE [USER SEGMENT @ 000A:0007:3]
                                SEG DESC: 3 800002 2AE0EE
                                CODE: 3 A6BAAE 600EB1 3 5006BD 5000AF 3 ABA260 07AE50 >3 07B0AB AE6007< 3 ABA3AE 600CAB
0028 ----D[03]=>3 70E001 40C014 *MSCW: PREVIOUS MSCW @ 0014; D[02]=0014

0027 (02,0013) 6 800000 080800 SCW: (BLOCK BELOW DECLARED SNGL-DIM ARRAYS, LIB TEMPLATES)
0026 (02,0012) 0 000000 000000
0025 (02,0011) 2 000000 000000 DPOP: OCT:00000000 00000000 , 2ND:0
0024 (02,0010) 2 000000 000000 DPOP: OCT:00000000 00000000 , 1ST:0, DBL:0.0
0023 (02,000F) 0 000000 000000
0022 (02,000E) 0 000000 000000
0021 (02,000D) 5 040010 500000 DESC [ABSENT-MOM]: STRING (8-BIT), LENGTH=261 (UNREFERENCED OLAY SPACE)
0020 (02,000C) 1 710001 400005 SIRW: OFFSET=0019 (0014+0005) IN STACK 310
001F (02,000B) 6 8C2000 000000 SCW: (LIBRARY STRUCTURE MARKER)
001E (02,000A) 5 800003 99F141 LIBRARY STRUCTURE DESC [PRESENT-MOM]: DATA, LENGTH=57
001D (02,0009) 1 70F000 402000 SIRW: OFFSET=0004 (0004+0000) IN STACK 30F
001C (02,0008) 5 C00000 1976AE DESC [PRESENT-COPY]: DATA, LENGTH=1 (MOM @ OFFSET=0017 IN THIS STACK)
001B (02,0007) 7 30E200 00E002 PCW: LL=03, D[1] SEGMENT @ 0002:0000:1, NORML STATE
001A (02,0006) 5 000001 040001 DESC [ABSENT-MOM]: DATA, LENGTH=16 (CODEFILE ADRS=1)
0019 (02,0005) 7 30E800 40E005 PCW: LL=03, D[1] SEGMENT @ 0005:0004:4, NORML STATE
0018 (02,0004) 7 30E000 00E005 PCW: LL=03, D[1] SEGMENT @ 0005:0000:0, NORML STATE
0017 (02,0003) 5 820000 C976AE DESC [PRESENT-MOM]: STRING (4-BIT), LENGTH=12
0016 (02,0002) 0 000000 000000
0015          3 00024F E88FCD RCW: LL=02, CNTRL STATE [MCP SEGMENT @ 0FCD:04EE:1 (16562000)]
                                SEG DESC: 3 8800B6 ABCAC1
                                CODE: 3 BEFFFF FFFFFFFF 3 4A6870 C30C30 3 ABA3AE 400AAB >3 A3B234 B22695< 3 B8B180 95B9A2
0014 ----D[02]=>3 F0F000 408002 *MSCW: PREVIOUS MSCW @ 0012; D[01]=0004 IN STACK 30F

0013          3 000000 002000 RCW: DUMMY (RUN)
```

C-39

A SERIES AND B 5000/B 6000/B 7000 CONCEPTS  
COMPILE LISTINGS AND PROGRAM DUMPS

0012 ----D[02]=>3 F82055 108011 \*MSCW: PREVIOUS MSCW @ 0001; D[01]=0551 IN STACK 382

0000 = BOSR (00050EE8)

```
%%%%% ALGOL PROGRAM TO ILLUSTRATE SIMPLE CALCULATION %%%%%  
$ SET LIST STACK CODE LINEINFO $  
BEGIN  
REAL X, Y, Z, ANS;  
X := 1;  
Y := 5;  
Z := 7;  
ANS := (X + Y + 1) / Z;  
PROGRAMDUMP;  
END.
```

```
00000050  
00000100  
00000200  
00000300  
00000400  
00000500  
00000600  
00000700  
00000800  
00000900
```

O B J E C T / E P 4 1 9 5 / C A L C / A L G O L O N D I S K  
 = = = = =

%%%% ALGOL PROGRAM TO ILLUSTRATE SIMPLE CALCULATION %%%  
 BEGIN

0000050 000:0000:0  
 00000200 000:0000:0

(01,0002) = BLOCK#1  
 (01,0003) = SEGMENT DESCRIPTOR

BLOCK#1 IS SEGMENT 0003

REAL X, Y, Z, ANS; 003:0000:0 NVLD FF

1 00000300 003:0000:1

(02,0002) = X  
 (02,0003) = Y  
 (02,0004) = Z  
 (02,0005) = ANS

X := 1;  
 003:0000:1 ONE B1  
 003:0000:2 NAMC (02,0002) 5002  
 003:0000:4 STOD B8

00000400 003:0000:1

Y := 5;  
 003:0000:5 LT8 5 B205  
 003:0001:1 NAMC (02,0003) 5003  
 003:0001:3 STOD B8

00000500 003:0000:5

Z := 7;  
 003:0001:4 LT8 7 B207  
 003:0002:0 NAMC (02,0004) 5004  
 003:0002:2 STOD B8

00000600 003:0001:4

ANS := (X + Y + 1) / Z;  
 003:0002:3 VALC (02,0002) 1002  
 003:0002:5 VALC (02,0003) 1003  
 003:0003:1 ADD 80  
 003:0003:2 ONE B1  
 003:0003:3 ADD 80  
 003:0003:4 VALC (02,0004) 1004  
 003:0004:0 DIVD 83  
 003:0004:1 NAMC (02,0005) 5005  
 003:0004:3 STOD B8

00000700 003:0002:3

PROGRAMDUMP;

(01,0004) = PROGRAMDUMP

003:0004:4 MKST AE  
 003:0004:5 NAMC (01,0004) 6004  
 003:0005:1 LT8 2 B202  
 003:0005:3 ENTR AB

00000800 003:0004:4

END.

(01,0005) = BLOCKEXIT

003:0005:4 MKST AE  
 003:0005:5 NAMC (01,0005) 6005  
 003:0006:1 ENTR AB  
 003:0006:2 EXIT A3

00000900 003:0005:4

\*\*\*\*\* STACK BUILDING CODE FOR LEVEL 02 \*\*\*\*\*  
 003:0006:3 ZERO B0  
 003:0006:4 ZERO B0  
 003:0006:5 ZERO B0  
 003:0007:0 ZERO B0  
 003:0007:1 PUSH B4  
 003:0007:2 ZERO B0  
 003:0007:3 BSET 47 962F

C-42

A SERIES AND B 5000/B 6000/B 7000 CONCEPTS  
 COMPILE LISTINGS AND PROGRAM DUMPS



003:0007:5	LT8	6		B206
003:0008:1	STAG			9584
003:0008:3	BRUN		0000:1	A22000
003:0009:0	NVLD			FF
003:0009:1	NVLD			FF
003:0009:2	NVLD			FF
003:0009:3	NVLD			FF
003:0009:4	NVLD			FF
003:0009:5	NVLD			FF

BLOCK#1(003) LENGTH IN WORDS IS 000A

```

=====
NUMBER OF ERRORS DETECTED = 0.
NUMBER OF SEGMENTS = 3. TOTAL SEGMENT SIZE = 10 WORDS. CORE ESTIMATE = 16 WORDS. STACK ESTIMATE = 6
PROGRAM SIZE = 9 CARDS, 36 SYNTACTIC ITEMS, 9 DISK SECTORS.
PROGRAM FILE NAME: OBJECT/EP4195/CALC/ALGOL ON DISK. B5/6000 CODE GENERATED.
COMPILATION TIME = 2.891 SECONDS ELAPSED; 0.402 SECONDS PROCESSING; 0.746 SECONDS I/O.
=====

```

MCP 36.140.3025: \*SYSTEM/MCP36140. INTRINSICS: SYSTEM/INTRINSICS ON DISK. (LOADED)  
SYSTEM SERIAL: #2372 HOSTNAME: SYSEDB6900. GROUP ID: DEFAULT.

CAUSE OF DUMP: PROGRAM REQUESTED @ 003:0005:4 (00000800).

PIB HISTORY WORD AT ENTRY TO PROGRAM DUMP: 0 000000 000000.

PROGRAMDUMP OPTIONS: (DEFAULT)

031B = LOSR (00017788)

001D (01,0002) 0 000000 000001 OP: OCT:00000000 00000001 , EBC:??????, DEC:1  
001C 3 000800 50A003 RCW: LL=02, NORML STATE [USER SEGMENT @ 0003:0005:4 (00000800)]  
SEG DESC: 3 800000 AE3035  
CODE: 3 500488 100210 3 0380B1 801004 3 835005 B8AE60 >3 048202 ABAE60< 3 05ABA3 B0B0B0  
001B ----D[01]=>3 C12000 804007 \*MSCW: PREVIOUS MSCW @ 0014; D[00]=0008 IN STACK 012

001A (02,0006) 6 800000 000000 SCW:  
0019 (02,0005) 0 261000 000000 OP: OCT:11410000 00000000 , EBC:??????, DEC:1.0  
0018 (02,0004) 0 000000 000007 OP: OCT:00000000 00000007 , EBC:??????, DEC:7  
0017 (02,0003) 0 000000 000005 OP: OCT:00000000 00000005 , EBC:??????, DEC:5  
0016 (02,0002) 0 000000 000001 OP: OCT:00000000 00000001 , EBC:??????, DEC:1  
0015 3 00024E E88FCD RCW: LL=02, CNTRL STATE [MCP SEGMENT @ 0FCD:04EE:1 (16562000)]  
SEG DESC: 3 8800B6 ABCAC1  
CODE: 3 BEFFFF FFFFFF 3 4A6870 C30C30 3 ABA3AE 400AAB >3 A3B234 B22695< 3 B8B180 95B9A2

0014 ----D[02]=>3 F29000 408002 \*MSCW: PREVIOUS MSCW @ 0012; D[01]=0004 IN STACK 329

0013 3 000000 002000 RCW: DUMMY (RUN)  
0012 ----D[02]=>3 F81B74 E08011 \*MSCW: PREVIOUS MSCW @ 0001; D[01]=B74E IN STACK 381

0000 = BOSR (0001746D)

C-44

```
***** COBOL74 PROGRAM TO ILLUSTRATE SIMPLE CALCULATION *****  
$ SET LIST MAP CODE LINEINFO  
IDENTIFICATION DIVISION.  
ENVIRONMENT DIVISION.  
DATA DIVISION.  
WORKING-STORAGE SECTION.  
77 X PIC 99 COMP.  
77 Y PIC 99 COMP.  
77 Z PIC 99 COMP.  
77 ANS PIC 99 COMP.  
PROCEDURE DIVISION.  
100-MAIN-PARA.  
    MOVE 1 TO X.  
    MOVE 5 TO Y.  
    MOVE 7 TO Z.  
    COMPUTE ANS = (X + Y + 1) / Z.  
    CALL SYSTEM DUMP.  
    STOP RUN.
```

```
000050  
000100  
000200  
000300  
000400  
000500  
000600  
000700  
000800  
000900  
001000  
001100  
001200  
001300  
001400  
001500  
001600  
001700
```

(CONCEPTS)OBJECT/EP4195/CALC/C74 ON SYSTEMSED

```

000050***** COBOL74 PROGRAM TO ILLUSTRATE SIMPLE CALCULATION *****
000200 IDENTIFICATION DIVISION.
                                MACRO 0001:06:KBLSM 00AA 0000

000300 ENVIRONMENT DIVISION.
000400 DATA DIVISION.
                                MACRO 0001:0B:SWMCO 0057 0001

000500 WORKING-STORAGE SECTION.
000600 77 X PIC 99 COMP.
                                MACRO 0001:10:DCL77 0002 0000

000700 77 Y PIC 99 COMP.
                                MACRO 0001:13:DCL77 0003 0000

000800 77 Z PIC 99 COMP.
                                MACRO 0001:16:DCL77 0004 0000

000900 77 ANS PIC 99 COMP.
                                MACRO 0001:19:DCL77 0005 0000

001000 PROCEDURE DIVISION.
                                MACRO 0001:1C:SWMCO 0057 0000
                                MACRO 0001:1D:KBLSM 0109 0000

001100 100-MAIN-PARA.
                                MACRO 0001:20:DEFLM 0000 0000
                                FE
                                0002:0000:0 NOOP

                                MACRO 0001:21:DEFLM 0006 0000

001200 MOVE 1 TO X.
                                MACRO 0001:24:SVNLT 0000 0000
                                MACRO 0001:25:LITCM 0001 0400
                                MACRO 0001:26:FIXMM 0000 0000
                                MACRO 0001:27:MVNLT 0000 0002
                                0002:0000:1 LT8 B20A 10.0
                                0002:0000:1 NAMC (02,0005) 5005
                                0002:0000:3 ONE B1
                                0002:0000:4 STOD B8

                                MACRO 0001:28:SNTNC 0441 0000

001300 MOVE 5 TO Y.
                                MACRO 0001:2B:SVNLT 0000 0000
                                MACRO 0001:2C:LITCM 0005 0400
                                MACRO 0001:2D:FIXMM 0000 0000
                                MACRO 0001:2E:MVNLT 0000 0003
                                0002:0000:5 NAMC (02,0006) 5006 Y
                                0002:0001:1 LT8 B205 5.0
                                0002:0001:3 STOD B8

                                MACRO 0001:2F:SNTNC 0441 0000

001400 MOVE 7 TO Z.
                                MACRO 0001:32:SVNLT 0000 0000
                                MACRO 0001:33:LITCM 0007 0400

```

```

0000:0000:0
0000:0000:0
START OF SEGMENT AT (01,002)
0002:0000:0
0002:0000:0
0002:0000:0
0002:0000:0
x = (02,00A)
0002:0000:0
y = (02,006)
0002:0000:0
z = (02,007)
0002:0000:0
ANS = (02,008)
0002:0000:0
0002:0000:0
PCW = (02,00B)
PCW = (02,00C)
LIBRARY DIRECTORY = (02,00D)
PCW(002:000:1) = (02,00E)
0002:0000:1
0002:0000:5
0002:0001:4

```

C-46

A SERIES AND B 5000/B 6000/B 7000 CONCEPTS  
COMPILE LISTINGS AND PROGRAM DUMPS



```

MACRO 0001:56:ENDLM 0000 0000
0002:0008:4  NVLD                                FF                                SEGMENT 0002 IS 0009 LONG
MACRO 0001:57:THEND 0000 0000
START OF SEGMENT AT (01,005)
PCW(005:000:0) = (02,008)

0005:0000:0  LT8                                B235                                53.0
0005:0000:2  RPRR                                95B8

LIBRARY USER = (02,00F)

0005:0000:4  NAMC (02,000F) 500F
0005:0001:0  LOAD                                BD
0005:0001:1  EQU L                                8C
0005:0001:2  FIXUP
0005:0001:5  NVLD                                FF
* 0005:0001:2  BRFL 0002:0  A00002
0005:0002:0  MKST                                AE
MCP PROCEDURE: MUTATE = (01,006)

0005:0002:1  NAMC (01,0006) 6006 MCP.MUTATE
0005:0002:3  NAMC (02,0002) 5002
0005:0002:5  LOAD                                BD
0005:0003:0  LT8                                B20C                                12.0
0005:0003:2  LT8                                B206                                6.0
0005:0003:4  ENTR                                AB
0005:0003:5  MKST                                AE
MCP PROCEDURE: BLOCKEXIT = (01,007)

0005:0004:0  NAMC (01,0007) 6007 MCP.BLOCKEXIT
0005:0004:2  ENTR                                AB
0005:0004:3  EXIT                                A3
PCW(005:004:4) = (02,00C)
LIBRARY LOCK = (02,010)
LIBRARY EVENT = (02,011)

0005:0004:4  LT8                                B235                                53.0
0005:0005:0  RPRR                                95B8
0005:0005:2  NAMC (02,0010) 5010
0005:0005:4  RDLK                                95BA
0005:0006:0  ZERO                                B0
0005:0006:1  EQU L                                8C
0005:0006:2  FIXUP
0005:0006:5  MKST                                AE
MCP PROCEDURE: WAIT = (01,008)

0005:0007:0  NAMC (01,0008) 6008 MCP.WAIT
0005:0007:2  NAMC (02,0011) 5011
0005:0007:4  STFF                                AF
0005:0007:5  ONE                                B1
0005:0008:0  ENTR                                AB
0005:0008:1  ONE                                B1
0005:0008:2  NAMC (02,0010) 5010
0005:0008:4  RDLK                                95BA
0005:0009:0  ZERO                                B0
0005:0009:1  EQU L                                8C
0005:0009:2  BRFL 0006:5  A0A006
* 0005:0006:2  BRTR 0009:5  A1A009
0005:0009:5  LT8                                B235                                53.0
0005:000A:1  RPRR                                95B8
0005:000A:3  NAMC (02,000F) 500F
0005:000A:5  STOD                                B8
0005:000B:0  FIXUP
0005:000B:3  MKST                                AE
0005:000B:4  NAMC (01,0007) 6007 MCP.BLOCKEXIT
0005:000C:0  ENTR                                AB
0005:000C:1  EXIT                                A3

```

```

0005:000C:2 ZERO 80
0005:000C:3 NAMC (02,000F) 500F
0005:000C:5 STOD 88
0005:000D:0 ZERO 80
0005:000D:1 NAMC (02,0010) 5010
0005:000D:3 RDLK 958A
0005:000D:5 LT8 8235 53.0
0005:000E:1 RPRR 9588
0005:000E:3 EQU 8C
0005:000E:4 FIXUP
0005:000F:1 MKST AE
MCP PROCEDURE: CAUSEP = (01,009)
0005:000F:2 NAMC (01,0009) 6009 MCP.CAUSEP
0005:000F:4 NAMC (02,0011) 5011
0005:0010:0 STFF AF
0005:0010:1 ONE B1
0005:0010:2 ENTR AB
* 0005:000E:4 BRTR 0010:3 A16010
0005:0010:3 ZERO 80
0005:0010:4 RETN A7
* 0005:0008:0 BRUN 0010:5 A2A010
0005:0010:5 MPCW BF000600B0E005
0005:0012:0 NAMC (03,0002) 7002
0005:0012:2 STFF AF
LIBRARY EXIT PCW = (02,013)
0005:0012:3 NAMC (02,0013) 5013
0005:0012:5 OVRD BA
0005:0013:0 MPCW BF000400C12005
0005:0015:0 LT48 BF800000100000 1048576.0
0005:0017:0 LT8 B206 6.0
0005:0017:2 STAG 9584
0005:0017:4 ZERO 80
0005:0017:5 NAMC (02,000E) 500E
0005:0018:1 DBUN AA
SEGMENT 0005 IS 0019 LONG
START OF SEGMENT AT (01,00A)
LIBRARY FIRST EXECUTABLE PCW(00A:008:2) = (01,00B)
000A:0008:2 MKST AE
MCP PROCEDURE: MYSELF = (01,00C)
000A:0008:3 NAMC (01,000C) 600C MCP.MYSELF
000A:0008:5 ENTR AB
000A:0009:0 FIXUP
NORMAL FIRST EXECUTABLE PCW(00A:009:3) = (01,00D)
000A:0009:3 ZERO 80
* 000A:0009:0 BRUN 0009:4 A78009
STKPARAM = (02,002)
000A:0009:4 ZERO 80
000A:0009:5 LT8 B204 4.0
000A:000A:1 INSR 9C2714
000A:000A:4 LT8 B205 5.0
000A:000B:0 STAG 9584
STKARRAY = (02,003)
000A:000B:2 ZERO 80
TEMPORARY = (02,004)
000A:000B:3 ZERO 80
TEMPORARY = (02,005)
000A:000B:4 ZERO 80
STKOP = (02,006)
000A:000B:5 ZERO 80
STKOP = (02,007)
000A:000C:0 ZERO 80

```

```

                                STKOP = (02,008)
000A:000C:1 ZERO                B0
000A:000C:2 MKST                AE
                                MCP PROCEDURE: INSTACKARRAYDEC = (01,00E)
000A:000C:3 NAMC (01,000E)      600E MCP.INSTACKARRAYDEC
000A:000C:5 NAMC (02,0003)      5003
000A:000D:1 STFF                AF
000A:000D:2 NAMC (02,0005)      5005
000A:000D:4 STFF                AF
000A:000D:5 ONE                 B1
000A:000E:0 ENTR                AB
                                = (02,009)
000A:000E:1 PUSH                B4
000A:000E:2 NAMC (02,0003)      5003
000A:000E:4 LOAD                BD
000A:000E:5 LT8                 B230          2097200.0
000A:000F:1 BSET                9615
000A:000F:3 INSR                9C2A17
                                STKARRAY = (02,00A)
000A:0010:0 MPCW                BF00000000E005
                                STKPCW = (02,00B)
000A:0012:0 MPCW                BF00080040E005
                                STKPCW = (02,00C)
000A:0014:0 LT48                BE000001040001 17039361.0
000A:0016:0 LT8                 B205          5.0
000A:0016:2 STAG                95B4
                                STKARRAY = (02,00D)
000A:0016:4 MPCW                BF00020000E002
                                STKPCW = (02,00E)
000A:0018:0 ZERO                B0
                                TEMPORARY = (02,00F)
000A:0018:1 ZERO                B0
                                STKOP = (02,010)
000A:0018:2 ZERO                B0
                                STKOP = (02,011)
000A:0018:3 ZERO                B0
000A:0018:4 JOIN                9542
                                STKOP = (02,012)
000A:0019:0 ZERO                B0
                                TEMPORARY = (02,013)
000A:0019:1 LT16                B30800        2048.0
000A:0019:4 BSET                962F
000A:001A:0 LT8                 B206          6.0
000A:001A:2 STAG                95B4
000A:001A:4 PUSH                B4
000A:001A:5 BRUN 0000:0        A20000
000A:0000:0 NAMC (02,0002)      5002
                                LIBRARY VARIABLE = (02,002)
000A:0000:2 LOAD                BD
000A:0000:3 ZERO                B0
000A:0000:4 SAME                94
000A:0000:5 FIXUP
000A:0001:2 NAMC (02,000B)      500B
000A:0001:4 STFF                AF
000A:0001:5 LT8                 B202          2.0
000A:0002:1 NAMC (02,000D)      500D
000A:0002:3 INDX                A6
000A:0002:4 OVRD                BA
000A:0002:5 NAMC (02,000C)      500C
000A:0003:1 STFF                AF
000A:0003:2 LT8                 B204          4.0

```



000A:0003:4	NAMC (02,000D)	500D	
000A:0004:0	INDX	A6	
000A:0004:1	OVRD	BA	
000A:0004:2	MKST	AE	
			MCP PROCEDURE: FREEZELIB = (01,00F)
000A:0004:3	NAMC (01,000F)	600F	MCP.FREEZELIB
000A:0004:5	ONE	B1	
000A:0005:0	NAMC (02,000D)	500D	
000A:0005:2	LOAD	BD	
000A:0005:3	NAMC (02,000D)	5000	
000A:0005:5	STFF	AF	
000A:0006:0	ENTR	AB	
000A:0006:1	FIXUP		
* 000A:0000:5	BRTR 0006:4	A18006	
000A:0006:4	MKST	AE	
000A:0006:5	NAMC (02,000E)	500E	
000A:0007:1	ZERO	B0	
000A:0007:2	ENTR	AB	
* 000A:0006:1	BRUN 0007:3	A26007	
000A:0007:3	MKST	AE	
000A:0007:4	NAMC (01,0007)	6007	MCP.BLOCKEXIT
000A:0008:0	ENTR	AB	
000A:0008:1	EXIT	A3	

SEGMENT 000A IS 001C LONG  
DATA SEGMENT 0001 IS 0010 LONG

COMPILE O.K.  
NUMBER OF WARNINGS DETECTED = 0001  
LAST WARNING AT 001600  
TOTAL CARD COUNT: 17  
D[01] STACK SIZE: 0016(010) WORDS  
D[02] STACK SIZE: 0020(014) WORDS  
CORE ESTIMATE: 604 WORDS  
STACK ESTIMATE: 310 WORDS  
CODE FILE SIZE: 9 RECORDS  
PROGRAM SIZE: 3 CODE SEGMENTS, 62 TOTAL WORDS  
SUBROUTINE NAME: C74, LEVEL 02  
COMPILED ON THE B6900 FOR THE LEVELO SERIES  
COMPILER COMPILED WITH THE FOLLOWING OPTIONS:  
BDMS.  
COMPILE TIMES: ELAPSED CPU I-O RPM  
0010.971 0001.457 0001.284 00700

C-51

A SERIES AND B 5000/B 6000/B 7000 CONCEPTS  
COMPILE LISTINGS AND PROGRAM DUMPS

MCP 36.140.3025: \*SYSTEM/MCP36140. INTRINSICS: SYSTEM/INTRINSICS ON DISK. (LOADED)  
SYSTEM SERIAL: #2372 HOSTNAME: SYSEDB6900. GROUP ID: DEFAULT.

CAUSE OF DUMP: PROGRAM REQUESTED @ 002:0007:4 (001600), 00A:0007:3.

PIB HISTORY WORD AT ENTRY TO PROGRAM DUMP: 0 000000 000000.

PROGRAMDUMP OPTIONS: (DEFAULT)

032C = LOSR (00029D85)

002E (01,0002) 0 000000 000001 OP: OCT:00000000 00000001 , EBC:??????, DEC:1  
002D 3 200800 70E002 RCW: LL=03, NORML STATE, TRUE [USER SEGMENT @ 0002:0007:4 (001600)]  
SEG DESC: 3 800000 9C61A0  
CODE: 3 0680B1 801007 3 849E2E B26485 3 865008 B8AE60 >3 03B202 ABFEAE< 3 6004B0 ABFF00  
002C ----D[01]=>3 C12000 804003 \*MSCW: PREVIOUS MSCW @ 0029; D[00]=0008 IN STACK 012  
002B (03,0002) 0 000000 000000  
002A 3 000600 70A00A RCW: LL=02, NORML STATE [USER SEGMENT @ 000A:0007:3]  
SEG DESC: 3 800001 CDE381  
CODE: 3 A6BAAE 600FB1 3 500DBD 5000AF 3 ABA260 07AE50 >3 0EBOAB AE6007< 3 ABA3AE 600CAB  
0029 ----D[03]=>3 72C001 40C015 \*MSCW: PREVIOUS MSCW @ 0014; D[02]=0014  
0028 (02,0014) 6 800000 000800 SCW: (BLOCK BELOW DECLARED SNGL-DIM ARRAYS)  
0027 (02,0013) 0 000000 000000  
0026 (02,0012) 2 000000 000000 DPOP: OCT:00000000 00000000 , 2ND:0  
0025 (02,0011) 2 000000 000000 DPOP: OCT:00000000 00000000 , 1ST:0, DBL:0.0  
0024 (02,0010) 0 000000 000000  
0023 (02,000F) 0 000000 000000  
0022 (02,000E) 7 32C200 00E002 PCW: LL=03, D[1] SEGMENT @ 0002:0000:1, NORML STATE  
0021 (02,000D) 5 000001 040001 DESC [ABSENT-MOM]: DATA, LENGTH=16 (CODEFILE ADRS=1)  
0020 (02,000C) 7 32C800 40E005 PCW: LL=03, D[1] SEGMENT @ 0005:0004:4, NORML STATE  
001F (02,000B) 7 32C000 00E005 PCW: LL=03, D[1] SEGMENT @ 0005:0000:0, NORML STATE  
001E (02,000A) 5 C20003 029A72 DESC [PRESENT-COPY]: STRING (4-BIT), LENGTH=48 (POINTS @ OFFSET=0019 IN THIS STACK)  
001D (02,0009) 3 000000 000000 SEG DESC [ABSENT-MOM]: LENGTH=0 (CODEFILE ADRS=0)  
001C (02,0008) 0 000000 000001 OP: OCT:00000000 00000001 , EBC:??????, DEC:1  
001B (02,0007) 0 000000 000007 OP: OCT:00000000 00000007 , EBC:??????, DEC:7  
001A (02,0006) 0 000000 000005 OP: OCT:00000000 00000005 , EBC:??????, DEC:5  
0019 (02,0005) 0 000000 000001 OP: OCT:00000000 00000001 , EBC:??????, DEC:1  
0018 (02,0004) 5 400000 129A70 DESC [ABSENT-COPY]: DATA, LENGTH=1 (MOM @ OFFSET=0017 IN THIS STACK)  
0017 (02,0003) 5 C00000 429A72 DESC [PRESENT-COPY]: DATA, LENGTH=4 (POINTS @ OFFSET=0019 IN THIS STACK)  
0016 (02,0002) 0 000000 000000  
0015 3 00024E E88FCD RCW: LL=02, CNTRL STATE [MCP SEGMENT @ 0FCD:04EE:1 (16562000)]  
SEG DESC: 3 8800B6 ABCAC1  
CODE: 3 BEFFFF FFFFFFFF 3 4A6870 C30C30 3 ABA3AE 400AAB >3 A3B234 B22695< 3 B8B180 95B9A2  
0014 ----D[02]=>3 F2D000 408002 \*MSCW: PREVIOUS MSCW @ 0012; D[01]=0004 IN STACK 32D  
0013 3 000000 002000 RCW: DUMMY (RUN)  
0012 ----D[02]=>3 F81B74 E08011 \*MSCW: PREVIOUS MSCW @ 0001; D[01]=B74E IN STACK 381  
0000 = BOSR (00029A59)

C-52

A SERIES AND B 5000/B 6000/B 7000 CONCEPTS  
COMPILE LISTINGS AND PROGRAM DUMPS



FILE ATTRIBUTES FOR: FIN TITLE=(CONCEPTS)EP4195/DATA/FILE ON SYSTEMSED HOSTNAME=SYSEDB6900 KIND=PACK INTMODE=EBCDIC  
 EXTMODE=EBCDIC FILETYPE=0 MINRECSIZE=0 MAXRECSIZE=15 BLOCKSIZE=60 FRAME SIZE=48 MYUSE=IN BUFFERS=2 TRANSLATE=FULLTRANS  
 PROTECTION=SAVE POPULATION=1 AREAS=20 AREALENGTH=15000 FLEXIBLE LASTRECORD=6 FILEKIND=DATA ROWSINUSE=1 USERINFO=000000000000  
 CREATIONDATE=11/22/82(82326) LASTACCESSDATE=02/11/86(86042) CYCLE=1 VERSION=0 SAVEFACTOR=0 SECURITYTYPE=PRIVATE SECURITYUSE=10  
 PACKNAME=SYSTEMSED

1E!	REC001	A1234F	IRST C	USTOME	R		{?????	??????	!
H!	D9C5C3F0F0F1	C1F1F2F3F4C6	C9D9E2E340C3	E4E2E3D6D4C5	D94040404040	404040404040	C00000012500	000000000000	!
(0000048)E!	??????	??????	??????	??????	??????	??????	??????	??????	!...90
(0000096)H!	000000000000	000000000000	000000000000	000000000000	000000000000	000000000000	000000000000	000000000000	!...180
2E!	REC002	S5432B	ALANCE	IS 34	2.57		{?????	??????	!
H!	D9C5C3F0F0F2	E2F5F4F3F2C2	C1D3C1D5C3C5	40C9E240F3F4	F24BF5F74040	404040404040	C00000034257	000000000000	!
(0000048)E!	??????	??????	??????	??????	??????	??????	??????	??????	!...90
(0000096)H!	000000000000	000000000000	000000000000	000000000000	000000000000	000000000000	000000000000	000000000000	!...180
3E!	REC003	Z6789Z	ERO BA	LANCE	CUSTOM	ER	{?????	??????	!
H!	D9C5C3F0F0F3	E9F6F7F8F9E9	C5D9D640C2C1	D3C1D5C3C540	C3E4E2E3D6D4	C5D940404040	C00000000000	000000000000	!
(0000048)E!	??????	??????	??????	??????	??????	??????	??????	??????	!...90
(0000096)H!	000000000000	000000000000	000000000000	000000000000	000000000000	000000000000	000000000000	000000000000	!...180
4E!	REC004	CR333C	REDIT	BALANC	E CUST	OMER	}?????	??????	!
H!	D9C5C3F0F0F4	C3D9F3F3F3C3	D9C5C4C9E340	C2C1D3C1D5C3	C540C3E4E2E3	D6D4C5D94040	D00000005478	000000000000	!
(0000048)E!	??????	??????	??????	??????	??????	??????	??????	??????	!...90
(0000096)H!	000000000000	000000000000	000000000000	000000000000	000000000000	000000000000	000000000000	000000000000	!...180
5E!	REC005	B2222B	EGINNI	NG OF	NEW BL	OCK	{?????	??????	!
H!	D9C5C3F0F0F5	C2F2F2F2F2C2	C5C7C9D5D5C9	D5C740D6C640	D5C5E640C2D3	D6C3D2404040	C00000014239	000000000000	!
(0000048)E!	??????	??????	??????	??????	??????	??????	??????	??????	!...90
(0000096)H!	000000000000	000000000000	000000000000	000000000000	000000000000	000000000000	000000000000	000000000000	!...180
6E!	REC006	M8765B	IG SPE	NDER			{??g/?	??????	!
H!	D9C5C3F0F0F6	D4F8F7F6F5C2	C9C740E2D7C5	D5C4C5D94040	404040404040	404040404040	C00054876117	000000000000	!
(0000048)E!	??????	??????	??????	??????	??????	??????	??????	??????	!...90
(0000096)H!	000000000000	000000000000	000000000000	000000000000	000000000000	000000000000	000000000000	000000000000	!...180
7E!	REC007	T2875S	KINFLI	NT			{?????	??????	!
H!	D9C5C3F0F0F7	E3F2F8F7F5E2	D2C9D5C6D3C9	D5E340404040	404040404040	404040404040	C00000000025	000000000000	!
(0000048)E!	??????	??????	??????	??????	??????	??????	??????	??????	!...90
(0000096)H!	000000000000	000000000000	000000000000	000000000000	000000000000	000000000000	000000000000	000000000000	!...180

EOF - FILE CONTAINS 7 RECORDS  
 7 RECORDS PROCESSED

C-54

A SERIES AND B 5000/B 6000/B 7000 CONCEPTS  
 COMPILE LISTINGS AND PROGRAM DUMPS

```

***** COBOL74 PROGRAM TO SHOW DUMP WITH DISK FILE BUFFERS *****
$ SET LIST MAP
IDENTIFICATION DIVISION.
ENVIRONMENT DIVISION.
INPUT-OUTPUT SECTION.
FILE-CONTROL.
    SELECT DATAFILE ASSIGN TO DISK.
DATA DIVISION.
FILE SECTION.
FD DATAFILE
    RECORD CONTAINS 90 CHARACTERS;
    BLOCK CONTAINS 360 CHARACTERS;
    VALUE OF TITLE IS "EP4195/DATA/FILE".
01 DATA-RECORD.
    03 RECNO          PIC X(6).
    03 ACCTNO         PIC X(5).
    03 NAME           PIC X(25).
    03 BALANCE        PIC S9(9)V99  COMP.
    03 FILLER         PIC X(48).
WORKING-STORAGE SECTION.
77 COUNTER           PIC 99 COMP.
77 EOF-SW            PIC 9.
88 EOF               VALUE 1.
PROCEDURE DIVISION.
100-MAIN-LOGIC.
    PERFORM 200-OPEN-FILE.
    PERFORM 300-READ-LOOP UNTIL EOF.
    PERFORM 400-CLOSE-FILE.
    STOP RUN.
200-OPEN-FILE.
    OPEN INPUT DATAFILE.
    READ DATAFILE AT END MOVE 1 TO EOF-SW.
    MOVE 1 TO COUNTER.
300-READ-LOOP.
    IF COUNTER EQUAL 4
        CALL SYSTEM DUMP.
    READ DATAFILE AT END MOVE 1 TO EOF-SW.
    ADD 1 TO COUNTER.
400-CLOSE-FILE.
    CLOSE DATAFILE WITH LOCK.

```

000050  
000070  
000100  
000300  
000400  
000500  
000600  
000700  
000800  
000900  
000950  
000955  
001000  
001100  
001200  
001300  
001320  
001340  
001400  
001400  
001500  
001600  
001700  
001800  
001900  
002000  
002100  
002200  
002300  
002400  
002500  
002600  
002700  
002800  
002900  
003000  
003100  
003200  
003300  
003400  
003500

(CONCEPTS)OBJECT/EP4195/FILE/DUMP/C74 ON SYSTEMSED

000050\*\*\*\*\* COBOL74 PROGRAM TO SHOW DUMP WITH DISK FILE BUFFERS \*\*\*\*\*  
 000100 IDENTIFICATION DIVISION.

000300 ENVIRONMENT DIVISION.  
 000400 INPUT-OUTPUT SECTION.  
 000500 FILE-CONTROL.  
 000600 SELECT DATAFILE ASSIGN TO DISK.  
 000700 DATA DIVISION.  
 000800 FILE SECTION.  
 000900 FD DATAFILE  
 000950 RECORD CONTAINS 90 CHARACTERS;

000955 BLOCK CONTAINS 360 CHARACTERS;  
 001000 VALUE OF TITLE IS "EP4195/DATA/FILE".  
 001100 01 DATA-RECORD.  
 001200 03 RECNO PIC X(6).  
 001300 03 ACCTNO PIC X(5).  
 001320 03 NAME PIC X(25).  
 001340 03 BALANCE PIC S9(9)V99 COMP.  
 001400 03 FILLER PIC X(48).

001500 WORKING-STORAGE SECTION.

001600 77 COUNTER PIC 99 COMP.  
 001700 77 EOF-SW PIC 9.  
 001800 88 EOF VALUE 1.  
 001900 PROCEDURE DIVISION.  
 002000 100-MAIN-LOGIC.

002100 PERFORM 200-OPEN-FILE.  
 002200 PERFORM 300-READ-LOOP UNTIL EOF.  
 002300 PERFORM 400-CLOSE-FILE.  
 002400 STOP RUN.

002500 200-OPEN-FILE.  
 002600 OPEN INPUT DATAFILE.

002700 READ DATAFILE AT END MOVE 1 TO EOF-SW.  
 002800 MOVE 1 TO COUNTER.  
 002900 300 READ LOOP.

0000:0000:0  
 0000:0000:0  
 START OF SEGMENT AT (01,002)  
 0002:0000:0  
 0002:0000:0  
 0002:0000:0  
 0002:0000:0  
 0002:0000:0  
 0002:0000:0  
 0002:0000:0  
 0002:0000:0  
 0002:0000:0  
 SIRW TO D[01] = (02,003)  
 MYUSE VALUE = (02,005)  
 DATAFILE = (02,004)  
 0002:0000:0  
 0002:0000:0  
 0002:0000:0  
 0002:0000:0  
 0002:0000:0  
 DATA-RECORD = (02,006)  
 0002:0000:0  
 0006(0001:0)  
 0002:0000:0  
 0008(0001:5)  
 0002:0000:0  
 0048(0006:0)  
 0002:0000:0  
 BALANCE = (02,007)  
 0002:0000:0  
 DATAFILE(MAXRECSIZE) = 90  
 DATAFILE(BLOCKSIZE) = 360  
 DATAFILE(INTMODE) = EBCDIC  
 0002:0000:0  
 COUNTER = (02,00C)  
 0002:0000:0  
 EOF-SW = (02,011)  
 0002:0000:0  
 0002:0000:0  
 0002:0000:0  
 PCW = (02,012)  
 PCW = (02,013)  
 LIBRARY DIRECTORY = (02,014)  
 PCW(002:000:1) = (02,015)  
 0002:0000:1  
 0002:0003:0  
 0002:0003:0  
 0002:000A:0  
 MCP PROCEDURE: GOTOSOLVER = (01,003)  
 0002:000A:5  
 0002:000A:5  
 MCP PROCEDURE: ATTRIBUTEGRABBER = (01,004)  
 MCP PROCEDURE: ATTRIBUTEHANDLER = (01,005)  
 MCP PROCEDURE: NEWOPEN = (01,006)  
 MCP PROCEDURE: HANDLEERROR = (01,007)  
 0002:001B:0  
 0002:0023:1  
 0002:0023:5

C-56

A SERIES AND B 5000/B 6000/B 7000 CONCEPTS  
 COMPILE LISTINGS AND PROGRAM DUMPS

003000 IF COUNTER EQUAL 4  
 003100 CALL SYSTEM DUMP.  
 WARNING 435 : BURROUGHS EXTENSION EXCEEDS U.S. HIGH LEVEL \*\*\* SYSTEM

003200 READ DATAFILE AT END MOVE 1 TO EOF-SW.  
 003300 ADD 1 TO COUNTER.  
 003400 400-CLOSE-FILE.  
 003500 CLOSE DATAFILE WITH LOCK.

0002:0025:2  
 0002:0027:4  
 <<0001>>  
 MCP PROCEDURE: PROGRAMDUMP = (01,008)  
 0002:0028:5  
 0002:0031:0  
 0002:0033:5  
 0002:0035:2  
 MCP PROCEDURE: NEWCLOSE = (01,009)  
 SEGMENT 0002 IS 003C LONG  
 START OF SEGMENT AT (01,00A)  
 PCW(00A:000:0) = (02,012)  
 LIBRARY USER = (02,016)  
 MCP PROCEDURE: MUTATE = (01,00B)  
 MCP PROCEDURE: BLOCKEXIT = (01,00C)  
 PCW(00A:004:4) = (02,013)  
 LIBRARY LOCK = (02,017)  
 LIBRARY EVENT = (02,018)  
 MCP PROCEDURE: WAIT = (01,00D)  
 MCP PROCEDURE: CAUSEP = (01,00E)  
 LIBRARY EXIT PCW = (02,01A)  
 SEGMENT 000A IS 0019 LONG  
 START OF SEGMENT AT (01,00F)  
 LIBRARY FIRST EXECUTABLE PCW(00F:008:2) = (01,010)  
 MCP PROCEDURE: MYSELF = (01,011)  
 NORMAL FIRST EXECUTABLE PCW(00F:009:3) = (01,012)  
 MCP PROCEDURE: INSTACKARRAYDEC = (01,013)  
 LIBRARY VARIABLE = (02,002)  
 MCP PROCEDURE: FREEZELIB = (01,014)  
 SEGMENT 000F IS 0027 LONG

COMPILE O.K.  
 NUMBER OF WARNINGS DETECTED = 0001  
 LAST WARNING AT 003100  
 TOTAL CARD COUNT: 39  
 D[01] STACK SIZE: 0021(015) WORDS  
 D[02] STACK SIZE: 0027(01B) WORDS  
 CORE ESTIMATE: 806 WORDS  
 STACK ESTIMATE: 310 WORDS  
 CODE FILE SIZE: 10 RECORDS  
 PROGRAM SIZE: 3 CODE SEGMENTS, 124 TOTAL WORDS  
 SUBROUTINE NAME: C74, LEVEL 02  
 COMPILED ON THE B6900 FOR THE LEVEL0 SERIES  
 COMPILER COMPILED WITH THE FOLLOWING OPTIONS:  
 BDMS.  
 COMPILE TIMES: ELAPSED CPU I-O RPM  
 0012.007 0001.234 0001.302 01896

C-57

A SERIES AND B 5000/B 6000/B 7000 CONCEPTS  
 COMPILE LISTINGS AND PROGRAM DUMPS

MCP 36.140.3025: \*SYSTEM/MCP36140. INTRINSICS: SYSTEM/INTRINSICS ON DISK. (LOADED)  
SYSTEM SERIAL: #2372 HOSTNAME: SYSEDB6900. GROUP ID: DEFAULT.

CAUSE OF DUMP: PROGRAM REQUESTED @ 002:0028:4, OOF:0007:3.

PIB HISTORY WORD AT ENTRY TO PROGRAM DUMP: 0 000000 000000.

PROGRAMDUMP OPTIONS: FILE(S)

0335 = LOSR (0003CE19)

0037 (01,0002) 0 000000 000401 OP: OCT:00000000 00002001 , EBC:??????, DEC:1025  
0036 3 200802 80E002 RCW: LL=03, NORML STATE, TRUE [USER SEGMENT @ 0002:0028:4]  
SEG DESC: 3 800003 CE1DAF  
CODE: 3 B5AAB2 0A500C 3 A6B202 CAB204 3 8CA0A0 28AE60 >3 08B202 ABFEAE <3 B05004 A6BD81  
0035 ----D[01]>=>3 C12000 804005 \*MSCW: PREVIOUS MSCW @ 0030; D[00]=0008 IN STACK 012  
0034 (03,0004) 0 000000 000003 OP: OCT:00000000 00000003 , EBC:??????, DEC:3  
0033 (03,0003) 7 332000 70E002 PCW: LL=03, D[1] SEGMENT @ 0002:0007:0, NORML STATE  
0032 (03,0002) 0 000000 000000  
0031 3 000600 70A00F RCW: LL=02, NORML STATE [USER SEGMENT @ 000F:0007:3]  
SEG DESC: 3 800002 787320  
CODE: 3 A6BAAE 6014B1 3 5014BD 5000AF 3 ABA260 07AE50 >3 15B0AB AE600C <3 ABA3AE 6011AB  
0030 ----D[03]>=>3 732001 40C01C \*MSCW: PREVIOUS MSCW @ 0014; D[02]=0014  
002F (02,001B) 6 800000 002800 SCW: (BLOCK BELOW DECLARED FILES, SNGL-DIM ARRAYS)  
002E (02,001A) 0 000000 000000  
002D (02,0019) 2 000000 000000 DPOP: OCT:00000000 00000000 , 2ND:0  
002C (02,0018) 2 000000 000000 DPOP: OCT:00000000 00000000 , 1ST:0, DBL:0.0  
002B (02,0017) 0 000000 000000  
002A (02,0016) 0 000000 000000  
0029 (02,0015) 7 332200 00E002 PCW: LL=03, D[1] SEGMENT @ 0002:0000:1, NORML STATE  
0028 (02,0014) 5 000001 040001 DESC [ABSENT-MOM]: DATA, LENGTH=16 (CODEFILE ADRS=1)  
0027 (02,0013) 7 332800 40E00A PCW: LL=03, D[1] SEGMENT @ 000A:0004:4, NORML STATE  
0026 (02,0012) 7 332000 00E00A PCW: LL=03, D[1] SEGMENT @ 000A:0000:0, NORML STATE  
0025 (02,0011) 5 C40000 63CB07 DESC [PRESENT-COPY]: STRING (8-BIT), LENGTH=6 (POINTS @ OFFSET=0023 IN THIS STACK)  
0024 (02,0010) 3 000000 000000 SEG DESC [ABSENT-MOM]: LENGTH=0 (CODEFILE ADRS=0)  
0023 (02,000F) 0 000000 000000  
0022 (02,000E) 5 400000 13CB05 DESC [ABSENT-COPY]: DATA, LENGTH=1 (MOM @ OFFSET=0021 IN THIS STACK)  
0021 (02,000D) 5 C00000 13CB07 DESC [PRESENT-COPY]: DATA, LENGTH=1 (POINTS @ OFFSET=0023 IN THIS STACK)  
0020 (02,000C) 5 C20000 C3CB02 DESC [PRESENT-COPY]: STRING (4-BIT), LENGTH=12 (POINTS @ OFFSET=001E IN THIS STACK)  
001F (02,000B) 3 000000 000000 SEG DESC [ABSENT-MOM]: LENGTH=0 (CODEFILE ADRS=0)  
001E (02,000A) 0 000000 000004 OP: OCT:00000000 00000004 , EBC:??????, DEC:4  
001D (02,0009) 5 400000 13CB00 DESC [ABSENT-COPY]: DATA, LENGTH=1 (MOM @ OFFSET=001C IN THIS STACK)  
001C (02,0008) 5 C00000 13CB02 DESC [PRESENT-COPY]: DATA, LENGTH=1 (POINTS @ OFFSET=001E IN THIS STACK)  
001B (02,0007) 5 C2000B 4E8107 DESC [PRESENT-COPY]: STRING (4-BIT), LENGTH=180 (MOM @ OFFSET=001A IN THIS STACK)  
001A (02,0006) 5 840005 AE8107 DESC [PRESENT-MOM]: STRING (8-BIT), LENGTH=90  
0019 (02,0005) 0 000000 000001 OP: OCT:00000000 00000001 , EBC:??????, DEC:1  
0018 (02,0004) 5 800004 42E801 DESC [PRESENT-MOM]: DATA, LENGTH=68 (FILE: INTNAME=DATAFILE., TITLE=EP4195/DATA/FILE.)  
0(00) 1 782E80 200010 SELECTOR  
1(01) 3 C12000 804002 FIBMSCW  
2(02) 0 000000 000000 FIBLOCK  
3(03) 0 0005A0 000000 RECORDSTATUS  
4(04) 0 011100 466027 FILESTATUS (LEVEL=1,COBOL74, BUFFERS=2, NEWBUFFER, READSERIAL STATE)  
5(05) 0 C98000 000000 TANKDATA1 (FILETYPE=0, UNITS=CHARS, EXTMODE=EBCDIC, INTMODE=EBCDIC, BLOCKED)  
6(06) 0 016800 00005A TANKDATA2 (PHYSICAL: BLOCKSIZE=360, MINRECSIZE=0, MAXRECSIZE=90)  
7(07) 0 002000 400000 DISKBLOCK (SEGPBRBLK=2, RECPBRBLK=4)

C-58

A SERIES AND B 5000/B 6000/B 7000 CONCEPTS  
COMPILE LISTINGS AND PROGRAM DUMPS



8(08) 0 000800 000000 PAGESPEC  
 9(09) 0 130000 000011 IOINFO (KIND=PACK)  
 10(OA) 5 800003 032F2B LEB (LEB SHOWN BELOW IN HEX)

```

=====
0(0000) 0 B32002 400411 0 000001 000001 0 000000 000000 0 020440 000000 0 600A21 FFE834
5(0005) 0 016800 00005A 0 000000 000000 0 000004 0000C0 0 020200 001000 0 000C00 001000
10(000A) 0 000000 000000 THRU 11(000B)
12(000C) 0 000000 010000 0 000300 000000 0 000000 000000 0 808308 240001 0 000000 C00000
17(0011) 0 000000 000082 0 800000 000000 0 000000 000000 0 000000 000000 0 000000 000000
22(0016) 0 000000 00001D 0 000000 00001F 0 000000 000021 0 140103 06C5D7 0 F4F1F9 F504C4
27(001B) 0 C1E3C1 04C6C9 0 D3C500 000000 0 0C0101 08C4C1 0 E3C1C5 C9D3C5 0 040102 110000
32(0020) 0 020000 000000 0 080101 04C4C9 0 E2D200 000000 0 000000 000000 0 000000 000000
37(0025) 0 000000 000000 THRU 47(002F)

```

(SPECIFIED ATTRIBUTES: TITLE, KIND, BLOCKSIZE, MAXRECSIZE, MYUSE, BUFFERS, INTMODE, UPDATEFILE, INTNAME, UNITS, NEWFILE, FILEUSE, FILEORGANIZATION)

```

=====
11(0B) 5 C00000 A43BB2 IOAREA
12(0C) 5 C40016 81B80C BUFFDESC
13(0D) 0 000000 000000 SIOAREA
14(0E) 5 000001 420001 FMTBUFFDESC
15(0F) 0 000000 000000 FMTLOCK
16(10) 0 000000 00733D FILIOTIME
17(11) 7 382000 00933D PWRITES (54804000)
18(12) 7 38200A 00930C PREADS (54285200)
19(13) 7 382000 00933D PWRITEN (54804000)
20(14) 7 382000 00936A PREADN (56582900)
21(15) 7 382000 009368 PSEEK (56598650)
22(16) 1 412000 80037B PINITIAE
23(17) 7 38201B 309308 PSEARCH (55174800)
24(18) 7 382000 009328 PLOCKER (54227250)
25(19) 7 382011 809308 PRELEASE (55070000)
26(1A) 7 38200F F0930C PMOVEOUT (54491400)
27(1B) 7 382010 B0930C PMOVEIN (54492800)
28(1C) 7 382032 709308 PWAIT (55495800)
29(1D) 7 382000 009373 PFLOAT (56863900)
30(1E) 0 011F00 360042 PCWCONTROL (54283600) (54256600) (54657600)
31(1F) 0 000000 000168 OFFSET
32(20) 0 000000 000003 RECORDCOUNT
33(21) 0 000000 000006 BLOCKCOUNT
34(22) 0 000000 000000 LOWER
35(23) 0 000000 000002 UPPER
36(24) 0 000000 00005A MINRECSZ
37(25) 0 000000 00005A RECSIZE
38(26) 0 000000 000201 I
39(27) 0 000000 000000 T
40(28) 0 000000 000000 AEXP
41(29) 1 77D000 000092 DHEADER (SHOWN BELOW IN HEX)

```

```

=====
0(0000) 0 3F3F05 C0042A 0 0010C0 912000 0 040000 000000 0 003C00 00000F 0 335F5D AD1EB6
5(0005) 0 630014 0001F4 0 000001 000000 0 000202 E00000 0 01D02B 008000 0 002D00 000003
10(000A) 0 30263C 550156 0 30263C 550157 0 3EAA55 A41F33 0 000000 000000 0 050003 320092
15(000F) 0 000000 000000 THRU 17(0011)
18(0012) 0 080600 0180FA 0 000000 000000 0 000000 000000 0 000000 000000 0 000000 000000
23(0017) 0 000000 000000 THRU 37(0025)
38(0026) 0 800000 000000 THRU 39(0027)
40(0028) 0 1D0304 08C3D6 0 D5C3C5 D7E3E2 0 06C5D7 F4F1F9 0 F504C4 C1E3C1 0 04C6C9 D3C500
45(002D) 0 BDAF44 0E958B

```

```

=====
42(2A) 0 0010E0 000002 FIBEOF (EOFU=270, EOFV=2, LASTRECORD=6)
43(2B) 0 001000 0180FA ACTNUM
44(2C) 0 000000 000000 SBLCKING

```

```

45(2D) 0 000000 000000 SIOINFO
46(2E) 0 000000 000000 SOFFSET
47(2F) 0 000000 000168 CURRENTBLOCK
48(30) 2 000000 000000 FILEEVENT1
49(31) 2 000000 000000 FILEEVENT2
50(32) 0 000000 000000 OUTPUTTRANSLATION
51(33) 0 000000 000000 INPUTTRANSLATION
52(34) 0 000000 000000 USERROUTINES
53(35) 0 000000 000000 FLOPPYMISC (NORMALIOLENGTH=0)
54(36) 0 000000 000000 FIBLOCKSNR
55(37) 5 C00001 E1CB2A IOCB
      MLIP CNTRL: 0 10CB00 000C28 READ, WORD, CAUSE IO FINISH
      DLP ADDRESS: 0 000000 100005 HDPNO = 1, HDP PORT = 0, LEM PORT = 0, REL LCP = 5
      CMND QUEUE: 5 C00000 80F7CA
      SELF PTR: 5 C00001 E1CB2A
      COMMAND PTR: 5 E00001 91CB2A
      RESULT PTR: 5 E00001 B1CB2A
      C/R LENGTHS: 0 000000 0A000C
      RESULT MASK: 0 000000 000000
      RSLT QUEUE: 5 E00000 10F4E2
      NEXT LINK: 0 000000 000000
      DATA PTR: 5 E00003 C3C506
      CURR LENGTH: 0 000000 000000
      MLIP RESULT: 0 000000 000000
      START TIME: 0 00055A B4FFAB
      FINISH TIME: 0 000000 001C80
      INFO1 WORD: 0 332332 040008 UNITNUMBER = 51, OWNER STKNO = 332, INITIATING STKNO = 332,
      REQUESTOR = 1 (USERIO)
      INFO2 WORD: 0 000030 011100 STATE = 1 (INACTIVE), PATHNAME = 03, PROC NO = 1, HDP NO = 1
      LOG DESC: 0 000000 000000
      IOCW: 0 330000 0180FC IOSTANDARDFIELD: ATTENTION, READ, 8-BIT, MEMORY PROTECT
      BUFFER DESC: 5 C00003 C3C506
      EVENT REF: 5 E10000 243BB2
      FIB DESC: 5 C00004 42EB01
      IO MASK: 0 000000 000000
      LOGICAL RD: 0 000000 000003 UNITS XFERRED = 00000 (0), ATTENTION, EXCEPTION
      IO INFO: 0 800003 C00000 WORD, BUFFERINDEX = 00000 (0), IOLENGTH = 0003C (60)
      COMMAND1: 0 807000 0180FC
      COMMAND2: 0 000000 000000
      RESULT1: 0 000000 000000
      RESULT2: 0 000000 000000
      SAVED INFO: 0 330000 000033
56(38) 0 000000 000004 TRANSACTIONCOUNT
57(39) 0 000000 000000 LIBRARYINFO
58(3A) 0 000002 000000 PHYSICALIOCOUNT (READ=2, WRITE=0)
59(3B) 0 24B666 6E0455 LOGINTSTARTTIME
60(3C) 5 C00004 42EB01 SELFDESC
61(3D) 0 000000 000201 BUFLINKS
62(3E) 5 800000 22F897 IOMOM (BUFFER #1, IS THE TOP BUFFER, BUFFER POOL SIZE = 2)
      (BUFFER POOL DOPE VECTOR).....
      0(0000) 5 800000 A2A503 5 800000 A43BB2
      (BUFFER #0) .....
      IOCBDESC: 5 800001 E4D915
      .....
      MLIP CNTRL: 0 10CB00 000C28 READ, WORD, CAUSE IO FINISH
      DLP ADDRESS: 0 000000 100005 HDPNO = 1, HDP PORT = 0, LEM PORT = 0, REL LCP = 5
      CMND QUEUE: 5 C00000 80F7CA
      SELF PTR: 5 C00001 E4D915
      COMMAND PTR: 5 E00001 94D915
      RESULT PTR: 5 E00001 B4D915
      C/R LENGTHS: 0 000000 0A000C

```





0013           3 000000 002000 RCW: DUMMY (RUN)  
0012 ----D[02]=>3 F81B74 E08011 \*MSCW: PREVIOUS MSCW @ 0001; D[01]=B74E IN STACK 381  
0000 = BOSR (0003CAE4)

```

%%%%% ALGOL PROGRAM WITH PROCEDURES FOR OPTIONAL LAB 4 %%%%%
$ SET LIST STACK LINEINFO $
BEGIN
  REAL V1, V2;
  %%%
  PROCEDURE A;
  BEGIN
    REAL V3;
    PROCEDURE B;
    BEGIN
      V3 := 3;
      V1 := V3/V2 + V2/V3;
    END OF PROC B;
  B;
  END OF PROC A;
  %%%
  PROCEDURE C;
  BEGIN
    REAL V4;
    PROCEDURE D;
    BEGIN
      REAL V5;
      V4 := 4;
      V5 := 5;
      A;
      V2 := V4;
    END OF PROC D;
  D;
  END OF PROC C;
  %%% OUTER BLOCK
  C;
  END OF PROGRAM.

```

```

0000050
00000100
00000200
00000300
00000350
00000400
00000500
00000600
00000700
00000800
00000900
00001000
00001100
00001200
00001300
00001400
00001500
00001600
00001700
00001800
00001900
00002000
00002100
00002200
00002300
00002400
00002500
00002600
00002700
00002800
00002900
00003000

```

O B J E C T / E P 4 1 9 5 / L A B 4 / D U M P O N D I S K  
 = = = = =

C-65

```

        %%%% ALGOL PROGRAM WITH PROCEDURES FOR OPTIONAL LAB 4 %%%%
        BEGIN
(01,0002) = BLOCK#1
(01,0003) = SEGMENT DESCRIPTOR
                REAL V1, V2;
(02,0002) = V1
(02,0003) = V2
        %%%%
(02,0004) = A      PROCEDURE A;
                BEGIN
(01,0004) = SEGMENT DESCRIPTOR      REAL V3;
(03,0002) = V3      PROCEDURE B;
(03,0003) = B      BEGIN
                V3 := 3;
                V1 := V3/V2 + V2/V3;
                END OF PROC B;
                B;
                END OF PROC A;
        %%%%
(02,0005) = C      PROCEDURE C;
                BEGIN
(01,0005) = SEGMENT DESCRIPTOR      REAL V4;
(03,0002) = V4      PROCEDURE D;
(03,0003) = D      BEGIN
                REAL V5;
(01,0006) = SEGMENT DESCRIPTOR
(04,0002) = V5      V4 := 4;
                V5 := 5;
                A;
                V2 := V4;
                END OF PROC D;
                D;
                END OF PROC C;
        %%%% OUTER BLOCK
        C;
        END OF PROGRAM.
(01,0007) = BLOCKEXIT
    
```

```

00000050 000:0000:0
00000200 000:0000:0
BLOCK#1 IS SEGMENT 0003
1 00000300 003:0000:1
00000350 003:0000:1
00000400 003:0000:1
00000500 003:0000:1
00000600 003:0000:1
A IS SEGMENT 0004
2 00000700 004:0000:1
00000800 004:0000:1
00000900 004:0000:1
00001000 004:0001:0
00001100 004:0003:0
00001200 004:0003:1
00001300 004:0003:5
A(004) LENGTH IN WORDS IS 0007
2 00001400 003:0000:1
00001500 003:0000:1
00001600 003:0000:1
00001700 003:0000:1
C IS SEGMENT 0005
2 00001800 005:0000:1
00001900 005:0000:1
00002000 005:0000:1
D IS SEGMENT 0006
3 00002100 006:0000:1
00002200 006:0001:0
00002300 006:0001:5
00002400 006:0002:3
00002500 006:0003:2
D(006) LENGTH IN WORDS IS 0005
3 00002600 005:0000:1
00002700 005:0000:5
C(005) LENGTH IN WORDS IS 0004
2 00002800 003:0000:1
00002900 003:0000:1
00003000 003:0000:5
    
```

A SERIES AND B 5000/B 6000/B 7000 CONCEPTS  
 COMPILE LISTINGS AND PROGRAM DUMPS

=====

NUMBER OF ERRORS DETECTED = 0.  
NUMBER OF SEGMENTS = 5. TOTAL SEGMENT SIZE = 24 WORDS. CORE ESTIMATE = 37 WORDS. STACK ESTIMATE = 13  
PROGRAM SIZE = 31 CARDS, 70 SYNTACTIC ITEMS, 18 DISK SECTORS.  
PROGRAM FILE NAME: OBJECT/EP4195/LAB4/DUMP ON DISK. B5/6000 CODE GENERATED.  
COMPILATION TIME = 3.198 SECONDS ELAPSED; 0.510 SECONDS PROCESSING; 0.885 SECONDS I/O.

=====



MCP 36.140.3025: \*SYSTEM/MCP36140. INTRINSICS: SYSTEM/INTRINSICS ON DISK. (LOADED)  
SYSTEM SERIAL: #2372 HOSTNAME: SYSEDB6900. GROUP ID: DEFAULT.

CAUSE OF DUMP: FAULT TERMINATION @ 004:0001:2 (00001000), 004:0003:5 (00001200), 006:0002:3 (00002300), 005:0000:5 (00002600).

PIB HISTORY WORD AT ENTRY TO PROGRAM DUMP: 0 000000 010404.

PROGRAMDUMP OPTIONS: (DEFAULT)

033A = LOSR (00029D93)

0039 (01,0008) 5 C00045 A29A59 DESC [PRESENT-COPY]: DATA, LENGTH=1114 (POINTS @ OFFSET=0000 IN THIS STACK)  
0038 (01,0007) 0 000000 000000  
0037 (01,0006) 0 000000 000039 OP: OCT:00000000 00000071 , EBC:??????, DEC:57  
0036 (01,0005) 0 000000 00033C OP: OCT:00000000 00001474 , EBC:??????, DEC:828  
0035 (01,0004) 0 000000 000000  
0034 (01,0003) 0 FFFFFFFF FFFFFFFF OP: OCT:77777777 77777777 , EBC:??????, DEC:-7.0064923216E-46  
0033 (01,0002) 0 000000 000002 OP: OCT:00000000 00000002 , EBC:??????, DEC:2  
0032 3 000A26 E84FA3 RCW: LL=01, CNTRL STATE [MCP SEGMENT @ OFA3:026E:5 (11384800)]  
SEG DESC: 3 A00027 0016DF  
CODE: 3 61ADAB B23595 3 B840CB A5B261 3 A6B7BD 9E27B6 >3 B8AE40 3AABB5< 3 A3A3FF FFFFFFFF  
0031 ----D[01]=>3 C12000 804008 \*MSCW: PREVIOUS MSCW @ 0029; D[00]=0008 IN STACK 012  
0030 (01,0007) 7 336625 608FA3 PCW: LL=02, D[0] SEGMENT @ OFA3:0256:3, NORML STATE  
002F (01,0006) 0 000000 000001 OP: OCT:00000000 00000001 , EBC:??????, DEC:1  
002E (01,0005) 0 000000 000001 OP: OCT:00000000 00000001 , EBC:??????, DEC:1  
002D (01,0004) 0 000000 000000  
002C (01,0003) 0 334520 083020 OP: OCT:14642440 02030040 , EBC:??????, DEC:1.42947407974E-23  
002B (01,0002) 0 000009 000004 \*\*\*\*\* DIVIDE BY ZERO \*\*\*\*\*  
002A 3 000400 112004 RCW: LL=04, NORML STATE [USER SEGMENT @ 0004:0001:2 (00001000)]  
SEG DESC: 3 800000 7B1A81  
CODE: 3 FFB203 7002B9 >3 100383 100330< 3 028380 5002B8  
0029 ----D[01]=>3 C12000 804003 \*MSCW: PREVIOUS MSCW @ 0026; D[00]=0008 IN STACK 012  
0028 (04,0002) 0 000000 000003 OP: OCT:00000000 00000003 , EBC:??????, DEC:3  
0027 3 000A00 30E004 RCW: LL=03, NORML STATE [USER SEGMENT @ 0004:0003:5 (00001200)]  
SEG DESC: 3 800000 7B1A81  
CODE: 3 FFB203 7002B9 3 100383 100330 3 028380 5002B8 >3 A3AE70 03ABA3< 3 B0BFFF FFFFFFFF  
0026 ----D[04]=>3 736002 210004 \*MSCW: PREVIOUS MSCW @ 0022; D[03]=0022  
0025 (03,0003) 7 336200 012004 PCW: LL=04, D[1] SEGMENT @ 0004:0000:1, NORML STATE  
0024 (03,0002) 0 000000 000003 OP: OCT:00000000 00000003 , EBC:??????, DEC:3  
0023 3 000600 212006 RCW: LL=04, NORML STATE [USER SEGMENT @ 0006:0002:3 (00002300)]  
SEG DESC: 3 800000 5CF77D  
CODE: 3 FFB204 7002B8 3 B20548 02B8AE >3 5004AB 300250< 3 0388A3 B0B4A2  
0022 ----D[03]=>3 736001 40C003 \*MSCW: PREVIOUS MSCW @ 001F; D[02]=0014  
0021 (04,0002) 0 000000 000005 OP: OCT:00000000 00000005 , EBC:??????, DEC:5  
0020 3 000A00 00E005 RCW: LL=03, NORML STATE [USER SEGMENT @ 0005:0000:5 (00002600)]  
SEG DESC: 3 800000 487EBD  
CODE: >3 FFAE70 03ABA3< 3 B0BFFF FFFFFFFF  
001F ----D[04]=>3 736001 B10004 \*MSCW: PREVIOUS MSCW @ 001B; D[03]=001B  
001E (03,0003) 7 336600 312006 PCW: LL=04, D[1] SEGMENT @ 0006:0003:3, NORML STATE  
001D (03,0002) 0 000000 000004 OP: OCT:00000000 00000004 , EBC:??????, DEC:4  
001C 3 000A00 00A003 RCW: LL=02, NORML STATE [USER SEGMENT @ 0003:0000:5 (00002900)]  
SEG DESC: 3 800000 807777

C-67

```

CODE: >3 FFAE50 05ABAE< 3 6007AB A3B0B0
001B ----D[03]=>3 736001 40C007 *MSCW: PREVIOUS MSCW @ 0014; D[02]=0014

001A (02,0006) 6 800000 000000 SCW:
0019 (02,0005) 7 336000 10E005 PCW: LL=03, D[1] SEGMENT @ 0005:0001:0, NORML STATE
0018 (02,0004) 7 336000 40E004 PCW: LL=03, D[1] SEGMENT @ 0004:0004:0, NORML STATE
0017 (02,0003) 0 000000 000000
0016 (02,0002) 0 000000 000000
0015          3 00024E E88FCD RCW: LL=02, CNTRL STATE [MCP SEGMENT @ OFCD:04EE:1 (16562000)]
SEG DESC: 3 8800B6 ABCAC1
CODE: 3 BEFFFF FFFFFFF 3 4A6870 C30C30 3 ABA3AE 400AAB >3 A3B234 B22695< 3 B8B180 95B9A2
0014 ----D[02]=>3 F37000 408002 *MSCW: PREVIOUS MSCW @ 0012; D[01]=0004 IN STACK 337

0013          3 000000 002000 RCW: DUMMY (RUN)
0012 ----D[02]=>3 F81B74 E08011 *MSCW: PREVIOUS MSCW @ 0001; D[01]=B74E IN STACK 381

0000 = BOSR (00029A59)

```

## **BIBLIOGRAPHY**

## A SERIES, LINE 5 5000/B 6000/B 7000 CONCEPTS

### BIBLIOGRAPHY

Manual	Form Number
A Series Systems An Introduction	1169562
A Series Menu-Assisted Resource Control (MARC) User's Guide 3.6	1169588
A Series Screen Design Facility (SDF) Capabilities 3.6	1180437
A Series Advanced Data Dictionary System (ADDS) Capabilities 3.6	1180569
A Series Communications Management System (COMS) Capabilities 3.6	1180494
A Series Interactive Data Comm Configurator (IDC) User's Guide 3.6	1169810
A Series Extended Retrieval With Graphic Output (ERGO) User's Guide 3.6	1164027
A Series Disk Subsystem Software Overview 3.6	1169992
A Series Mark 3.6.0 System Software Installation Guide	1170040
A Series A 3 System Software 3.6 Installation Guide	1169679
A Series A 9 System Software 3.6 Installation Guide	1169695
A Series A 10 System Software 3.6 Installation Guide	1169935
A Series I/O Subsystem Reference 3.6	1169984
A Series System Software Utilities Reference 3.6	1170024
A Series Work Flow Language (WFL)	1169802
A Series System Architecture Reference Volume 2	5014954
A Series Print System (PrintS/ReprintS) User's Guide 3.6	1169919
A Series Printing Utilities User's Guide 3.6	1169950
A Series Operator Display Terminal (ODT) 3.6	1169612
A Series System Software Site Management Reference 3.6	1170008
A Series System Software Support Reference 3.6	1170016
A Series CANDE 3.6	1169869

## A SERIES AND B 5000/B 6000/B 7000 CONCEPTS

A Series CANDE Operations 3.6	1170065
A Series DMS II Data and Structure Definition Language (DASDL)	1163805
A Series DMS II Utilities and Operations Guide	1163839
A Series Advanced Data Dictionary System (ADDS) User's Guide	1180551
A Series Extended Retrieval with Graphic Output (ERGO) User's Manual	1164027
A Series DMS II Inquiry Software Operation Guide	1164035
A Series DMS II DataAid User's Guide	1180544
SMF II System Resource Management Manual	5015688
SMF II Site Management Manual	5012016
LINC II Reference Manual	1163961
Reporter III Report Language User's Guide	1177185
A Series MultiLingual System User's Guide	1169646
A Series ALGOL Test and Debug System (TADS) User's Guide	1169539
A Series COBOL74 Test and Debug System (TADS) User's Guide	1169901
A Series Burroughs Network Architecture (BNA) User's Guide	1169687
Office Management System (OMS II) Planning and Installation Guide	1153194
A Series Intelligent Distributed Editor (IDE) User's Guide	1182474
A Series Data Transfer System (DTS) User's Guide	1180320
A 3 System Reference	5013196
A 9 System Reference	5012305
A 10 System Reference	5016579
A 15 System Capabilities and Features	1182508
A 15 System Hardware Operational Guide	1183134
A 15 System Operating Guide	1182516

## **A SERIES AND B 5000/B 6000/B 7000 CONCEPTS**

<b>B 5900 System Reference</b>	<b>5011034</b>
<b>B 6900 System Reference</b>	<b>5010986</b>
<b>B 7900 System Operator's Guide</b>	<b>1182151</b>
<b>B 7900 System Hardware Operational Guide</b>	<b>1161353</b>
<b>B 7900 System Capabilities and Features Guide</b>	<b>1166170</b>

***LAB EXERCISES***

**A SERIES AND B 5000/B 6000/B 7000 CONCEPTS  
LAB EXERCISES**

This page left blank for formatting.



**A SERIES AND B 5000/B 6000/B 7000 CONCEPTS  
LAB EXERCISES**

**MARC LAB**

Objectives:            Use MARC menu choices, actions, and commands.  
                              Inquire into file existence and family substitution.

1.    Log on to MARC with the usercode and password provided by the instructor:  
  
\_\_\_\_\_
  
2.    Observe the Home Menu that is displayed. What is your MARC session number? \_\_\_\_\_
  
3.    Enter **INTRO** on the Choice line and transmit.
  
4.    Enter **MENUS** on the Choice line, to read about MARC menus.
  - a.    Use the + and - Actions to scroll through the Menus information. Read the text displayed. The Intro screen will appear at the end of the Menus text.
  
5.    Also read the information displayed for the **ACTION, COMND, and TYPE** choices. When you are finished reading, enter **HOME** on the Action line of the Intro screen to return to the Home Menu.
  
6.    Inquire into the existence of the file EP4195/SOURCE/ENROLLMENT, using the **FILE** menu and its successors, as illustrated in the Student Guide, Figures 3-4 through 3-7.
  - a.    What is the file creation date? \_\_\_\_\_
  - b.    How large is the file in disk segments? \_\_\_\_\_
  - c.    What is the complete title of this file?  
  
\_\_\_\_\_
  - d.    Enter **HOME** on the Action line to return to the Home Menu.
  
7.    Inquire into the existence of EP4195/SOURCE/ENROLLMENT again, using Choice Field Typeahead as shown in the Student Guide, Figures 3-8 and 3-9. Then return to the Home Menu.
  
8.    To inquire about the same file in Command Mode, enter **CO PD EP4195/SOURCE/ENROLLMENT** on the Action line. Observe the output, and return to the Home Menu.
  
9.    Do the inquiry in step 8 again, but omit the CO

**A SERIES AND B 5000/B 6000/B 7000 CONCEPTS  
LAB EXERCISES**

**MARC LAB, cont.**

10. Enter **FAMILY** on the Choice line to inquire into the family substitution statement currently in effect. Write down the family statement and return to the Home Menu.  

---
11. Use the **CHFAM** Choice to change your family statement to **FAMILY DISK = DISK ONLY**.
12. Inquire into the existence of **EP4195/SOURCE/ENROLLMENT** using one of the methods from steps 6, 7, 8, or 9 above. What is the response? Why?  

---
13. Use the **CHFAM** Choice on the Home Menu to restore the value of the family statement from step 10.
14. Determine the purpose of the **INFO** Choice on the Home Menu by reading the Help information.
  - a. Place the cursor over the word **INFO**, and depress the **SPCFY** key. Short Help will be displayed at the bottom of the screen.
  - b. Depress **SPCFY** again to display longer Help information.
  - c. Return to the Home Menu.
15. Use the **INFO** Choice on the Home Menu to inquire into attributes for the file **EP4195/SOURCE/ENROLLMENT**.
  - a. Why is the system able to locate the file now? \_\_\_\_\_
  - b. Enter **GO INFO** on the Action line, to go directly to the Info screen, and inquire into the file again.
  - c. Use Typeahead to do the same inquiry again, by entering **INFO EP4195/SOURCE/ENROLLMENT** on the Choice line.
16. Enter the Choice **BYE** to log off MARC.

# A SERIES AND B 5000/B 6000/B 7000 CONCEPTS LAB EXERCISES

## CANDE LAB

- Objectives:            Use basic CANDE commands to enter, edit, compile, and run a program.  
                          Use basic CANDE control commands.  
                          Use COMS windows and dialogs to manage two concurrent CANDE sessions.

### A. Basic CANDE commands

1. Log on to CANDE with the usercode and password provided by the instructor:  
\_\_\_\_\_.
2. Check to make sure that the family statement associated with your usercode is correct:  
\_\_\_\_\_.
3. Make a workfile called <your initials>/CANDELAB. The workfile type should be ALGOL. If another student has the same initials, agree on a unique naming convention.
4. Enter the ALGOL source below into the workfile using sequence mode.

```
100 $RESET LIST
200 BEGIN
300 REAL R1, R2, R3;
400 FILE OUTFILE (KIND = REMOTE);
500 R4 := 3;
600 R1 := 3;
700 K := 5;
800 R3 := 7;
900 R3 := R1 + K + R3;
1000 WRITE (OUTFILE, <X20, "HOORAY! I DID IT...R3 = ", X3,R10.2>,R3);
1100 END.
```
5. Compile the workfile. Are there errors? \_\_\_\_\_
6. Add this line to the workfile: 225 INTEGER K;
7. Display the workfile on the terminal.
8. Renumber the lines, starting at 100 and counting by 50. Display the file again.
9. Compile the workfile. Are there errors? \_\_\_\_\_
10. Remove line 350 from the file.
11. Compile the workfile. Are there errors? \_\_\_\_\_
12. Save the workfiles.
13. Execute the program that you compiled. What value is displayed for R3? \_\_\_\_\_

**A SERIES AND B 5000/B 6000/B 7000 CONCEPTS  
LAB EXERCISES**

**CANDE LAB, cont.**

14. What are the names of the files you have created?

\_\_\_\_\_

\_\_\_\_\_

15. What are the values of the file attributes below for the source and object files that you created?

Attribute	Source File	Object File
Maxrecsize	_____	_____
Blocksize	_____	_____
Area size	_____	_____
Lastrecord	_____	_____
Creationdate	_____	_____

B. CANDE Control Commands. Write the control command for each step and the response displayed.

1. Inquire into the most recent messages displayed for your session. \_\_\_\_\_

2. Determine the name of your station. \_\_\_\_\_

3. Determine your CANDE session number. \_\_\_\_\_

4. Inquire into the name of the executing MCP. \_\_\_\_\_

5. Inquire into the system time and date. \_\_\_\_\_

6. Determine the number of active tasks, compiles, and stations under CANDE.

\_\_\_\_\_

**A SERIES AND B 5000/B 6000/B 7000 CONCEPTS  
LAB EXERCISES**

**CANDE LAB, cont.**

C. COMS Windows and Dialogs

1. Enter ?ON MARC to move to the MARC/1 window.
2. Use the MARC menus as in the MARC Lab to inquire into files under the directory <your initials>.
3. Enter ?ON CANDE on the Action line to move back to the CANDE window.
4. Is the session number the same as before? \_\_\_\_\_
5. Enter ?ON CANDE/2 to move to dialog 2 of the CANDE window.
6. Is the session number the same as before? \_\_\_\_\_
7. Make a workfile called ADDRESS/<your initials> with the type SEQ for sequential data.
8. Enter your name and business address into the workfile. Use 3 or 4 lines, as if you were addressing an envelope. Sequence numbers are required.
9. Save the workfile.
10. Return to dialog 1 of the CANDE window.
11. What is the CANDE session number? \_\_\_\_\_
12. What is the name of the workfile now? \_\_\_\_\_
13. Does the file ADDRESS/<your initials> exist on your pack? Why or why not?  
\_\_\_\_\_
14. Use the ?WINDOWS command to inquire into the windows available.
  - a. What is the current window? \_\_\_\_\_
  - b. What is the status of the CANDE/2 window? \_\_\_\_\_
15. Log off both CANDE and MARC. This will end both CANDE sessions.

**A SERIES AND B 5000/B 6000/B 7000 CONCEPTS  
LAB EXERCISES**

This page left blank for formatting.

# A SERIES AND B 5000/B 6000/B 7000 CONCEPTS LAB EXERCISES

## CANDE/DUMPALL LAB

Objectives: Use additional CANDE commands such as Insert, Move, Find, Replace, Write, Back, Split, and Do.  
Obtain information about a file from DUMPALL.

### A. Additional CANDE Commands

1. Make a data file called <your initials>/ADDRESS/MASTER with the type SEQ.
  - a. Insert the individual address files created by all the students during the last lab into your master file. The individual addresses are under the ADDRESS directory. Leave a blank line between the addresses for readability.
  - b. After you have all the addresses inserted, move them into alphabetical order. If there are many students in class, this may become tedious. Move a few addresses for practice, and then continue with the next step.
2. Save your workfile, and get it again as <your initials>/ADDRESS/PRACTICE.
  - a. Use the Find command to locate all records containing the word **Burroughs**, and all records containing the word **Street**.
  - b. Use the Replace command to change all occurrences of **Street** to **Avenue** (if Street does not occur, select a word that does occur in your file and replace that word).
  - c. Save your workfile.
3.
  - a. Make a Do file containing commands to accomplish the following:  

```
Get <your initials>/ADDRESS/PRACTICE
Undo the replacement from step 2b above
Write the workfile
Save the workfile
```
  - b. Do the Do file, and observe the results.
4. Use the Back command to browse through the printer backup file created by the Write statement above.
  - a. Try options such as Help, First, Last, +, -.
  - b. When you are done browsing, remove the backup file and end the backupprocessor.

**A SERIES AND B 5000/B 6000/B 7000 CONCEPTS  
LAB EXERCISES**

**CANDE/DUMPALL LAB , cont.**

**B. SYSTEM/DUMPALL**

1. Run DUMPALL interactively to list records from <your initials>/ADDRESS/MASTER on the screen. Try options such as Teach, Hex, Record, Skip.
  
2. Run DUMPALL through MARC to print <your initials>/ADDRESS/MASTER in both alpha and numeric formats. End or split the session to release the printer backup file.
  
3. Use the DUMPALL listing printed above to answer the following questions.
  - a. Which student does record 6 pertain to? \_\_\_\_\_
  - b. What characters are in the third word of record 6? \_\_\_\_\_
  - c. What is your first name in EBCDIC? \_\_\_\_\_
  - d. How many words are in each record of the file? How many characters? \_\_\_\_\_
  - e. How many words are in each block of the file? How many characters? \_\_\_\_\_
  - f. What is the Filekind of this file? \_\_\_\_\_
  - g. How many records are in the file? \_\_\_\_\_
  - h. What is the value of the Lastrecord attribute? \_\_\_\_\_



**A SERIES AND B 5000/B 6000/B 7000 CONCEPTS  
LAB EXERCISES**

**DUMP LAB (OPTIONAL)**

Objective: Identify and interpret portions of a program dump for a program that failed. This optional exercise is designed for students who are interested in dump reading.

This exercise refers to the program listing on page C-64, the compile listing on pages C-65 and C-66, and the program dump on pages C-67 and C-68.

1. Name the procedures in which the following variables can be referenced.

V1 \_\_\_\_\_

V3 \_\_\_\_\_

V5 \_\_\_\_\_

2. How much memory is required to execute this program? \_\_\_\_\_

3. For each of the following variables, give the stack address and value at the time of the dump.

	Stack Address	Value
V1	_____	_____
V3	_____	_____
V5	_____	_____

4. What caused PCWs to be placed at (2,4) and (2,5) on the dump?

\_\_\_\_\_

5. What is the first executable statement in the program? \_\_\_\_\_

**A SERIES AND B 5000/B 6000/B 7000 CONCEPTS  
LAB EXERCISES**

**DUMP LAB (OPTIONAL), cont.**

6. Trace the execution of the program, and compare it to the dump, by completing the chart below for each procedure invoked.

Sequence Number	Procedure	Lex Level	Offset in Dump
<u>  200  </u>	<u>  BOT  </u>	<u>  2  </u>	<u> 0014 </u>
<u> 2900 </u>	<u>  C  </u>	<u>     </u>	<u>     </u>
<u>     </u>	<u>     </u>	<u>     </u>	<u>     </u>
<u>     </u>	<u>     </u>	<u>     </u>	<u>     </u>
<u>     </u>	<u>     </u>	<u>     </u>	<u>     </u>

7. Why does D[3] appear at offsets 001B and 0022 both? Where was D[3] pointing at the time of the dump?

8. a. What is the significance of the numbers 1000, 1200, 2300, 2600 after "Fault Termination" on page C-67 of the dump?

- b. These numbers are not printed on every dump. Why were they printed on this dump?

9. Why did this program fail?

**A SERIES AND B 5000/B 6000/B 7000 CONCEPTS  
LAB EXERCISES**

**WFL LAB**

Objectives:        Write and execute a simple WFL job.  
                      Use ADM and/or MARC to observe the execution of the job.  
                      Use ODT or MARC commands to inquire into system libraries.

A.    Write and start a WFL job that includes steps 1-9 below. Continue modifying and starting the job until it executes properly. Save your final job summary to show the instructor.

1.    Name the workfile and the job <your initials>/TEST/JOB.
2.    The job should be executed under your class usercode and password:  
  
\_\_\_\_\_
3.    The job should use the family substitution statement provided by the instructor:  
  
\_\_\_\_\_
4.    The job should run at priority 50.
5.    The file EP4195/SOURCE/ENROLLMENT should be under your usercode on your pack. Make a copy of this file under the name <your initials>/ROSTER/SOURCE.
6.    Compile <your initials>/ROSTER/SOURCE using the ALGOL compiler. The object should be called <your initials>/ROSTER/OBJECT.
7.    Execute <your initials>/ROSTER/OBJECT, the program compiled above. This program reads and prints a file whose internal name is INPUTDATA. For this execution, it should read and print from <your initials>/ADDRESS/MASTER, the name and address file used in previous labs.
8.    Remove the files <your initials>/ROSTER/SOURCE and <your initials>/ROSTER/OBJECT from your pack.
9.    Run the object program OBJECT/EP4195/WFLLAB/FINISH, which is already on your pack.

B.    Start your job again from your terminal or the ODT, and observe the execution of the job using ADM at the ODT or at a REMOTESPO, or using MARC at your terminal. Look for the information displayed when your tasks are active, waiting, and completed. Look for messages also.

**A SERIES AND B 5000/B 6000/B 7000 CONCEPTS  
LAB EXERCISES**

**WFL LAB, cont.**

**C. Library Inquiries**

1. Use the LIBS command through MARC or the ODT to determine which libraries are in the mix.
2. Use the SL command to inquire into the function names and associated library names.