Burroughs

Introduction to A Series Systems

(Relative to the Mark 3.6 System Software Release)

Copyright © 1986 Burroughs Corporation, Detroit, Michigan 48232

Distribution Code SE

Priced Item
Printed in U.S.A.
April 1986

1170057

Burroughs cannot accept any financial or other responsibilities that may be the result of your use of this information or software material, including direct, indirect, special or consequential damages. There are no warranties extended or granted by this document or software material.

You should be very careful to ensure that the use of this software material and/or information complies with the laws, rules, and regulations of the jurisdictions with respect to which it is used.

The information contained herein is subject to change without notice. Revisions may be issued to advise of such changes and/or additions.

Comments or suggestions regarding this document should be submitted on a Field Communication Form (FCF) with the Class specified as "2" (System Software), the Type specified as "1" (F.T.R.), and the Product specified as the seven-digit form number of the manual (for example, "1170057").

ABOUT THIS MANUAL

Welcome to A Series systems! Whether you have already acquired or are considering acquiring an A Series system, Burroughs wants you to understand all the many advantages and features of these systems. Burroughs A Series systems offer a complete environment to meet your information management needs, an environment that provides for information processing, control, and accessibility. You should be aware of what is available in this environment in order to tailor a system for your specific uses and to achieve maximum productivity.

To help you become aware of what you need to know, Burroughs has produced the "Introduction to A Series Systems." This is the manual to read first. It presents an overview of A Series systems and acts as a central source of information for the systems.

Because this manual covers so many individual subjects, it is not possible to discuss each one in depth. You can find more information in the sources mentioned in the text and in the A Series manuals published by Burroughs. The "Where to Find More Information" section describes the documentation available and how to order it.

WHO SHOULD READ THIS MANUAL

If you are a new A Series system customer, are considering acquiring a system, or just want a general introduction to A Series systems, then you should read this manual. In addition, because the A Series systems are functionally compatible with the B 5000/B 6000/B 7000 computer systems, anyone wanting an introduction to those systems should read this manual.

You do not need any prior knowledge of Burroughs systems to understand the information in the manual, but you should have a basic grasp of elementary computer concepts. Each section of the manual addresses a specific audience, and the presentation of the material in some sections assumes that you have read prior sections. The following paragraphs explain more about the use of the manual.

HOW TO USE THIS MANUAL

The manual is structured so that you can select and read only the sections that apply to your needs.

The first three sections of the manual are a general introduction to A Series systems. They should be read by anyone new to Burroughs systems or to A Series systems.

Section 1, "What's in an A Series System," introduces A Series systems and describes the complete package that you get when you acquire a system. Section 2, "Virtual Memory, Stacks, and Other System Concepts," gives you a short history of Burroughs large systems and explains concepts that form the building blocks for A Series systems. Section 3, "A User's View of System Functions," presents the system according to its use and as seen by the different types of users.

The next four sections are designed to help users find guidance on particular subjects. For maximum benefit, you should read Sections 1, 2, and 3 before reading these sections.

Section 4, "Planning for Effective Operations," is directed to system administrators and operators who are responsible for planning and implementing the initial and day-to-day operations of the system. Section 5, "Programming on A Series Systems," is written for both experienced and inexperienced programmers and explains issues that influence program efficiency.

"If You Have a Problem," Section 6, is for all users and explains what to do and where to go for help when the unexpected happens. Section 7, "Where to Find More Information," also for all users, lists available manuals by subject area, briefly describes them, and tells you how to order them.

The following table summarizes which sections each kind of user should read.

IF YOU ARE	THESE SECTIONS ARE OF PRIMARY INTEREST TO YOU
A customer	1, 2, 3, 6, 7
A MIS manager or other company manager	1, 2, 3, 6, 7
An operator	1, 2, 3, 4, 6, 7
A programmer	1, 2, 3, 5, 6, 7
A system administrator	All sections

CONTENTS

1 **WHAT'S IN AN A SERIES SYSTEM** **1**
A SERIES SYSTEM HARDWARE 3
Processor Subsystem. 4
Memory Subsystem 4
Input/Output (I/O) Subsystem 5
Maintenance Subsystem. 6
Updating Equipment 6
A SERIES SYSTEM SOFTWARE 8
Essential System Software. 8
Optional System Software 9
Updating System Software 11
SYSTEM DOCUMENTATION 13
Printed Documentation. 13
On-line Documentation. 14
SUPPORT FOR BURROUGHS SYSTEMS. 15
Hardware Equipment Support 15
Software Products Support. 16

2 **VIRTUAL MEMORY, STACKS, AND OTHER SYSTEM CONCEPTS.** **19**
HOW THE SOFTWARE IS ORGANIZED. 19
System Software. 20
Productivity Software. 22
Application Software 23
LANGUAGES. 24
TASKS AND JOBS 25
MULTIPROGRAMMING AND MULTIPROCESSING 27
STACKS AND PROCESSES 29
VIRTUAL MEMORY 31
INTERACTIVE AND BATCH PROGRAMS 33
LIBRARIES. 34
CONFIGURATION, RECONFIGURATION, AND SOFT CONFIGURATION . . 36
A SHORT HISTORY OF BURROUGHS SYSTEMS 37

3 **A USER'S VIEW OF SYSTEM FUNCTIONS.** **41**
Menu-Assisted Resource Control (MARC). 41
MESSAGE CONTROL SYSTEMS. 43
Communications Management System (COMS) 44
Command AND Edit (CANDE) Language. 46
PROGRAMMING LANGUAGES. 49
Standard Languages 49
Combining Programs in Different Languages. 51
JOB AND TASK MANAGEMENT. 52
Controlling Jobs And Tasks 53
Billing. 54
Tasking. 54
Restarting Jobs. 55
FILE MANAGEMENT. 56
File Types and Names 56
File Structure 58

Disk File Maintenance. 58
The DUMPALL File Maintenance Utility 59
Cataloging 60
Files on Tape. 61
DATA MANAGEMENT. 62
Data Management System II (DMSII) 62
DataAid. 63
Extended Retrieval with Graphic Output (ERGO) 63
Advanced Data Dictionary System (ADDS). 64
DM Interpreter 65
Database Certification (DBCERTIFICATION). 66
The DB MONITOR Utility 66
The DBANALYZER Utility 67
The INQUIRY Utility. 67
DATA COMMUNICATIONS. 68
Data Communications Hardware 68
Data Communications Software 72
Data Communications Utilities. 74
BURROUGHS NETWORK ARCHITECTURE (BNA) 75
How It Works 76
Using BNA. 78
OPERATIONAL INTERFACE. 79
Direct Entry of System Commands. 79
SECURITY 80
Software Access Control. 80
Network Security 83
Compiler Security. 84
INPUT/OUTPUT SUBSYSTEM 85
Physical I/O 85
Logical I/O. 87
Input/Output Operations. 87
File Attributes. 89
DISK SUBSYSTEM 90
Mirrored Disk. 94
PRINT SYSTEM 97
Controlling the Print System 98
INTRINSICS AND LIBRARIES 100
SORT Intrinsic 101
SYSTEM PERFORMANCE MONITORING. 102
System Management Facility II (SMFII). 102
The BARS Utility 105
LOGGER and LOGANALYZER 105
The SUSPENDER Utility. 106
The Utilization Command. 106
ODT Commands 107
GETSTATUS, SETSTATUS, and SYSTEMSTATUS 108
The MARC Menu System 108
SOFT RECONFIGURATION 109
ASD Extended Memory. 109
Memory Partitioning. 110
Sharing of I/O Subsystems. 111
The Configuration File 112
The Configuration Utility. 113

Reconfiguring the System 113

4 PLANNING FOR EFFECTIVE OPERATIONS. 115

4.1 INITIAL PLANNING 115

SYSTEM CONFIGURATION 115

PROGRESSION AND CONVERSION 116

PROGRESSION FROM BURROUGHS SYSTEMS 117

4.2 PLANNING FOR BASIC SYSTEM OPERATION. 118

BASIC SYSTEM INITIALIZATION. 118

AFTER COLD START 120

SOFTWARE CONFIGURATION 121

ASD Systems Memory Management. 121

ASN and GLOBAL/Tightly-Coupled Systems 121

Setting Memory Factors 122

Setting System Options 124

Setting Up User Identification 125

Setting Up Security. 125

Distributing Files on Disk 126

Setting Up Supervisor Programs 127

Setting Up Job Queues. 128

Setting Up Data Communications 128

Setting Up a Multilingual System 131

4.3 PLANNING DAY-TO-DAY OPERATIONS 132

KEEPING RECORDS. 132

MONITORING SYSTEM PERFORMANCE. 134

MANAGING JOBS. 135

MAINTAINING DISK FILES 137

MANAGING BACKUPS 138

4.4 PLANNING FOR EXCEPTIONAL OPERATIONS. 139

MEMORY DUMPS 139

RECONFIGURING THE SYSTEM 140

UPDATING THE SOFTWARE. 141

PTD CONFIDENCE AND DIAGNOSTIC TESTING. 142

5 PROGRAMMING ON A SERIES SYSTEMS. 143

5.1 PROGRESSION AND CONVERSION 143

AVAILABLE TOOLS. 143

CONVERSION OF PROGRAMS 144

CONVERSION OF DATA 145

5.2 PROGRAM INITIALIZATION, RECOVERY, AND RESTART. 146

APPLICATION PROGRAM INITIALIZATION 146

RECOVERY AND RESTART 147

Interactive Programs 147

Batch Programs and Jobs. 148

SUPERVISOR PROGRAMS. 149

5.3 ONGOING CONSIDERATIONS 150

DESIGN OF PROGRAMS 150

File and Data Structure. 150

The Effect of Program Structure. 152

General Practice 153

Program Maintenance. 154

PROGRAM DEVELOPMENT. 155

APPLICATION PROGRAM MANAGEMENT 157

Code File Compatibility. 157
Distribution of Files. 158
CONTROLLING PROGRAM CHANGES. 159
TESTING. 160
5.4 EXCEPTIONAL SITUATIONS 161
NEW RELEASES 161
DEIMPLEMENTATION OF OLD FEATURES 163

6 IF YOU HAVE A PROBLEM. 165
DEFINING THE PROBLEM 166
Make Quick Checks. 166
Ask Basic Questions. 167
Break the Problem Down 168
WHERE TO GO FOR HELP 170
Hardware Problems. 170
Software Problems. 170
User Groups. 171
REPORTING SYSTEM PROBLEMS AND SUGGESTIONS. 174
General Problems 174
Data Communications Problems 175
Data Management Problems 176
Compiler Problems. 176
Completing Field Communication Forms (FCFs). 177

7 WHERE TO FIND MORE INFORMATION 185
DOCUMENTATION LISTED BY SUBJECT. 185
Database Management. 185
Data Communications. 186
Languages. 186
Message Control Systems. 186
Multilingual Capabilities Management 187
Operations 187
Performance Monitoring 187
Peripherals. 187
Programming Productivity Aids. 188
Progression. 188
Reference Cards. 188
System Software. 188
DOCUMENTATION LISTED ALPHABETICALLY. 189
HOW TO ORDER DOCUMENTATION 203
Domestic (United States) Customers 203
International Customers. 203
DOCUMENTATION CONVENTIONS. 204
How to Read a Railroad Diagram 204
Constants. 205
Variables. 206
Required Items and Punctuation 207
Optional Items 207
Loops. 208

GLOSSARY. 213

INDEX 233

1 WHAT'S IN AN A SERIES SYSTEM

The A Series systems are the newest members of the Burroughs family of computer systems. With the A Series systems, Burroughs continues its commitment to provide sophisticated, easy-to-use, cost-effective, and compatible hardware and software. These systems offer additional features and enhanced performance while maintaining compatibility with the Burroughs B 5000/B 6000/B 7000 Series systems. Current users of the B 5000, B 6000, and B 7000 Series can grow into the A Series to obtain more features and performance without undergoing a conversion. The A Series systems support a full line of mainframe computers, general- and specific-purpose terminals, printers, document processors, and data storage devices.

Large-scale A Series configurations can act as central systems, managing on-line networks and large integrated databases. Smaller configurations can also act as central systems and, in addition, serve decentralized data processing needs by providing high levels of computer power at regional or departmental sites for transaction processing, data communications, and terminal network control.

The A Series systems provide state-of-the-art hardware technology in systems that span a wide range of operating power. A Series systems are designed to efficiently execute high-level languages like COBOL rather than low-level assembly or machine languages. The user can achieve the high productivity of third- and fourth-generation tools without sacrificing performance.

Through the use of advanced network and data management products, users can, with minimum effort, produce transaction-oriented data management systems and extensive on-line communication networks.

Burroughs A Series systems offer a complete package:

- Hardware equipment
- System software to run that equipment
- Documentation for both hardware and software
- Support services to help you obtain the optimum use of your system

The remainder of this section describes, at an overview level, the complete package that you receive. For more detailed descriptions, refer to the section "A User's View of System Functions" for software and to the section "If You Have a Problem" for support services. If you need answers to specific questions not covered by this manual, consult your Burroughs support or sales representative.

A SERIES SYSTEM HARDWARE

The hardware equipment for A Series systems is modular, to allow maximum ease and freedom of system expansion. The hardware and software together adjust system operation to efficiently use whatever hardware modules are available. This system-wide adjustment means the hardware configuration can change without requiring that application software or the operating system be changed or recompiled.

Each A Series system requires a minimum configuration of hardware modules in order to function. This configuration consists of the following modules:

- Processor
- Memory
- Input/output (I/O)
- Maintenance subsystem

Additional processor, memory, and I/O modules can be added to the minimum configuration to increase capacity and capability and to enhance performance. The total number of modules of each type in any particular system depends on the type of A Series system and the memory and I/O needs of the particular user site.

In addition to the minimum configuration modules, a selection of peripheral devices is available to customize the system to the needs of each site. These devices include peripherals such as hard disk drives, terminals, magnetic tape drives, card handlers, printers, and data communications equipment.

Most peripheral devices used on any A Series system are compatible with all other A Series systems, B 7900 systems, and in some instances, with peripherals currently used on other Burroughs systems.

Within the hardware system, one or more hardware modules of a particular type comprise a functional subsystem. The following topics explain the purpose and responsibilities of each subsystem.

Processor Subsystem

The processor subsystem consists of a series of processors that operate together to execute application and system programs. To be executed, programs must be translated into the particular instruction set understood by the processors. In the past, this instruction set was hard-wired into the machine that executed the task. It thus was a part of the hardware and not easily changed. On current systems, the instruction set is a program in the low-level machine that executes the application and systems program. Therefore it can be treated in much the same manner as any system program. Programs for this low-level machine are known as firmware, and the code that makes up the program is known as microcode.

On the A Series systems, most of the firmware is initially loaded into the system from a portable medium, such as tape or a floppy diskette, and stored on a hard disk. These firmware programs are in the form of object code. Firmware cannot be programmed or modified by customers.

Memory Subsystem

The memory subsystem provides storage and handles all transfers of data between main memory and the main processor. This subsystem consists of one or more memory control units and memory storage units, plus the memory interface, which may be part of a control unit.

Also within the memory subsystem is error detection and correction circuitry, which can correct any single-bit errors with no degradation in system performance. The logging of these correctable memory errors allows minor hardware problems to be caught before they have an adverse effect on the system.

A Series memory is composed of 6-byte (48-bit) words. Systems with the Master Control Program/Advanced Systems (MCP/AS) operating system can have up to 4 billion words (24 billion bytes) of directly addressable memory, depending on their hardware. Such a large amount of memory is made available by the Actual Segment Descriptor (ASD) memory subsystem, implemented by the MCP/AS.

The term "ASD" refers to the Actual Segment Descriptor that identifies locations in memory. The ASD memory architecture supports a single large memory structure that requires no partitioning. The ASD's architectural design is organized as a monolithic storage area totally available to all processors in the system. All allocated memory areas are accessed and controlled through a central structure called the Actual Segment Descriptor (ASD) table. The ASD table is a central repository of information for all memory in use by the system.

"ASD Extended Memory" in the section "A User's View of System Functions" explains how the ASD system works. The manual "Memory Subsystem Overview" provides more detailed information.

The memory architecture is different for systems running the MCP (not the MCP/AS). On smaller systems (those with up to six million bytes of memory), the entire memory is visible to all programs. The user need not be concerned with partitioning the memory system. To allow more than six million bytes, a partitioning mechanism called Address Space Number (ASN) is used. This scheme increases the physical memory available to all programs beyond the architectural maximum. "Memory Partitioning" in the section "A User's View of System Functions" explains how the ASN scheme works.

On some A Series systems, processor performance is enhanced through the use of data or code cache memory. A cache is a small component of memory used to store the contents of recently referenced memory locations. When a read request is received, the system checks the cache first for the data. Because cache memory access is much faster than main memory access, the read request is completed in less time if the data is found in the cache. Burroughs has optimized the memory design of A Series systems so that, today, the vast majority of all memory accesses can be serviced from a cache.

Input/Output (I/O) Subsystem

The input/output (I/O) subsystem manages all transfers of information between the operating system and peripheral devices. It consists of a series of specialized processors that are responsible for performing the actual transfers.

I/O operations are initiated by the operating system, but, once started, continue under the control of these specialized processor modules. This system of coordinated activities of multiple, independent hardware modules can be thought of as a form of distributed processing within each A Series system. Because the operating system shares control of I/O operations, multiple simultaneous I/O operations are possible and are a normal part of system operation.

The peripheral devices are connected to the system and controlled through Data Link Processors (DLPs), specialized processors that perform the peripheral-dependent functions required for I/O operations. Each DLP contains microcoded programs that accommodate the unique characteristics of the type of peripheral device it controls. A standard DLP exists for each type of peripheral device supported by A Series systems.

Maintenance Subsystem

The maintenance subsystem serves multiple purposes. It is the interface through which the operator controls the hardware when the hardware system is to be initialized, configured, or halted, or when system software is to be loaded for the first time.

The maintenance subsystem is also a means to diagnose hardware problems. Through this subsystem, an operator can load diagnostic code and standalone programs from the storage devices and run this software to test the components of the system.

The remote support system is connected to the hardware through the maintenance subsystem. Remote support enables system problems to be diagnosed through specialized programs run from a Burroughs central support center. Thus, when technical personnel arrive at a site, they can fix the problem efficiently with minimal loss of system time.

Updating Equipment

As part of an ongoing effort to further enhance performance and maintainability, Burroughs, from time to time, initiates design changes to the hardware equipment. These are known as Engineering Changes, and are provided to field engineers as Logic Improvement Notices (LINs) or Reliability Improvement Notices (RINs). A LIN describes a functional change that affects the operation of the product and is required for all applicable hardware. A RIN describes a change that improves the reliability of a product, but does not affect its functional operation. The change described by a RIN can normally be implemented on an as-needed basis and generally is not mandatory.

LINs and RINs can be installed independently. However, on A Series systems, they are usually combined into an Engineering Release Level (ERL) that can be installed as a kit. So as to minimize disruption to the system, LINs, RINs, and kits are usually incorporated into the system during preventive maintenance periods.

On occasion, Burroughs issues firmware update releases. These releases are handled in the same manner as software releases and require only that an operator load the files from the release media onto the appropriate disk or into the appropriate controller. Software releases are discussed later in this section.

A SERIES SYSTEM SOFTWARE

Essential System Software

System software performs overall control, traffic-directing, and housekeeping functions for the hardware and software in the system. Certain system software is essential for the operation of the system; the essential software is marketed under the title of the System Software Facility (SSF). Other system software is optional and can be ordered at any time when an installation needs the special functions they perform.

The System Software Facility product contents vary by system, but typically consist of the following programs:

- o The operating system, either the Master Control Program (MCP) or the Master Control Program/Advanced Systems (MCP/AS). The MCP/AS implements the Actual Segment Descriptor (ASD) extended memory (available for single-processor A 3, dual-processor A 3, A 9, and A 10 systems).
- o Work Flow Language (WFL), to construct jobs that compile or run programs on A Series systems
- o System libraries, to supply necessary and frequently used routines
- o The ALGOL compiler, to compile ALGOL programs and changes to system software
- o The DCALGOL compiler, to compile changes to the system software written in DCALGOL
- o The NEWP compiler, to compile changes to the MCP and other system software
- o The Binder program, to bind separately compiled subprograms to produce an executable program
- o The System Management Facility II (SMFII), to evaluate system performance
- o MARC (Menu-Assisted Resource Control), a modifiable menu interface to the computer system
- o PRINTS (Print System), deals with the output of files to printing devices such as image printers and line printers
- o Other products, depending on the type of system

The section "A User's View of System Functions" explains these programs and their functions in greater detail.

Optional System Software

Optional system software perform special functions that may not be needed on all systems. These programs can be ordered at any time. In the following list, the programs are grouped by functional area. When you are ready to order any of these software products, consult your Burroughs representative for prerequisites or concurrent product requirements. The section "A User's View of System Functions" explains many of these products in greater detail.

Message Control Systems

Message Control Systems (MCSs) are transaction-oriented interfaces to the computer system that support various forms of data communications processing.

- COMS (Communications Management System)
- CANDE (Command AND Edit language)
- GEMCOS (GEneralized Message COntrol System)
- RJE (Remote Job Entry)
- X25

Language Compilers

- APL/700 and APLB
- BASIC
- COBOL(68) and COBOL74
- FORTRAN(66) and FORTRAN77
- Pascal
- PL/I
- RPGII
- Sort
- DCALGOL

Productivity Aids

- TADS (Test And Debug System) - a high-level debugging aid. TADS versions are available for ALGOL, COBOL74, and FORTRAN77
- DES (Data Entry System) - a tool for large-volume data entry
- DTS (Data Transfer System) - a facilitator of data transfer between intelligent workstations and mainframes
- IDE (Intelligent Distributed Editor) - an environment for program development on intelligent workstations
- INFOVIEW (tm)* - a workstation/host system interface and windowing facility
- LINC II (Logic and Information Network Compiler II) - a program generator
- Editor - a text entry and manipulation program
- ADDS (Advanced Data Dictionary System) - a system for interactively designing and defining databases
- COMS (Communications Management System) - a communications monitor for transaction processing
- ERGO (Extended Retrieval with Graphic Output) - a program for retrieval and graphic display of information
- IDC (Interactive Datacomm Configurator) - a utility for on-line definition of a data communications network
- SDF (Screen Design Facility) - a tool for creating screens
- MLS (MultiLingual System) - a system for developing and accessing output messages, on-line help text, and menu screens in different natural languages, such as English, French, and Spanish
- Utility programs - programs that perform frequently needed tasks such as comparing files, merging patch files, and generating printouts of files

* INFOVIEW is a trademark of Burroughs Corporation.

What's in an A Series System

Information Management Systems

- o ADDS (Advanced Data Dictionary System)
- o DMSII (Data Management System II)
- o DMSII - Inquiry
- o Data Base Analyzer
- o Data Base Monitor
- o Data Dictionary
- o DM Interpreter
- o DB Certification
- o ERGO (Extended Retrieval with Graphic Output)
- o TPS (Transaction Processing System)

Network Management

- o BNA (Burroughs Network Architecture) - for distributed processing
- o NDLII (Network Definition Language II) - for terminal network support
- o DATACOMINFO File - contains a complete description of the datacomm configuration
- o IDC (Interactive Datacomm Configurator)

Reporting Systems

REPORTER III and On-Line REPORTER III - tools for retrieving, analyzing, and reporting information from a database

Updating System Software

Updates to system software products are made in the form of periodic software releases. Updates for major changes in the capability or design of the product are called Mark releases; updates for minor changes such as problem remedies are called support releases. Each release is identified by a three-part release level number, such as 3.6.0. When a major change to a product is made, the level number increases by 0.1.0; in this case, 3.6.0 becomes 3.7.0. For minor changes between major releases, the level number increases by 0.0.1; for example, 3.6.0 becomes 3.6.1.

After the Mark release, support releases will be provided periodically.

Each software release is referred to as the "Mark X.X.X" release, where "X.X.X" is the level number. At times, releases may be identified as "Mark X.X" rather than "Mark X.X.X", as in "the Mark 3.6 release." In this case, the reference includes the level 3.6.0 release and all support releases after it, up to, but not including, the level 3.7.0 release.

When the software displays the release level, it may show it as a number slightly different from the official release level for that software; for example, 35.270 rather than 3.5.3. The "different" number reflects a Burroughs internal release level for that particular software. These numbers are retained to allow precise identification of the content of the software for coordination of releases and for problem reporting.

When you purchase or lease a Burroughs system, you receive written notices of new releases. Updates are sent on machine-readable media compatible with your system. You are charged for the media, at the current price, plus shipping costs. If you want updates on disk packs, you may supply your own pack and eliminate the media charge. However, a handling charge is added to the mailing costs.

Because system software products are used in a wide variety of circumstances, users of these products occasionally encounter situations that cause a software fault. When a fault does arise, Burroughs may issue a software source patch to correct or avoid it. Source code patches require recompilation of the affected program.

Patches are issued either in printed form or on computer-readable media. Occasionally, in critical situations, personnel at a support center issue a patch over the telephone as a temporary solution.

SYSTEM DOCUMENTATION

Printed Documentation

Customers of Burroughs equipment and software products can choose from a complete range of supporting publications. These publications include manuals, guides, and technical information documents, such as Flashes and Technical Information Papers (TIPS). The section "Where to Find More Information" lists and describes the manuals and guides available.

A core group of manuals and guides arrives with the system. These documents contain information needed to start the system and install the software. The documents are shipped in the same boxes as the software.

All other documentation is available through the Literature Distribution Center in Detroit. The Customer Technical Publications Catalog lists the publications available to Burroughs customers. It is available free of charge from your Burroughs representative or from the Literature Distribution Center. When you request a catalog, also ask for a supply of the form for ordering documents, No. 3020003. The catalog contains instructions for using this form.

When you order software, you receive one copy of each of the manuals associated with the software. If you want additional copies, they can be ordered from the Literature Distribution Center.

Technical information documents provide timely solutions to software problems and keep customers current on the status of software corrections and updates. They are issued as needed and serve as temporary documentation until formal changes to product documentation are made.

- o Flashes are used to communicate either information about a patch or procedural advice for resolving software product problems.
- o Technical Information Papers (TIPS) are used to communicate technical information about installation or operational procedures.
- o Product Support Information Manuals inform users of the status of Burroughs software. They include new software features and their availability dates, new publications, an index of software Flashes, and a listing of known software problems and qualified solutions. A separate version of this manual is available for each product family. Refer to the Customer Technical Publications Catalog for ordering information.

On-line Documentation

In addition to printed publications, on-line documentation, in the form of help text, is available for certain Mark 3.6 and later software products. This on-line documentation is an integral part of the software products, and is accessed while using the products.

Help text is information explaining how to perform the tasks on a menu and how to fill in the fields on a screen. It also defines the terms shown on the screen. Two levels of help information are available. The first level consists of one or two lines displayed on the product screen you are currently viewing. The second level is a more detailed explanation of the same item and is displayed on a specially formatted Help screen. From this Help screen, you can return to the product screen or view additional help information.

SUPPORT FOR BURROUGHS SYSTEMS

Support for Burroughs systems is divided into two general areas: hardware equipment support and software products support.

Hardware Equipment Support

Hardware equipment support is both internal and external to the system. Internally, hardware designed specifically for maintenance is resident in the system and in each system module. This hardware, together with specialized software, makes up the maintenance subsystem, which can operate independently of the operating system. Through this subsystem, system status can be monitored, and diagnostic testing of the equipment, to the circuit board level, can be implemented. Results of this monitoring and testing are then used by external support personnel for both preventive and remedial maintenance of the equipment.

Externally, support is provided by field engineering personnel at customer sites and from Customer Service Centers (CSCs). Burroughs CSCs are regional service centers staffed by highly trained engineers who can, from the center, diagnose and isolate problems occurring at a customer site. This diagnosis is accomplished by establishing a communications link with the customer's system, running diagnostic programs, log analysis, and examining the status of certain hardware and software on that system.

Hardware equipment service is provided by Burroughs on a contractual or on an as-needed basis. Refer to the "Customer Guide to Burroughs Services and Support" or your Burroughs representative for details of the maintenance plans available.

If you are a domestic (United States) customer with a Product Support Agreement (PSA) and have an equipment problem, call the Customer Service Center on the toll-free number listed on the sticker on your equipment. These centers are in operation 24 hours a day, seven days a week. For all other domestic customers, contact your Burroughs service representative, as stated in your service agreement.

Software Products Support

Burroughs offers a complete set of services to support the software products it sells.

There are five levels of system software support available; the level suited to your site depends in part on the technical knowledge of your staff. When you purchase your software, your Burroughs representative will discuss with you the levels available for it.

For domestic (United States) customers, the full range of chargeable services offered includes the following items. Not all items are offered at every level.

- o Telephone assistance for
  - Operational questions and problems that are software associated
  - Software product information
  - Problem identification, advice, and follow-up
  - Arrangement for on-site service
- o On-site service for
  - Fault isolation, problem detours, fault report preparation
  - Installation of new software release levels
  - Extended day and holiday coverage, plus continuous on-site coverage
- o In-house analysis of fault reports, including responses and fixes
- o New software releases, with remedial changes when applicable

Reporting of software problems is the responsibility of the customer. (This applies to domestic customers only.) When a system is delivered, a supply of Field Communication Forms (FCFs) accompanies it. These forms are used to express comments about Burroughs products and report noncritical software-related problems to the Burroughs support centers. The section "If You Have a Problem" tells you how to fill out the forms.

In time-critical situations, telephone or Telex should be used to report problems. Often, this is followed up by use of an FCF. When reporting problems by telephone, first contact the Customer Service Center (CSC).

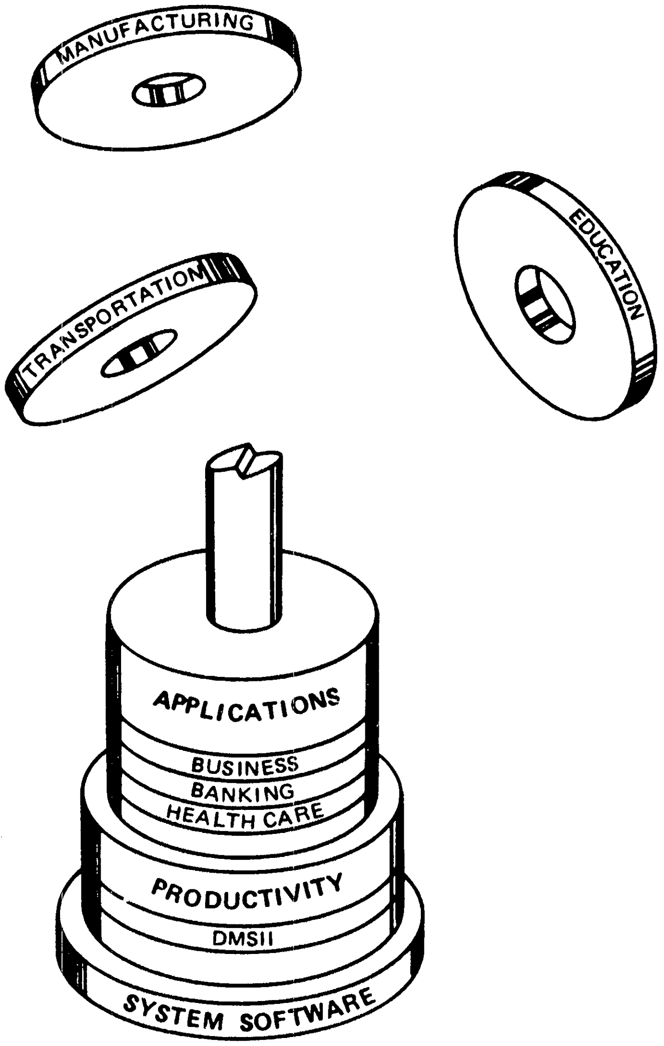
For application program support, Burroughs provides special toll-free Action lines. These lines tie in directly to the CSCs, each of which specializes in selected application lines of business. See your Burroughs system or marketing representative for the telephone numbers and contract fee for this service.

2 VIRTUAL MEMORY, STACKS, AND OTHER SYSTEM CONCEPTS

To fully understand A Series systems, you need to be familiar with certain concepts that form the building blocks for these systems. This section gives you an overview of those concepts. To help you place the concepts in perspective, the section ends with a short history of the evolution of large systems at Burroughs.

HOW THE SOFTWARE IS ORGANIZED

Software for A Series systems can be thought of as being structured in three functional layers, as shown in the following figure.



The first or foundation layer performs overall system control and so-called "housekeeping" functions, and is known as system software. This software consists of the operating system, which can be either the Master Control Program/Advanced Systems (MCP/AS) or the Master Control Program (MCP), the system utilities, and the language compilers.

The second layer, productivity software, consists of tools that help to minimize the time and cost of information retrieval and program development, change, and maintenance.

The third or top layer is application software, which can be special-purpose programs written by users or industry-specific programs supplied by Burroughs to perform frequently needed jobs.

System Software

System software determines how much of a hardware system's potential power is usable. Because the A Series hardware is designed to implement high-level languages, the system software, written in the high-level languages NEWP and ALGOL, can make full use of the power of the hardware. This, in turn, allows users to take maximum advantage of the system's potential.

There are two operating systems available to the A Series systems--the MCP/AS and the MCP.

MCP/AS is a new Burroughs operating system that can be used instead of the standard MCP. MCP/AS implements the Actual Segment Descriptor (ASD) extended memory system which allows MCP/AS to directly address over 4 billion words (24 billion bytes) of main memory, more memory than was previously addressable on any Burroughs system.

MCP/AS provides all the MCP features and is the operating system of choice for most A Series systems. MCP/AS implementation requires no program modification or recompilation, provided programs were recompiled under Release 3.4.1 or later, and Data Management System II (DMSII) databases have been updated to Release 3.5 or later.

Unless otherwise specified, descriptions of MCP features in this manual also apply to MCP/AS.

The MCP makes multiprogramming and multiprocessing a practical reality by automatically making optimum use of all system resources. Multiprogramming enables the system to run many jobs simultaneously. Multiprocessing is a state in which two or more processors in the same system run under the control of a single operating system. The MCP continually and dynamically assigns memory, manages input/output functions, schedules waiting jobs, communicates with the operator, logs system use, loads programs as needed, manages mass storage allocation, and supervises many other functions. Despite the complexity of the tasks it performs, the MCP remains efficient and easy to use.

The system utilities are programs that perform particular operations such as copying files, creating reports about files, creating guardfiles, and other functions. Utilities increase productivity by saving system users the task of writing programs to perform these housekeeping chores.

The programming language compilers provided by A Series systems translate the source language programs written in high-level languages into machine-language programs (object code) that can be read and executed by the hardware.

Productivity Software

Productivity software is an integrated set of tools to aid both information retrieval from the computer system and development of transaction-processing application programs. These tools are sophisticated yet easy to use, and accommodate varying levels of user experience. Productivity software makes it unnecessary for users to design applications around physical considerations such as database organization, network management, and program design--users are free to concentrate on the business function the application performs. This results in reduced development and maintenance costs and more efficient system utilization by programs.

Productivity software includes the following products. Many of these products are described more fully in the section "A User's View of System Functions."

- o InterPro (tm)* - Interactive Productivity Series, a synergistic family of software facilities to create new products and enhance existing ones. The following products are in the InterPro (tm) Series.
  - ADDS -- Advanced Data Dictionary System, for interactive database design and definition, plus management of all information entities
  - COMS - Communications Management System, a user-extendible system for handling transaction processing
  - ERGO - Extended Retrieval with Graphic Output, a tool for retrieval and graphic display of information
  - IDC - Interactive Datacomm Configurator, an on-line tool for defining and modifying a data communications network
  - MARC - Menu-Assisted Resource Control, a modifiable menu interface to the computer system
  - SDF - Screen Design Facility, for creating screens for on-line application systems
- o TADS - Test And Debug System, a high-level debugging aid for ALGOL, COBOL74, and FORTRAN77
- o BPS - Business Planning System, a business planning and analysis tool

* InterPro is a trademark of Burroughs Corporation.

- DMSII - Data Management System II, a multidimensional data management system
- DTS - Data Transfer System, for automatic transfer of data between a mainframe and an intelligent workstation
- Editor - a text entry and manipulation system
- Host-Link - a tool for expansion of the capabilities of the ET2000 intelligent workstation
- INFOVIEW (tm) - a workstation/host system interface and windowing facility
- IDE - Intelligent Distributed Editor, for program development on intelligent workstations
- LINC II - Logic and Information Network Compiler II, a tool that uses English-like terminology to generate custom programs
- REPORTER III and On-Line REPORTER III - report preparation programs that also retrieve and analyze the information

Application Software

Burroughs offers a large library of industry-specific application programs for:

general business	thrift industry
manufacturing	wholesale/distribution
health care	transportation
education	government
commercial banking	science/engineering

Because of the large amount of application software available, it is impractical to present an exhaustive list here. Your Burroughs representative has a complete listing of available programs.

LANGUAGES

Burroughs offers a selection of high-level computer languages for use in writing various types of programs. An assembler (machine level) language is neither needed nor provided, because the hardware has been designed for the efficient execution of high-level language programs.

The languages can be loosely divided into three groups: commercial languages, problem-oriented languages, and the ALGOL family of languages.

The commercial languages (all industry standard languages) and their general categories of use are the following:

- o APL/700 and APLB - numerical analysis applications
- o BASIC - general applications
- o COBOL(68) and COBOL74 - business applications
- o FORTRAN(66) and FORTRAN77 - scientific applications
- o Pascal - general applications
- o PL/I - combinations of scientific and business applications
- o RPGII - business applications

The ALGOL family of languages includes the following:

- o ALGOL - for system level and application programs
- o NEWP - for system level and application programs, as well as implementing operating systems, system libraries, and utilities
- o DCALGOL - for implementation and control of data communications and other programs that need access to the operating system interface

The problem-oriented languages have specialized uses:

- o DASDL - for describing the characteristics of a database
- o NDLII - for generating a network control program for data communications
- o Sort - for quick sorting of files
- o WFL - for managing the flow of tasks and jobs in the system

TASKS AND JOBS

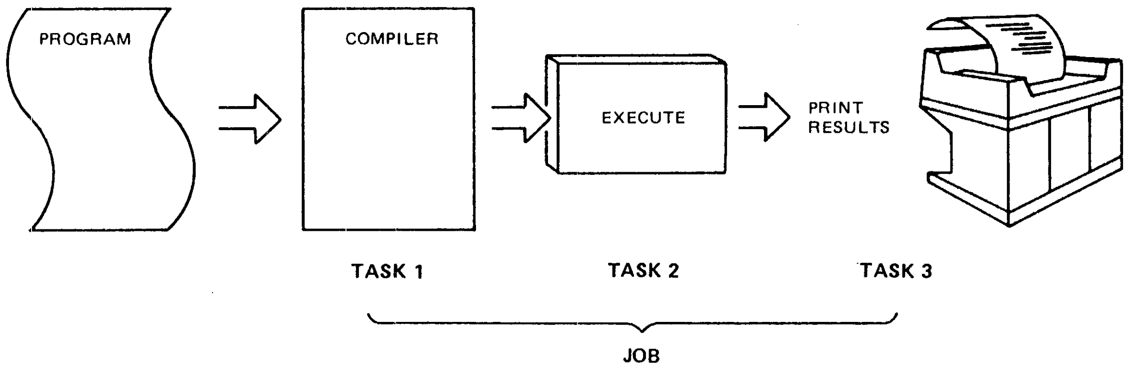
A task is a single, complete unit of work performed by the system. Examples of tasks are compiling or executing a program or copying a file from one disk to another. Tasks are initiated by a job, by another task, or directly by the user.

A job is one or more related tasks grouped together, as shown in the following example. Suppose you want to compile and run an application program and then view the output from the program; the system will perform the following tasks to perform this job:

1. Compile the program.

2. Execute the program.

3. Print the program output.



Together, these three tasks constitute a job. For a relatively simple job like this, you would very likely use a terminal to enter the sequence of commands one at a time, particularly if you don't do it very often. However, for more complicated or repetitious jobs, waiting at the terminal for a task to finish so that you can start the next task would be a waste of time. For these situations, it would be advantageous to enter all of the commands at once, as a job, and have the job execute independently while you work on something else.

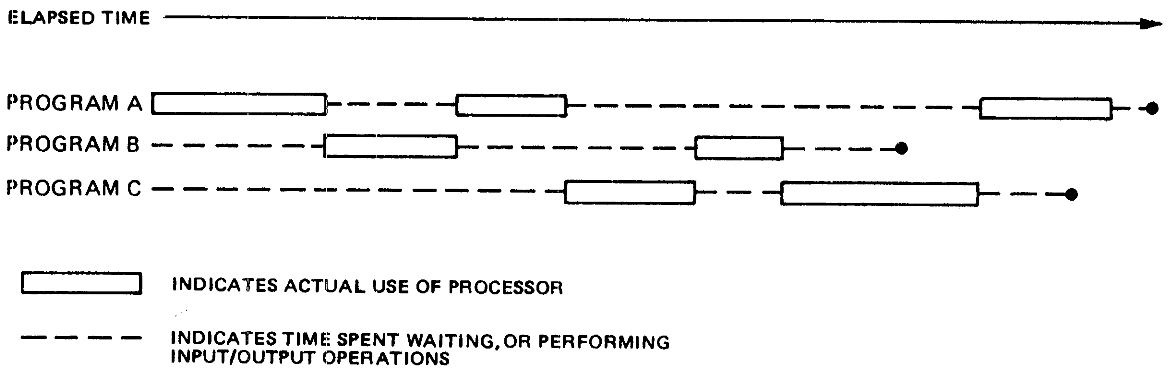
A Series systems allow you to do just that, through the use of Command AND Edit (CANDE) language and Work Flow Language (WFL) commands and statements. When you are using CANDE, you can start a job through entry of just one command, "DO myfile". CANDE then executes the sequence of CANDE commands contained in that file.

WFL is a true programming language and has its own compiler that produces a code file used in running the job. Programs written in WFL to execute jobs are both brief and straightforward, because the A Series operating system automatically performs functions that on other vendor's systems must be explicitly requested through complex job control statements. For frequently run jobs, you can store the WFL statements in a file and initiate the entire job with one command.

MULTIPROGRAMMING AND MULTIPROCESSING

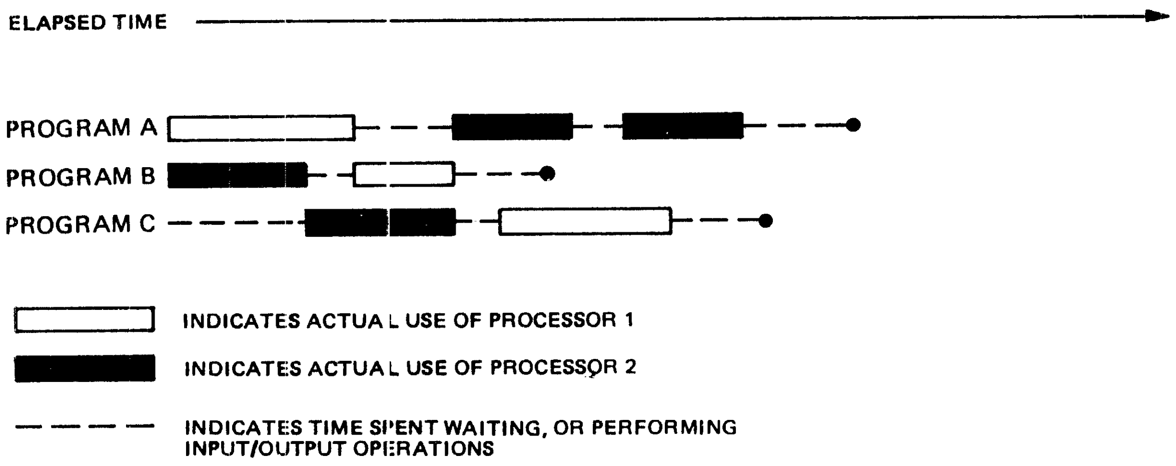
Multiprogramming and multiprocessing allow the system to make more efficient use of both processors and memory to significantly increase performance.

Multiprogramming means that the system has the ability to run many programs concurrently. Each program has the use of the processor for a limited period of time, often referred to as a time slice, to perform one task or part of a task. When that program reaches the end of its allocated time slice or is unable to proceed (for example, if it must wait for an I/O operation), the processor switches to another program.


ELAPSED TIME
PROGRAM A
PROGRAM B
PROGRAM C
INDICATES ACTUAL USE OF PROCESSOR
INDICATES TIME SPENT WAITING, OR PERFORMING INPUT/OUTPUT OPERATIONS

In this manner, a program keeps getting processor time until it finishes execution. This constant switching between various programs means the processor does not waste time waiting for I/O operations to complete, and, by using what could be wasted time, it can give each program the illusion that it has uninterrupted use of the processor.

Multiprocessing means that two or more processors in the same system are running under the control of a single operating system, the Master Control Program (MCP). In Burroughs systems, multiprocessing enhances multiprogramming, so several programs can execute simultaneously. This results in better utilization of memory and faster throughput on the system.


ELAPSED TIME
PROGRAM A
PROGRAM B
PROGRAM C
INDICATES ACTUAL USE OF PROCESSOR 1
INDICATES ACTUAL USE OF PROCESSOR 2
INDICATES TIME SPENT WAITING, OR PERFORMING INPUT/OUTPUT OPERATIONS

STACKS AND PROCESSES

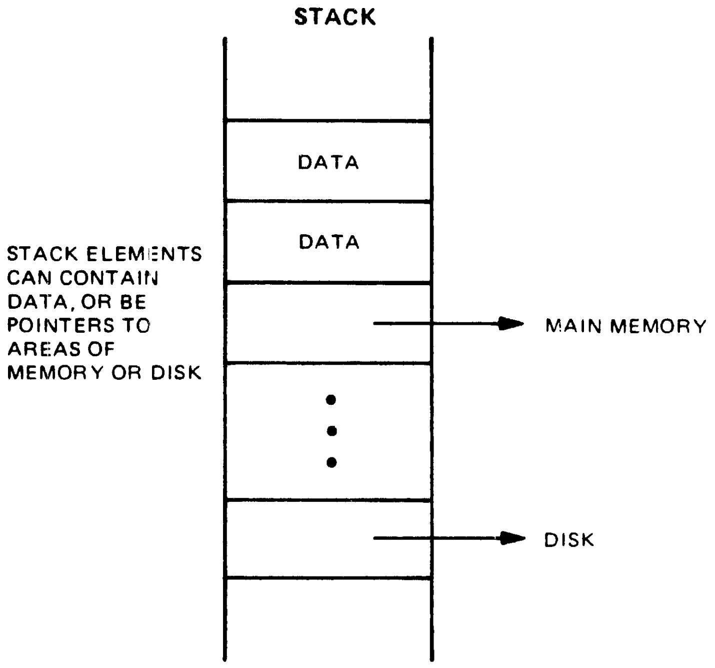
A single instance of running a program is known as a process. While a process runs, the system temporarily assigns to it a set of sequential locations in memory known as a stack. In the stack, the system stores the data, references to code and data, and the history for the process.

A system uses many stacks, but a processor is associated with only one stack at a time. If there are several processors in a system, at any moment each one is associated with a different stack. However, the stacks for many different processes can exist simultaneously. A processor can switch between stacks, performing part of a process each time it is associated with that process's stack. If a process initiates other processes, they can be linked so that they can share data.

The system originally determines the size of the stack needed for the execution of a program from calculations made by the compiler and stored with the program object code. If a program should need more stack space during execution, the system dynamically increases the stack size. Each time a program is executed, the system changes the size designation in the object code file to more closely reflect the stack size actually used, so that memory efficiency improves with each run.

Stack architecture allows easy implementation of the important features of A Series systems, such as virtual memory, dynamic allocation of resources, block-structured languages (Pascal, ALGOL), intertask communication and data sharing, and libraries. These features in turn make it easier for programmers to structure and write programs.

Implementation of these features is easy because A Series architecture allows the code and data for a program to be separated and placed anywhere in memory and on disk. The code and data are thus in separate segments but are linked to the stack by pointers called descriptors. There is a descriptor for each code and data segment, and they tell the system the location and size of the segment. If a required segment is on disk, the system brings it into memory.

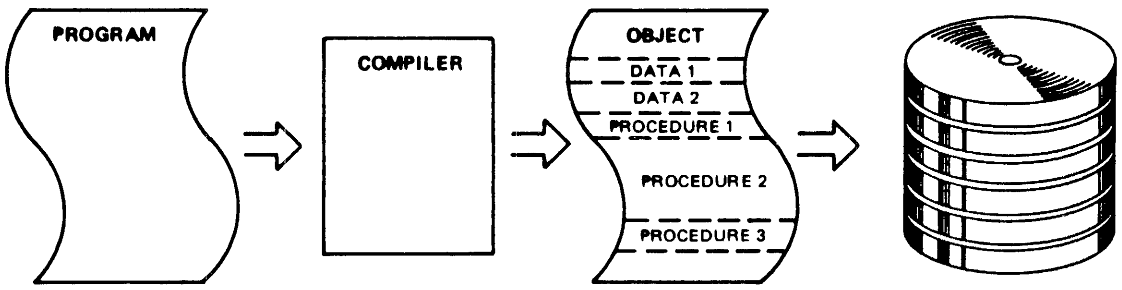

STACK
DATA
DATA
STACK ELEMENTS CAN CONTAIN DATA, OR BE POINTERS TO AREAS OF MEMORY OR DISK
MAIN MEMORY
DISK

This segmentation of code and data, linkage of segments to the stack, and swapping in and out of memory of segments means memory can be managed efficiently. Programs can be organized into logical tasks rather than constrained by fixed page sizes. These logical tasks can also be added to program libraries to be used by other programs.

VIRTUAL MEMORY

Virtual memory is a system technique that treats disk storage space as an extension of main memory, giving the appearance of a larger main memory than actually exists. With virtual memory, programmers can write programs that are independent of memory size restrictions--a program could require more physical memory than a system contains, yet run on that system.

To implement virtual memory, Burroughs compilers automatically divide programs into variable-length code and data segments. The size of each segment is determined by the logical structure of the individual program, rather than by using a "one size fits all" segment size.

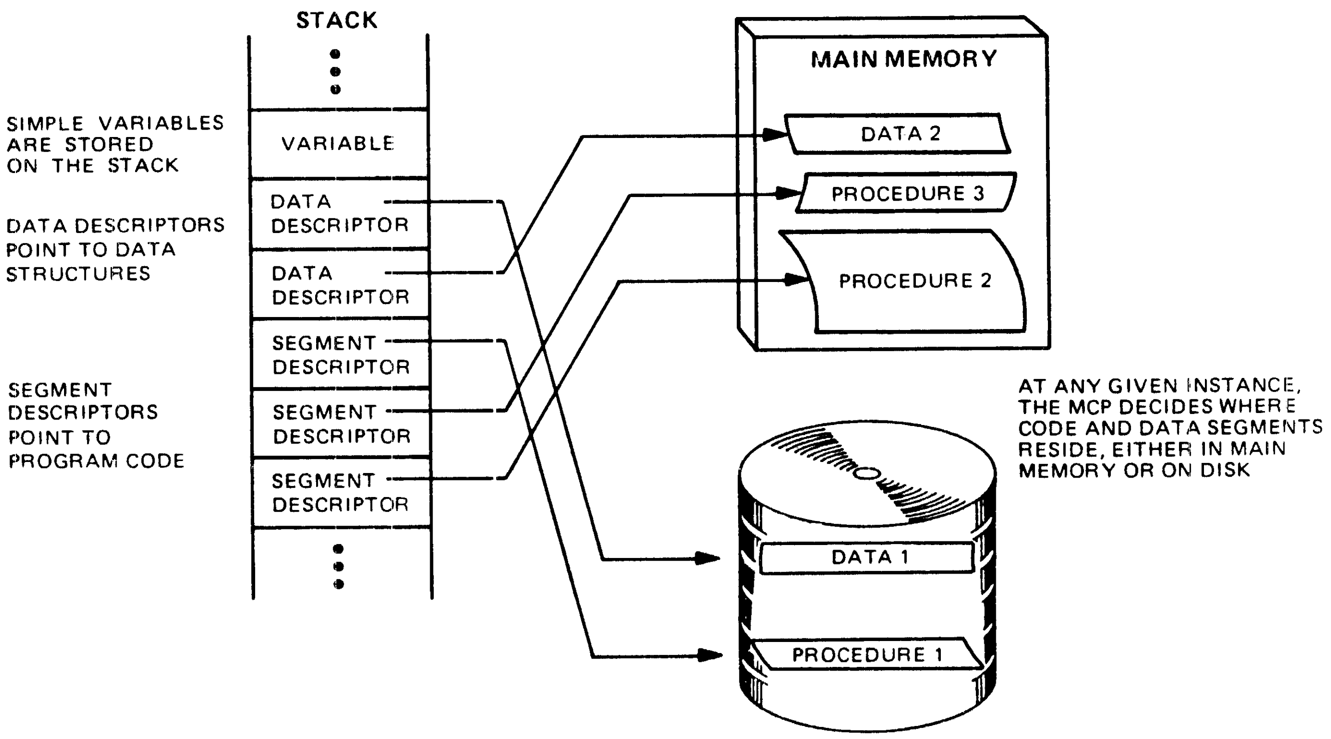


PROGRAM COMPILED INTO OBJECT CODE

OBJECT CODE DIVIDED INTO VARIABLE-LENGTH SEGMENTS

When the system executes a program, it brings in only those code and data segments that it needs at that moment for efficient execution. The other segments remain on disk until the program requires them. When the system finishes with a data segment, it writes that segment back to disk only if the data might have changed. Code segments do not need to be written back to disk because their contents cannot change. If a segment's memory space is then needed for some other use, the system frees that area of memory.

An executing program or task has descriptors in its working area (stack) pointing to each segment of code or data. Descriptors are words of a particular format that reference the size, status, and locations of segments in memory. The use of descriptors allows data and code to be located anywhere in memory. Space in memory for segments can then be allocated dynamically as needed.


STACK
VARIABLE
DATA DESCRIPTOR
DATA DESCRIPTOR
SEGMENT DESCRIPTOR
SEGMENT DESCRIPTOR
SEGMENT DESCRIPTOR
SIMPLE VARIABLES ARE STORED ON THE STACK
DATA DESCRIPTORS POINT TO DATA STRUCTURES
SEGMENT DESCRIPTORS POINT TO PROGRAM CODE
MAIN MEMORY
DATA 2
PROCEDURE 3
PROCEDURE 2
DATA 1
PROCEDURE 1
AT ANY GIVEN INSTANCE, THE MCP DECIDES WHERE CODE AND DATA SEGMENTS RESIDE, EITHER IN MAIN MEMORY OR ON DISK

The advantages of virtual memory include the following:

- o Less memory space per program execution
- o More tasks running concurrently
- o Increased system throughput
- o Simplified program creation

INTERACTIVE AND BATCH PROGRAMS

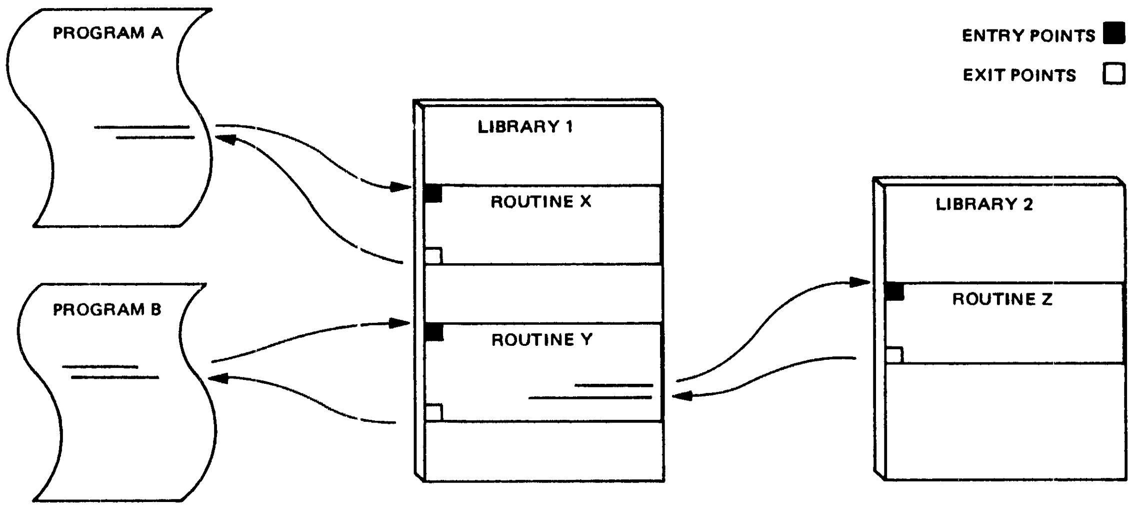
Programs can be roughly divided into two categories: interactive and batch. In an interactive program, the user gives commands or responses in a stepwise manner to queries from the system or program in order for a job to proceed. In a batch program, the system starts a job, and it proceeds without any user intervention.

The A Series Master Control Program (MCP) makes no discernible distinction between these two types of jobs. In fact, a single program can and may need to switch categories many times during its execution. One advantage of the Burroughs operating system is that it can run both batch and interactive jobs efficiently at the same time.

LIBRARIES

Through the Master Control Program (MCP), programs on A Series systems can access files of predefined routines (sequences of instructions) known as libraries. A library is generally a collection of named, related routines, such as data conversion routines or mathematical functions, that can be shared by many programs on a system. Library routines are designed to be called by programs to perform common functions and thus simplify program creation and maintenance. For example, a library might consist of routines that format program output for use on checks. Each routine would produce output in a different format, customized for a special type of check.

Libraries differ from regular programs in that the start of each routine forms an entry point into the library. This means a library can be entered at several points, whereas a regular program is always entered at one point, the beginning. Library routines can call other library routines. In fact, a chain of library linkages can be circular; thus, a library can reference itself, either directly or indirectly, through a chain of library references.



Some typical examples of the routines that form libraries are the following:

- FORTRAN formatting functions
- Data calculation routines
- Terminal formatting procedures
- Database transaction handlers

Burroughs supplies a set of libraries known as system libraries that perform many MCP-related functions considered intrinsic to the system, such as mathematical and formatting functions. System libraries are global; that is, they are available to all users in the system. Users can develop libraries, which can be either global to the system, local to the user who implements them, or specific to a particular user-created program. Library access can be controlled through the normal file access restrictions provided by the system.

For information on implementing libraries, see the "Libraries" section of the "System Software Utilities Reference Manual."

CONFIGURATION, RECONFIGURATION, AND SOFT CONFIGURATION

A configuration is the processor, memory, I/O devices, peripherals, and other hardware resources that together form a system. To configure a system means to allocate these resources. Because information about the way these resources are allocated is needed by the system, the information is stored in a special configuration file accessed during initialization.

For A Series systems with the MCP/AS ASD extended memory, memory is managed as a single monolithic area. Under a monolithic memory architecture, the operating system works with a single memory area available to all processors and programs in the system. ASD does not require memory partitioning.

For A Series systems controlled by the MCP, configuration information optionally consists of memory partitioning and I/O subsystem (peripheral) sharing of information. Two or more cooperating A Series systems can share an I/O subsystem and/or peripherals to reduce the total number of peripherals required to satisfy the peak needs of the individual systems. For systems connected to shared peripherals, a configuration file should be used to avoid possible contention between systems for the peripherals. However, for a standard, single-processor system without shared peripherals, use of a configuration file by the site is unnecessary.

Soft configuration refers to the ability to allocate and later reallocate hardware resources to the configuration through software. Through soft configuration, you can initially configure and later reconfigure resources through creation and modification of a configuration file and through associated operator commands. Configuring your system is a three-step process that involves building a configuration file, running the SYSTEM/CONFIGURATOR utility, and issuing an Operator Display Terminal (ODT) command or using the SYSTEM/LOADER utility to configure the system.

The section "A User's View of System Functions" contains a more detailed explanation of the soft configuration process.

A SHORT HISTORY OF BURROUGHS SYSTEMS

The current Burroughs A Series systems architecture is the result of a refinement of the architecture of the revolutionary B 5500 systems introduced in 1961. Prior to that time, most commercially produced computers were based on the stored program computer conceived by John von Neumann in 1946. The B 5500 retained the stored program computer concept but incorporated a number of radically different innovations.

The B 5500 was the first commercial computer system specifically designed to support high level, block-structured languages. The process of designing a system with this goal in mind led to innovations in the hardware and software. Among these were the following:

- Stack architecture, for fast and easy execution of block-structured languages
- Reentrant (nonmodifiable) code, to allow several programs to use the same code simultaneously
- Descriptors, to allow references to data to exist independently from the actual data
- Disk files, to allow random access to code and data on a disk
- Disk directories, to allow access to specific files on a disk
- A separate data communications processor, to ease the processing load on the main processor
- Virtual memory, to make programs independent of memory size
- Modular hardware architecture, to allow hardware modules to be added to or deleted from the system without any program changes

Some of these ideas were natural consequences of the others; for example, the use of array descriptors in a stack indirectly led to the first commercial use of virtual memory. An array descriptor describes the location of an array in memory outside the stack. If an array could be located outside of the program stack, but pointed to by a descriptor in the stack, then other parts of a program could be located elsewhere too. This led to segmentation of programs, and with segmented programs, only the part of the program actually executing needed to be in memory. The other parts could be kept in storage until needed. This reasoning, in turn, led to the treatment of disk storage space as an extension of main memory, hence virtual memory.

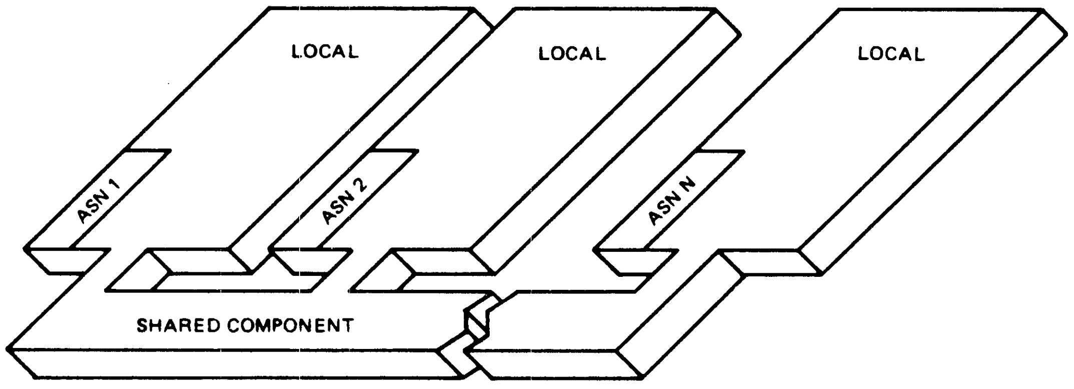
This explanation is simplified for brevity, but illustrates the continuing goal of Burroughs systems architecture, to make execution of high-level language programs as fast and as efficient as possible.

Since the B 5500, Burroughs has continually refined and built on that architecture to increase the speed and efficiency of program execution. The B 6700 used the same general architecture as the B 5500, but improved the implementation of it. On the B 6700, Burroughs implemented the following:

- o Programmable data communications processors, to allow the use of otherwise impossible or impractical protocols
- o Full employment of a block-structured addressing environment, which allows address space to be spread over several stacks
- o Tagged memory, to distinguish code from data and descriptors and to provide overwrite protection
- o Nesting of code blocks, with variables visible to all lower levels of the blocks
- o Support of multiprocess job families, including the introduction of the "cactus stack," which is a main process stack with multiple associated stacks

The B 7000 series of computers introduced pipelined processing to large systems to further speed processing. Pipelined processing involves beginning an instruction sequence before a prior sequence has finished.

The B 7900 provides separate address space for data and code. To be able to address each memory position and to make effective use of the B 7900's large memory capacity, Burroughs devised Address Space Number (ASN) memory. Because the system contains more physical memory than the addressing mechanism can address, memory can be divided into address spaces of up to one million words each, the maximum that can be addressed. Each address space is the memory that a task can access for execution and comprises a shared and a local component. The shared component is common to all address spaces and exists so that tasks can share code and data. The local component is unique to the address space and is identified by an Address Space Number (ASN). The sizes of both the shared and the local components are dynamically controlled by the system administrator through system software.



Burroughs newest systems, the A Series systems, provide users with the option to run under a new operating system, the Master Control Program/Advanced Systems (MCP/AS). MCP/AS, which provides all the features of the MCP, initiates a new generation of system software that complements the A Series architecture.

MCP/AS features a significant new form of memory management, the Actual Segment Descriptor (ASD) extended memory system.

The introduction of the ASD extended memory system, which is capable of directly addressing up to 4 billion words (24 billion bytes) of main memory (actual memory is machine-dependent), has made it feasible to directly address more memory than ever before possible on any Burroughs system.

ASD represents an upward evolution of the memory architecture of the Burroughs mainframes. In spite of this major change in the architecture of the Burroughs machines, the ASD system maintains object code compatibility with all previous large systems.

The ASD memory management system supports a single large memory structure that requires no partitioning. Because memory is not partitioned, programs are fully visible to one another while executing. In addition, the MCP can address very large amounts of memory. All allocated memory areas are accessed and controlled through a central structure called the Actual Segment Descriptor (ASD) table. The ASD table points to all locations in memory, virtual or physical.

The A Series systems incorporate the latest hardware technology to reduce the physical size of the machines while increasing processing power. These systems have the full functionality of previous large systems, plus menu-driven user/system communication. For instance, the A 3 has all of the power of the B 5900 in a small package compatible with office environments. The A 9 introduced the Multiple Logical Processor (MLP), which multiprocesses and multiprograms hardware instructions (operators) for greater execution speed.

3 A USER'S VIEW OF SYSTEM FUNCTIONS

This section discusses A Series systems as seen by you, the user. It cuts across traditional hardware and software boundaries to give you an overview of the system by broad functional areas. The purpose of this section is to give you an understanding of each area and of the parts of the system that affect that area. If you want more detailed information about an area, you can find it in the related documentation listed at the conclusion of each discussion.

Menu-Assisted Resource Control (MARC)

Menu-Assisted Resource Control (MARC) is a menu- and command-driven interface to Burroughs A Series systems. It was developed for systems operators, customers, and programmers of the system, and runs under the Communications Management System (COMS) message control system.

MARC supports the following functions:

- Burroughs Network Architecture (BNA) operator interface
- Operator Display Terminal (ODT) commands (those subsequent to system initialization)
- COMS control commands
- All Work Flow Language (WFL) commands
- Some system utilities
- Print System (PrintS) and Remote Print System (ReprintS)
- Mirrored Disk
- On-line help text
- On-site modification of screens
- Multiple versions of screens and help text
- Different natural-language versions

You can use MARC in three modes: menu, command, and tasking. The different modes are available to allow MARC to serve all levels of users, novice to expert, consistently and simultaneously.

In menu mode, MARC presents you with menus containing tasks and functions from which you choose the desired actions. You can go directly to the menu you want by entering the menu name, or you can reach the menu by moving through the menu structure. If the system needs more information for the desired action, it prompts you for the specific items. This mode eliminates the need to know an operations language in order to operate the system.

In command mode, you can bypass the menus and enter commands directly. This mode saves time for those familiar with the commands.

In tasking mode, you can initiate and track tasks through MARC. Tasking mode is entered automatically when menu selections or WFL commands require it.

MARC can be used in several different natural languages simultaneously, allowing each user to work in a separate language. The menus and screens in the menugraph, including messages and help information, may be translated into different languages by your vendor before you receive the software. If not, there are special utilities designed for that purpose: the Message Translation Utility, the Interactive Menugraph Generator (IMG), the Editor, and the Help Utility. The "MultiLingual System (MLS) User's Guide" explains how to translate items and when to use each utility.

Programmers and operators can modify the MARC menus and forms to suit the needs of their systems and of individual users and programs. Menu selections can be added or deleted, and new menus can be created, provided they use available commands. MARC also supports the creation of new commands via the "Directive" facility. In addition, a site can have several different sets of menus, called menugraphs, on a system and switch between them.

MARC provides on-line help information for each selection on the Burroughs-supplied menus. Users can access the help information by pressing a single function key. Through the Help Utility, programmers can modify existing help text and can write additional help text for any new menus and menu selections added. The "Help Utility User's Guide" explains how to do this.

Security supporting MARC incorporates all of the system security and COMS security features, including the log-on procedure. See the "Security" topic in this section for a description of system security.

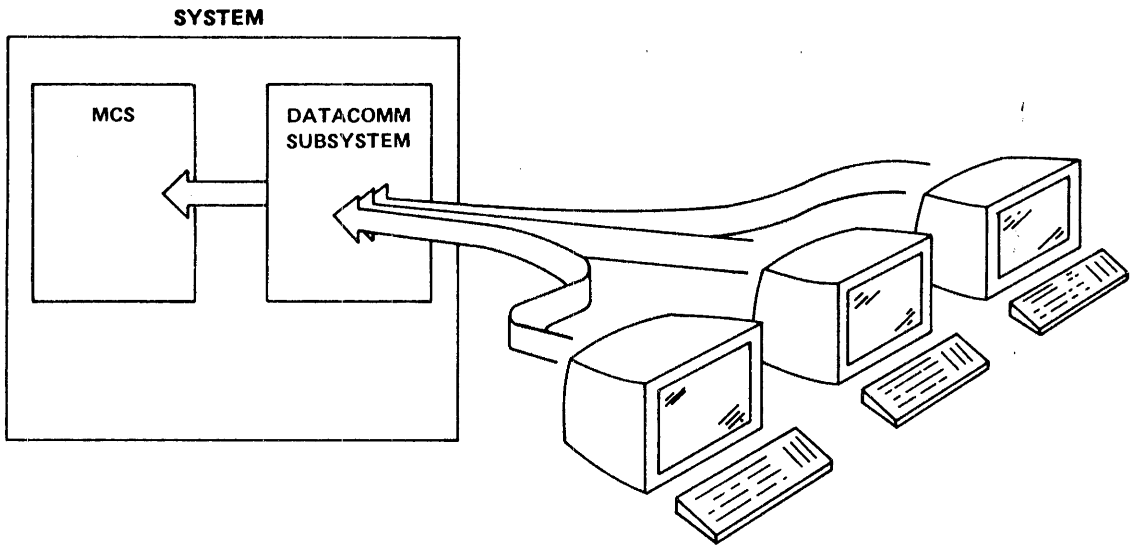
MESSAGE CONTROL SYSTEMS

A Message Control System (MCS) controls the flow of messages between terminals, application programs, and the operating system. In this context, a message is a transmitted series of words or symbols intended to convey information to or from the system, and thus includes most user communications with a computer system. In this communications environment, an MCS performs these additional major functions:

- o Logical control of stations to provide security and secondary error handling
- o Program initiation and control
- o Dynamic reconfiguration of the network, to add, remove, and reassign stations and to swap communications lines

An MCS can be general, so that it can be used for a number of different functions and by all system users, or it can be specific and be designed to serve specialized users and functions.

Refer to the "Menu-Assisted Resource Control (MARC) User's Guide" for complete information on MARC.



Communications Management System (COMS)

The Communications Management System (COMS) is a general MCS that supports a network of users and provides them with a consistent, on-line interface to the system. COMS is partially integrated into the MCP, which enhances its ability to efficiently handle a high volume of transactions from multiple programs, stations, and remote files.

COMS performs the following functions:

- o Accepts input from a station and returns output to that station or to a designated destination
- o Checks the security level of messages and treats them according to an installation-defined security scheme
- o Routes messages by transaction codes to, from, and between programs
- o Processes messages, if required, before sending them to or from a program
- o Initiates transaction programs
- o Performs synchronized recovery with Data Management System II (DMSII) and the Transaction Processing System (TPS)
- o Manages and handles errors for the data communications network

COMS is modular and designed to be customized to suit the needs of individual systems. From the COMS package, you select only the modules needed for your system. If desired, you can add user-written modules to perform site-specific functions. This modularity keeps software overhead to a minimum while providing the capabilities required by your system.

During a COMS session, each authorized user can have multiple "windows" into the system. Each window acts as a virtual terminal and provides access to a separate program or MCS. You can use the windows concurrently, and switch from window to window as desired. Through windows, COMS supports access to other MCSs such as CANDE, GEMCOS, and user-defined MCSs.

COMS defines the following windows at system initialization:

- MARC window. This window, which is always in operation in COMS, provides access to the Menu-Assisted Resource Control (MARC) program. One of the functions of the MARC program is to handle network and session control, and provide requested information and system messages for COMS.

- Utility window. This window provides access to the COMS utility, a utility used to define and change specifications in the tables of the COMS configuration file on-line.

- CANDE window. This window provides access to the Command And Edit (CANDE) program, if you have CANDE on your system.

- GEMCOS window. This window provides access to the Generalized Message Control System (GEMCOS), if you have GEMCOS on your system.

COMS provides two types of security:

1. Access security, which comprises the following:

   - COMS network access, to control who can log on to COMS

   - COMS station access, to control who can log on to individual stations

   - COMS window access, to control which windows each user can access

2. Process security, which controls the types of transactions allowed for each user or program

During installation, the system administrator defines a COMS security scheme for the system. Up to 32 security categories can be assigned to programs, stations, usercodes, windows, and transaction codes. COMS automatically checks security at each stage of message handling, and rejects any message that does not have the proper security level. Users can augment COMS security by adding security checking routines to application programs and pre- and postprocessing programs.

Synchronized recovery is a COMS function that completes any in-process transactions to a database after an abnormal termination of a program interacting with the database or after loss of the system. It ensures that all transactions that were in progress at the time of stoppage are completed. COMS resubmits the transactions in the same order in which they were originally submitted by the program or programs.

To define the operation of the COMS system or change that definition on-line, you use the COMS Utility. This utility operates in interactive (menu) or batch (command) mode and updates the COMS configuration file. The configuration file stores the information needed to operate and maintain the system and includes routing, processing, security checking, program initiation, and recovery information.

COMS is available in three versions:

- o Kernel MCS, which provides remote file handling, but no access to the COMS Utility for tailoring the COMS environment
- o COMS (Entry), which provides the basic functions excluding only transaction-based routing and synchronized recovery
- o COMS, which provides the full features of the system

The following documentation explains COMS in detail. You should read the Capabilities Manual first.

- o Communications Management System (COMS) Capabilities Manual
- o Communications Management System (COMS) Planning and Installation Manual
- o Communications Management System (COMS) Operator's Guide
- o Communications Management System (COMS) Programmer's Guide
- o Communications Management System (COMS) Migration Guide

Command AND Edit (CANDE) Language

The Command AND Edit (CANDE) language is a time-sharing MCS designed to simplify communication during the creation and execution of programs. CANDE provides a single, complete environment from which you can edit files and manage programs. Through CANDE, users can interact with and control programs while the programs are executing.

CANDE allows you to develop and test programs in an on-line, real-time environment. It reduces programming overhead by supplying tools to perform functions that would otherwise require user-written programs.

A User's View of System Functions

Through CANDE, users can perform the following tasks:

- 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

You access these functions through CANDE commands. CANDE restricts access to commands concerning system operation and security by requiring that those commands come from stations (terminals) with "control" status or from Operator Display Terminals. A system administrator normally gives control status only to those stations that are used to monitor and regulate CANDE activity.

Access to the system via CANDE is restricted through the use of a usercode, password, and, optionally, an accesscode and password preassigned by the system administrator. You begin a session by entering that usercode, password, and accesscode. During the session, the system logs the tasking commands issued and optionally prints the log and all print jobs when the session ends. This log can be used for accounting and tracking purposes.

CANDE contains a set of commands for the editing of text records in files. Through these commands you can change, insert, delete, move, find, and replace text within your workfile. When you "get" or "make" a file during your CANDE session, CANDE creates a workfile that is a copy of the source file but entirely separate from it. You can edit this workfile, test the effect of the changes, and edit and test multiple times without changing the original source file. The source file is not changed until you save the workfile.

To protect against the loss of data if the system should go down, CANDE continually saves the changes made to a workfile during a session. These changes are stored in a special CANDE file on disk. Certain editing commands, such as move, insert, replace, and merge, cause the workfile to be updated immediately. Other editing commands cause CANDE to accumulate several records, then update the special file. By using the special file when the system resumes operation, CANDE can create a recovery file containing the workfile, including the changes made to it during the interrupted session.

The following documentation explains CANDE in greater detail.

- CANDE Operations Manual
- CANDE Reference Manual

PROGRAMMING LANGUAGES

Standard Languages

Because programming languages are the principal means of performing work on a computer, standards have been developed for them to increase the portability of programs from one system to another. Frequently, a company will add extensions to a standard language to expand the language's usefulness and to simplify using special capabilities of the hardware.

Most of the languages supplied by Burroughs conform to international standards, when they exist, or to American National Standards Institute (ANSI) standards. If there are extensions or exceptions to the standard in the Burroughs implementation of the language, they are usually noted as such in the reference manual.

The following paragraphs describe the standard high-level languages for A Series systems and list the associated manuals. Acronyms used in the descriptions are:

ANSI - American National Standards Institute

ISO - International Standards Organization

ALGOL - ALGOrithmic Language. The extended ALGOL Burroughs offers is designed for the communication of algorithms or procedures to Burroughs systems and is used for applications and systems programming. ALGOL is based on the "Revised Report on the Algorithmic Language ALGOL 60" (Communications of the ACM, Vol. 6, No. 1; January, 1963). The manual for this language is the "ALGOL Reference Manual."

BASIC - Beginners All-purpose Symbolic Instruction Code. BASIC is a general-purpose language based on the ANSI X3.60-1978 standard for BASIC. The manual for this language is the "BASIC Language Reference Manual."

COBOL74 - COmmon Business Oriented Language. COBOL is designed for the implementation of business-oriented data processing programs. COBOL74 conforms to ANSI standard X3.23-1974. The manual for this language is the "COBOL ANSI-74 Reference Manual."

COBOL(68) - COmmon Business Oriented Language. COBOL(68) conforms to the COBOL standard established by the Conference On Data Systems Languages (CODASYL). The manual for this language is the "COBOL Reference Manual."

DCALGOL - Data Communications ALGOL. DCALGOL is intended for the implementation and control of a data communications system and for accessing privileged system functions other than privileged data management functions. The manual for this language is the "DCALGOL Reference Manual."

FORTRAN(66) and FORTRAN77 - FORmula TRANslation. FORTRAN was designed for scientific applications. FORTRAN(66) is compatible with FORTRAN IV, H level, and contains ANSI X3.9-1966 Standard FORTRAN as a subset. FORTRAN77 conforms to ANSI Standard X3.9-1978. The manual for FORTRAN(66) is the "FORTRAN Reference Manual" and for FORTRAN77 is the "FORTRAN77 Reference Manual."

NDLII - Network Definition Language II. NDLII is for the definition and implementation of data communications networks through generation of a Network Control Program. This language was created by Burroughs. The manual for this language is the "Network Definition Language II (NDLII) Reference Manual."

NEWP - NEW Programming language. NEWP is a variation of ALGOL and is used to write the operating system and system utility programs. NEWP can also be used for applications programming. The manual for NEWP is the NEWP Document on the Mark 3.6.0 Documents tape.

Pascal - Pascal is a block-structured, general-purpose language. Pascal implementation is based on the ANSI/IEEE 770X3.97-1982 and to ISO 7185 Level 0. The manual for this language is the "Pascal Reference Manual."

PL/I - Programming Language One. PL/I is a general-purpose language that conforms to ANSI Standard X3.74-1981. The manual for this language is the "PL/I Reference Manual."

RPGII - Report Program Generator II. RPGII is a language for accessing and processing data and specifying the type and format of reports to generate using this data. The documents for this language are the following:

- Report Program Generator (RPG) Reference Manual
- Report Program Generator (RPG) Debugging Template

Sort - The Sort language was created by Burroughs for the writing of programs that sort and merge files. The manual for this language is the "Sort Reference Manual."

WFL - Work Flow Language. WFL is Burroughs' high-level, block-structured language used for constructing jobs that run tasks and control their execution. WFL includes variables, expressions and flow-of-control statements that offer the programmer a wide range of capabilities with regard to task control. The manual for this language is the "Work Flow Language (WFL) Reference Manual."

Combining Programs in Different Languages

On A Series systems, programs and subprograms written in different languages can be bound together to produce one executable program. This feature allows standard procedures, such as mathematical functions, to be written only once yet used by all applications on the system. The procedure is compiled separately and later bound to a compiled program, in the same or another language, that references that procedure. The program that accomplishes this is the Binder. The "Binder Reference Manual" explains the program and the acceptable language combinations.

JOB AND TASK MANAGEMENT

A task is a single, complete unit of work performed by the system, and a job is one or more related tasks that are grouped together. Tasks are usually programs or subroutines that a job uses to accomplish its purpose.

A program written in WFL is referred to as a job. During execution, a job normally executes as a task with its own code file. A WFL job can initiate other tasks, such as compilations and executions of user programs.

The execution and flow of jobs and tasks in a system can be handled and monitored by the Work Flow Management system. This system gives the system administrator overall control of the installation's work flow, particularly control of billing, execution priorities, and resource sharing.

The Work Flow Management system includes the following program modules:

1. The Master Control Program (MCP), which initiates the WFL compiler and then processes the code file generated by the compiler

2. The WFL compiler, which compiles the control language used for job control

3. The CONTROLLER, which manages the display routines and the scheduling queues for the jobs provided by the Work Flow Management system

You can create a job by writing a program in a high-level control language called WFL. WFL is not a substitute for regular application languages but is used in addition to them.

Controlling Jobs And Tasks

You can control jobs and tasks through WFL statements, through the assignment of task attributes, through the establishment of job queues and assignment of jobs to them, and through assignment to memory subsystems. If you do not need this level of control, jobs and tasks can be handled automatically by the operating system (MCP). All that you need to do is compile or execute a program through CANDE.

WFL statements are used to start jobs, to manage files, to control the flow of tasks within a job, and to pass parameters to tasks.

Task attributes are the mechanisms used to specify and examine the characteristics of a task. Included in the attributes are some optional ones that tell the MCP how system resources are to be allocated for this task. These resources include execution priority, I/O time, processor time, tapes allowed, and lines printed.

Job queues are waiting lines for jobs submitted for processing. They are a means of controlling the amount of each system resource a given job can use and the execution priority of the job. Job queues are defined when the system is first initialized. They are independent of one another but can be assigned relative priorities. All WFL jobs go through job queues.

An individual job queue can place limits on the amount of certain system resources a job from that queue can use. A job can specify resource amounts up to an absolute limit, or can omit the specifications and use the default limits of the queue. When the system administrator defines a job queue, he or she also gives it a MIXLIMIT, which is the number of jobs and tasks from that queue that can be executing at any given time. If the creator of the job does not assign it to a job queue, the MCP places it into the highest-numbered queue allowed by the job's attributes or into a site-designated default queue.

Subsystems are memory environments created when memory is physically partitioned by Global Memory or Address Space Number (ASN). The system automatically distributes tasks to memory subsystems, but you can override this distribution by specifying, through a task attribute or job queue, which subsystem to use.

Billing

Billing for jobs can be handled by the JOBFORMATTER program. JOBFORMATTER reads information from the job logs, formats that information, and prints it. If jobs and costs are to be tracked, Burroughs supports an optional billing support library, SYSTEM/BILLING/SUPPORT, to which JOBFORMATTER passes the logged information. This library contains dummy procedures that you customize to suit your needs. The library procedures can pass information back to JOBFORMATTER to print along with the job summary.

Every job and task is identified by a mix number, a unique number assigned by the system when the job is submitted for processing. All subsequent commands concerning that job, and all logged information about the job, use the mix number.

JOBFORMATTER creates a printer backup file that can be printed locally or at the remote location.

Tasking

Tasking refers to the process of running multiple tasks concurrently or consecutively and coordinating the execution of those tasks through communication between the task and the job that initiated it.

WFL jobs can initiate a task as a synchronous task, one that runs separately from the job and that must finish before the job can continue. For concurrent execution of tasks, tasks can be run asynchronously, meaning that the job does not wait for them to finish in order to continue. WFL contains commands to allow tasks within a job to run asynchronously and to cause a job to wait if necessary until a certain condition is met.

Jobs and tasks communicate through the passing of parameters and through the use of task variables. The task variable gives the job or WFL user access to the values of the task's attributes at any time before, during, or after the execution of that task. By reading these attributes, the job or user can make decisions concerning which of several possible paths the job should take.

Restarting Jobs

If jobs are interrupted by a system failure, the system, by default, automatically restarts jobs at the last point where no task was running. However, the WFL programmer can override the default and control how and where the jobs restart.

Some application languages permit stops in a program, called checkpoints, for the system to save information regarding the state of the program at that point. This information consists of everything the system needs to know to resume execution of the program.

If the program must be restarted, the user can restart it at a checkpoint instead of at the beginning. WFL provides a command that directs the system to restart the program at a particular checkpoint.

FILE MANAGEMENT

Effective management of files implies, first of all, that files be distinguishable from each other and, second, that they can be handled as groups as well as individually. Therefore, a key to effective management is a file-naming scheme that allows easy identification and grouping of files. Another key is ease of manipulation and tracking of files.

File Types and Names

Individual disk files are known to users and to the system by a multipart file name followed by a family name. (A family is a disk or group of disks with a common name.) The file name indicates which usercode a file is stored under and the name that identifies that file. The family name specifies the physical disk family on which that file is stored. When you access a file, you identify it as "file name ON family name".

The disk file name contains from 1 to 12 parts separated by slashes. Each part consists of from 1 to 17 alphanumeric characters, hyphens, and underscores, or consists of a quoted string up to 17 characters long. The first part of a file name can be either a usercode in parentheses or an asterisk (*), indicating that the file has no associated usercode. If the first part is a usercode or asterisk, it is not followed by a slash. The following examples show possible valid combinations of file and family names.

```
(PAYROLL)WEEKLY/CHECK/PRINTING ON ACCOUNTING

(BUDGET)MONTH12/CHANGES/BY/DEPARTMENT ON CONTROLLER

*PATIENT/WEEKLY/CHARGES2 ON HOSPITAL
```

A file name with the last part or parts replaced by an equal sign is referred to as a directory name. A directory is a list of files organized into a hierarchy according to similarities in their names. Files are grouped together in a directory if their first name constants and associated usercodes are identical up to a slash or a right parenthesis and if they are on the same family (the family is the last part of their names). Thus, if you have files A1 and A2, although the first parts, "A", are identical, "1" and "2" are not, so these files will not be grouped in the "A" directory. However, suppose you have the following files:

```
(USER1)MONTH/END/PRELIM
(USER1)MONTH/END/FINAL
```

```
(USER1)MONTH/ESTIMATE
(USER1)BUDGET/FINAL/VERSION
(USER1)BUDGET/MONTHLY
```

When the files are organized into a directory hierarchy, the names are listed in the directory in the following manner:

```
(USER1)
    MONTH
        END
            PRELIM
            FINAL
        ESTIMATE
    BUDGET
        FINAL
            VERSION
        MONTHLY
```

The first three files are in the directory "(USER1)MONTH", and the last two are in the directory "(USER1)BUDGET". The first two files are also in the directory "(USER1)MONTH/END", and can be thought of as forming one branch of "(USER1)MONTH", while the file "(USER1)MONTH/ESTIMATE" forms the other branch.

Directories allow you to apply certain commands or statements available in the system, such as "PD (USER1)MONTH/=..." or "COPY(USER1)=...", to all the files in a directory.

You can specify a file name without a usercode or family name or both. The system automatically adds your usercode and assigned family name when it searches for the file or when you create the file. If it cannot find the file under your usercode, it searches for it among the files without a usercode, those starting with an asterisk (*).

Tape files have two-level file names. The first name is the volume name for the reel or reels of tape in the tape set, and the last name is the unique file identifier. Each part of the name can contain from 1 to 17 alphanumeric characters, hyphens, and underscores.

Printer and punch files contain program or system output and are written to disk or tape for later transfer to the designated peripheral. You can let the system name and manage these files automatically, or you can name the files yourself and store them under your own usercode, using the rules for disk or tape file naming.

A remote file is a means of passing information between a program and a remote device, such as a terminal, and is used by most interactive programs. The program writes data to and receives data from the remote file, treating it as it would treat a local peripheral. The file is attached to the remote device by the Message Control System, and can be attached to multiple stations simultaneously for multiuser programs. Remote files are named following the conventions for disk file names.

Port files permit direct communication between programs, including programs running on different systems connected through Burroughs Network Architecture (BNA). Port files are named using the conventions for disk files.

File Structure

A Series systems support many different kinds of file structures, including variable- and fixed-length records and blocked and unblocked files. Blocked files are those in which a number of physically adjacent records are transferred to or from the file as a group in order to reduce transfer time. Access to the records in files can be sequential (and also indexed) or random, using record numbers or keys and reading the files forward or backward. The user has control over the size of records, blocks, and files, and describes these and other characteristics of a file to the system through file attributes. See "Input/Output Subsystem" later in this section for more information on file attributes.

Disk File Maintenance

Disk file maintenance, sometimes referred to as "library maintenance," consists of those functions that add or delete files and modify their names or security. These functions are performed on either system or user files.

You perform file maintenance through the following Work Flow Language (WFL) statements:

o CHANGE	This statement changes the names of files on disk.
o COPY-ADD	COPY copies disk files to and from disk and tape. Disk files copied to tape are bit-for-bit images of the disk and thus are different from files created expressly for tape. ADD copies a disk file to a destination if a file with the same name does not already exist on the destination.

- REMOVE — This statement removes files from a disk.
- SECURITY — This statement changes the access security of a file on disk.

You can access these same functions interactively through CANDE commands with the same names. See the "CANDE Reference Manual" for information on the use of the commands.

Execution of these statements is restricted by file access security; thus, if you are a nonprivileged user, you cannot perform these functions on a particular disk file unless the security attributes of that file allow you the use of that function.

The file maintenance commands are described in detail in the "Work Flow Language (WFL) Reference Manual."

The DUMPALL File Maintenance Utility

DUMPALL is a generalized file maintenance utility used primarily for transfer of files from one medium to another. DUMPALL also generates listings of files and controls the movement of files from tapes.

Using this utility, you can copy any kind of file from any medium to any other medium. DUMPALL accepts standard or nonstandard tape labels or unlabeled tapes. Tape files created by DUMPALL from disk files are bit-for-bit disk images.

Input to DUMPALL consists of one or more utility commands that allow you to do the following operations:

- o List the attributes of a file or files
- o Concatenate files or parts of files
- o Copy one or multiple input files to one or multiple output files on the same or different devices
- o Display a file or a tape in hexadecimal, EBCDIC, and other formats
- o Print a file in one of several formats
- o Test a file for I/O errors

The "System Software Utilities Reference Manual" describes the operation of the DUMPALL utility.

Cataloging

Protecting information stored on disk files is a major concern of any computer installation. To achieve adequate protection, an installation will make copies of disk files on magnetic tape for storage at the site and in secure locations away from the site. However, keeping track of the copies and different versions of the files may get complicated. To track the copies, you can run Burroughs A Series systems with an optional feature called cataloging.

Cataloging has two main purposes:

1. To provide the means for finding a file or archived copy of a file
2. To track the generations of a file

The different copies of a file are referred to as generations. Cataloging keeps track of the different generations and of the copies of each generation. You are allowed to have from four to seven different generations and up to two copies of each generation. When you access a cataloged file, the system uses the catalog to locate the file. It gives you the resident copy of the file unless the USECATALOG file attribute is TRUE, in which case it gives you the latest generation unless you request otherwise.

Using cataloging to automate your handling of duplicated files involves a tradeoff. File search is slower, because extra disk I/O operations must occur for every opening or closing of a file. Disk space is also required to hold information about nonresident copies.

For more information on Cataloging, see the "Disk Subsystem Software Overview" manual.

<u>**Files on Tape**</u>

Magnetic tape is used as a medium in its own right and as a way to archive or transport disk file images.

A Series I/O subsystems accept either labeled or unlabeled magnetic tapes. Tape labels are special records at the start, end, and between the files of a reel of tape that provide details about the files stored on the tape. The I/O subsystem recognizes and reads all standard Burroughs tape labels, including those from medium and small systems. The subsystem also reads the following labels:

- o All ANSI Level 2 standard labels
- o IBM 360-format labels

In addition, the subsystem can handle unlabeled tapes. These require operator intervention to assign files to a tape on a specific unit.

The I/O subsystem writes ANSI standard labels when creating a labeled tape. Only the first and last parts of a file's name are used in the name of a file in the tape label, to conform to an ANSI requirement for two parts in the file name. Thus, if a file name was A/B/C/D, the name stored in the tape label would be A/D. ANSI also requires that the first part of the name be the same for all files on the tape; in other words, the files must be in the same directory.

The I/O subsystem supports multiple files on a single tape and one file split across multiple tapes. Also supported are multifile, multireel tapes, with files split across reels as necessary.

DATA MANAGEMENT

Data Management System II (DMSII)

A database management system is a specialized system software package used to describe a database, maintain the relationships between the various data elements, and access the information in the database. Burroughs Data Management System II (DMSII) supplies all of these functions and more to allow users to obtain information where and when it is needed. In addition, DMSII supports multiple databases and allows them to be accessed simultaneously from on-line time sharing, batch, and remote-job-entry environments.

DMSII centralizes management of information. Data file structures and interrelationships are described only once to the system; users and programs can then add, update, and retrieve data without concern for data placement, organization, or access methods. The database administrator uses the Advanced Data Dictionary System (ADDS) to create the physical database by describing the database structures and elements to the system. From these structures and elements, the database administrator creates logical databases that control which of the structures can be accessed and how they can be used.

DMSII databases can be described, initialized, accessed, loaded, and reported against in a simplified way through DataAid. DataAid, described later in this section, works in conjunction with ADDS and Extended Retriever with Graphic Output (ERGO).

Users can inquire about, add, modify, and delete data items within a DMSII database by using the ERGO program, described later in this section. ERGO has a menu-assisted interface to accommodate both inexperienced and experienced users.

Recovery of a database must occur anytime processing is halted abnormally. Recovery means reconstructing the database and ensuring that no partially-completed transactions are left in it. This involves undoing the most recent changes and notifying programs where to restart processing. Within DMSII, recovery is comprehensive.

For audited databases, recovery occurs automatically the next time the database is accessed. (An audited database is one for which a record of all changes and events is kept.) You can also initiate recovery manually by running the RECOVERY program. For unaudited databases, you must reload the database files and control file from a backup dump and reprocess the input.

Along with the basic facilities just described, DMSII is totally integrated with a number of utility modules for accessing information, modifying information, determining database integrity, and analyzing database status and structure. The following paragraphs briefly describe these utilities and tell where to find more detailed information.

DataAid

DataAid is an interactive, menu-driven system that simplifies the overall process of describing, initializing, loading, and accessing a DMSII database. DataAid is complemented by on-line documentation.

If you have to describe a new database, DataAid transfers you to ADDS. When the definition stage is complete, DataAid monitors the generation of the database software modules and the database initialization. In the following stage, DataAid automatically transfers you to ERGO where you can load the database through the ERGO update feature, and then develop reports from the database. DataAid also initiates and tracks the dump, copy, and recovery functions of the database, if requested.

DataAid simplifies the changes users most commonly make to databases. It does not attempt to present a complete interface to the operational capabilities of DMSII.

For details, refer to the "DMSII DataAid User's Guide."

Extended Retrieval with Graphic Output (ERGO)

Extended Retrieval with Graphic Output (ERGO) is a program used to retrieve data from a DMSII database and produce reports from that data, to display information graphically, and to relate and format data. At the option of the site management, ERGO can also be used to modify, add, or delete records from a database.

ERGO has an extensive capability for data manipulation through built-in mathematical and statistical functions. Its report capability includes tabular and statistical formats, data extraction to both flat files and B 20 workstations, plus a variety of graphic formats: discrete and continuous plot graphs, bar graphs, kiviat diagrams, and histograms. You can use standard report formats included in ERGO or define your own formats through control of over 50 variables.

ERGO is designed for both inexperienced and experienced users. A menu-assisted interface prompts the user through the processing of a request. In addition, help information is available for any command or data item. If desired, experienced users can bypass the menus by entering direct commands.

ERGO allows you to join data sets and access up to five DMSII databases and 10 conventional files in a single session. Also available are search and sort mechanisms, parameters for data selection, and extensive mathematical capabilities. Finished reports can be directed to a disk file, a printer, a B 20 workstation running Data Transfer System (DTS), or one or more terminals. For more information on ERGO, see the following manuals:

- o Extended Retrieval with Graphic Output (ERGO) Capabilities Manual
- o Extended Retrieval with Graphic Output (ERGO) User's Guide

Advanced Data Dictionary System (ADDS)

The InterPro (tm) Advanced Data Dictionary System (ADDS) is a software tool that provides centralized storage and retrieval of data definitions that are used in application programs and DMSII databases. These data definitions describe most elements of information on a system, including DMSII database definitions, Screen Design Facility (SDF) screen formats, COBOL74 file structures, and other collections of records. ADDS data definitions can be included in application programs and can be used to automatically generate DMSII databases.

ADDS allows you to define processes, which can be procedures, job streams, manual preparation procedures, or any user-defined entity. A process is a structure that describes or models a logical view of the relationships that exist between different parts of a system. Processes make it possible to document the flow of a total system and inquire against it for reporting.

ADDS makes it possible to eliminate inconsistencies and redundancies in data and to control additions, changes, and deletions. ADDS is a multiuser system that manages all concurrent retrieval and modification, thus ensuring data definition integrity.

You communicate with ADDS through menus, through which you define, modify, and retrieve data, plus obtain reports on the use and structure of the data within the dictionary. There are two levels of security to prevent unauthorized access to data: system security, for controlling system access, and ADDS security, for controlling access to individual data items. For more information about ADDS, refer to the following manuals:

- o Advanced Data Dictionary System (ADDS) Capabilities Manual
- o Advanced Data Dictionary System (ADDS) Installation and Operations Manual
- o Advanced Data Dictionary System (ADDS) User's Guide

DM Interpreter

DM Interpreter provides direct access to DMSII databases through application languages that do not have DMSII extensions. This interface eliminates the need both for special DMSII statements in programs and for prior knowledge of database operations. The primary requirement for using DM Interpreter is that the application language must support libraries.

DM Interpreter is a dynamic run-time interface that allows an application program to be compiled before the databases it uses are created. This is possible because an application program never directly accesses the database but instead calls the DM Interpreter library, which is tailored to particular databases when the library is compiled. Parameters that describe the types of operations to be performed on the database are specified when the application program is run. Thus changes to the database do not normally require changes to the application programs.

The "DMSII Interpretive Interface User's Manual" contains more information about DM Interpreter.

Database Certification (DBCERTIFICATION)

The utility program Database Certification (DBCERTIFICATION) determines the integrity of a DMSII database. It provides three levels of certification:

1. Physical integrity. The utility ensures that a file is physically intact and accessible, and isolates any problems at the data block level.

2. Internal, intrafile integrity. The utility verifies that data in a single structure is consistent, and that storage control and internal control information are valid.

3. Interstructure integrity. The utility verifies that relationships between data structures are correct (crosschecking).

This utility can be used interactively or in batch mode. You can select which structures in the database to verify and which tests to perform at each of the three levels.

For more information about this utility, see the "DMSII Data Base Certification Software Operation Guide."

The DB MONITOR Utility

DB MONITOR is an interactive program that provides database status and statistics information and permits changes to database options and parameters. The status and statistics information can be for the entire database or for selected structures of the database, and includes the following:

- The number of users
- Whether users are updating information or making inquiries
- The number of transactions
- Overlay rates
- The number of reads and writes performed

The "DMSII Utilities and Operations Guide" contains more information about DB MONITOR.

<u>The DBANALYZER Utility</u>

DBANALYZER is a software tool that analyzes the logical and physical structure of a database and provides reports that a database manager can use to modify, tune, reorganize, and document the database. DBANALYZER describes the structures in the database and their interrelationships, analyzes the database file, and provides information about how the files are referenced. The program does not, however, allow a user to examine the contents of the database, so it does not compromise database security.

You can use DBANALYZER interactively or in batch mode. Simple commands control the scope and depth of each analysis and allow you to direct the output to a terminal or a printer.

See the "DMSII Utilities and Operations Guide" for more information about DBANALYZER.

<u>The INQUIRY Utility</u>

DMSII INQUIRY is an interactive utility program for examining or modifying data in a DMSII database. INQUIRY was designed to allow users relatively unfamiliar with database concepts to easily and effectively access and use database information. Through this utility you can:

- Examine the contents or the description of a database
- Modify, create, or delete records within a database
- Generate reports from the database
- Extract information from the database and place it in a standard file

You generate an INQUIRY program by running the BUILDINQ program and answering a short series of prompts in which you specify which database you want to access and which operations you want to perform. From your answers, the utility creates a program that you then run. Both program generation and execution can be performed in either interactive or batch mode.

For more information about INQUIRY, see the "DMSII Inquiry Software Operation Guide."

DATA COMMUNICATIONS

The Data Communications Subsystem is the communications link between all remote devices, such as other hosts and terminals, and the central system. The Data Communications Subsystem serves as the transport mechanism for other communications networks. The following description assumes you have at least an elementary knowledge of data communications theory.

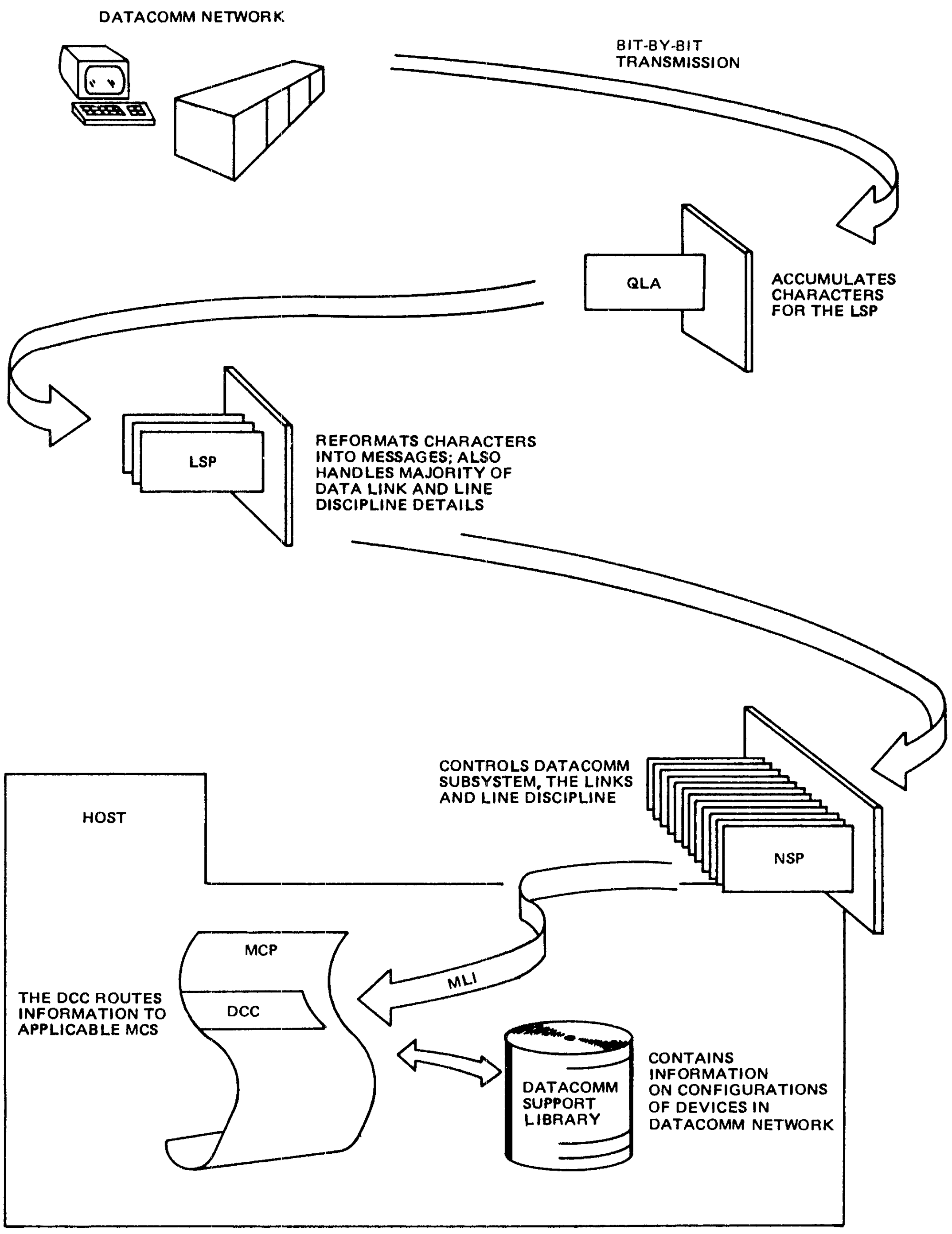
Data Communications Hardware

The Data Communications Subsystem is microprocessor-based and modular in design. It off-loads communications functions from the central processor and distributes them among specialized processors--the Network Support Processor (NSP), the Line Support Processor (LSP), and the Data Communications Data Link Processor (DC-DLP). The NSP and LSP are programmable and capable of implementing many different communications protocols. The DC-DLP is preprogrammed with selected protocols.

The low-level interface to the datacomm lines is a bit- or character-oriented device known as a Line Adaptor (LA). Each LSP can be connected to from one to four Quad Line Adaptors (QLAs). (A QLA is a set of four line adaptors on one circuit board.) The DC-DLP contains four single line adaptors.

The following illustration shows the flow of messages through the datacomm subsystem.

A User's View of System Functions


DATACOMM NETWORK
BIT-BY-BIT TRANSMISSION
QLA
ACCUMULATES CHARACTERS FOR THE LSP
LSP
REFORMATS CHARACTERS INTO MESSAGES; ALSO HANDLES MAJORITY OF DATA LINK AND LINE DISCIPLINE DETAILS
CONTROLS DATACOMM SUBSYSTEM, THE LINKS AND LINE DISCIPLINE
NSP
HOST
MLI
MCP
DCC
THE DCC ROUTES INFORMATION TO APPLICABLE MCS
DATACOMM SUPPORT LIBRARY
CONTAINS INFORMATION ON CONFIGURATIONS OF DEVICES IN DATACOMM NETWORK

Messages are entered at the remote device, where the firmware translates them for bit-by-bit transmission over the datacomm line. If the device is located at a distance from the host, a modem plus telephone line is used to connect to the Line Adaptor (LA). Otherwise, a direct connection is used.

The message flows through the Quad Line Adaptor (QLA), which is the electrical interface between the communications line and the Line Support Processor (LSP). The QLA accumulates characters for transfer to the LSP.

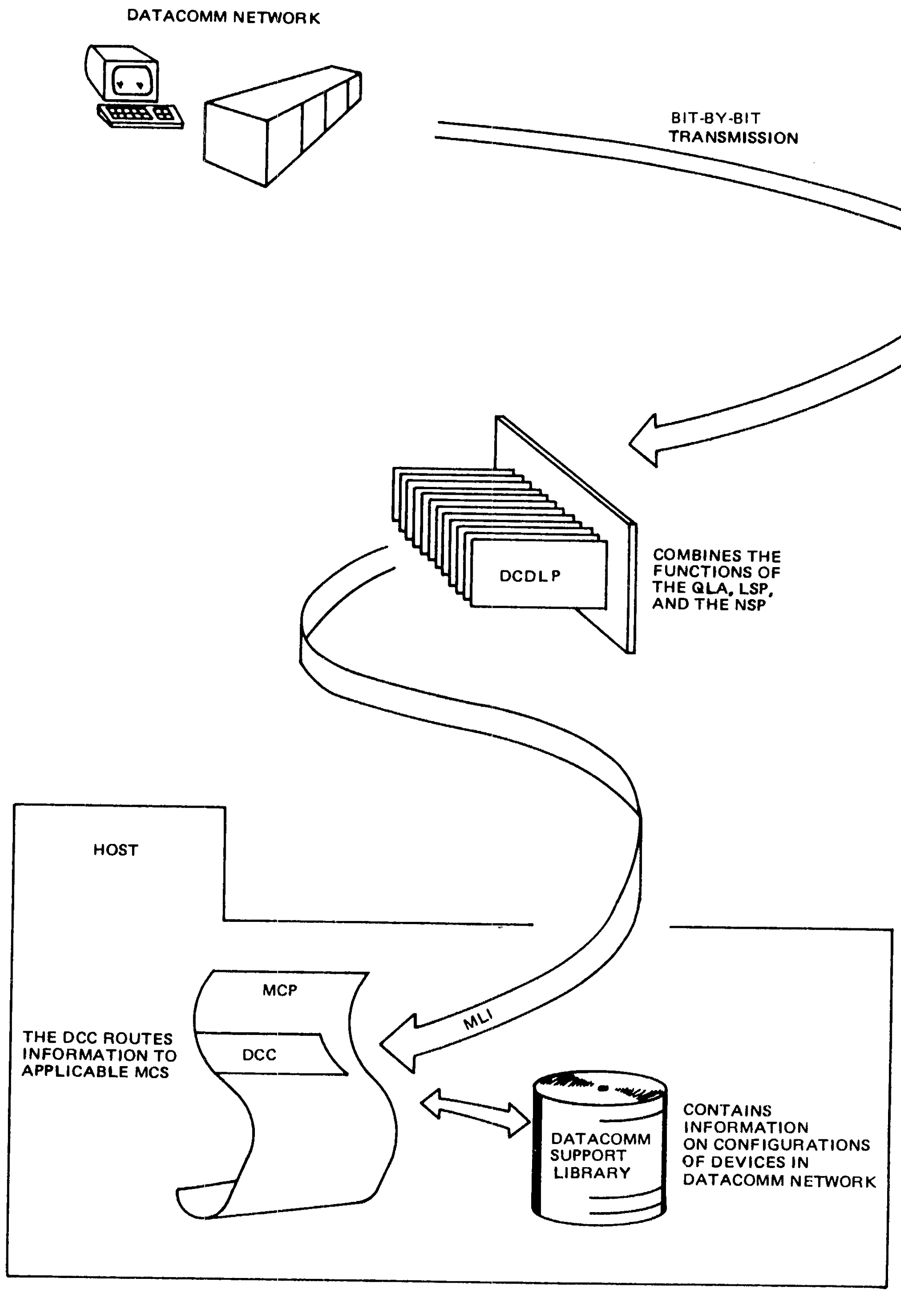
The LSP is the connection between the QLA and the Network Support Processor (NSP), and handles the majority of data link and line discipline control details for the NSP. The LSP reformats the characters it receives into messages, because from this point on data is sent as messages rather than as bits or characters. This message-level transfer reduces the number of interrupts and allows more information to be transferred at a time.

The NSP controls the datacomm subsystem, the links, and line discipline. By performing these functions, it allows the central processor to use processing cycles for message processing rather than datacomm "housekeeping." The host loads the user-specified line control algorithms (routines) into the NSP and, through the NSP, loads the line adaptor control routines into the LSP. The line control routines provide services to user programs such as handling line errors and error recovery. The NSP is a data link processor and is treated as a peripheral by the system.

The DC-DLP combines the functions of the NSP and LSP. It has dedicated protocols such as the Burroughs poll-select and teletype protocols. It has the advantages of speed and lower cost, but lacks the flexibility and programmability of the LSP/NSP combination. Functionally, the system treats a DC-DLP as an NSP and identifies both as NSPs: NSPs 1 through 4 are NSPs with LSPs attached, and NSP 5 is a DC-DLP.

The messages between the NSP and the central processor flow through the Message-Level Interface (MLI) of the I/O Base Module to the mainframe I/O subsystem, as shown in the following figure. From this point on, software takes over.

A User's View of System Functions


DATACOMM NETWORK
BIT-BY-BIT TRANSMISSION
DCDLP
COMBINES THE FUNCTIONS OF THE QLA, LSP, AND THE NSP
HOST
MCP
DCC
THE DCC ROUTES INFORMATION TO APPLICABLE MCS
MLI
DATACOMM SUPPORT LIBRARY
CONTAINS INFORMATION ON CONFIGURATIONS OF DEVICES IN DATACOMM NETWORK

Data Communications Software

Interactive Datacomm Configurator (IDC)

Data communications networks are defined and maintained through the Interactive Datacomm Configurator (IDC), a menu-driven utility. This utility modifies the datacomm information file to reflect the current datacomm network. IDC requires minimal knowledge of data communications, yet allows the definition of complex networks containing user-defined protocols and terminals.

IDC accepts standard and user-developed protocols. Burroughs supports industry-standard protocols and supplies them plus example definitions for lines and stations in the SYSTEM/DATACOMINFO file supplied with the software. The definitions can serve as models for new lines and stations to be added to a configuration.

IDC allows you to perform certain updates to the datacomm configuration currently running on the system. This ability means that you can make changes, such as adding and moving lines and stations, or test new algorithms and editors without having to interrupt the functioning of your datacomm network.

The DATACOMINFO file contains a complete description of the configuration, including algorithms, editors, and translate tables. IDC modifies the configuration by causing the DATACOMMINFO file to be updated.

An optional capability of IDC, the DataComm Processor (DCP) to Network Support Processor (NSP) configuration conversion aid, is an interactive, menu-driven utility that allows you to transform the configuration section of a DCP datacomm network to the configuration section of an NSP datacomm network. The configuration conversion is designed for the A Series systems, the B 5900, B 6900, B 7900 systems, and for TD-compatible terminals. Refer to the "DataComm Processor (DCP) to Network Processor (NSP) Configuration Conversion Aid" manual for more information.

To allow for flexibility, the datacomm subsystem accepts specialized, user-developed protocols. Users develop protocols by writing and compiling NDLII programs (see below) and then adding the resultant object program to the network configuration through IDC. For more information on IDC, see the "Interactive Datacomm Configurator (IDC) User's Guide."

Network Definition Language II (NDLII)

Network Definition Language II (NDLII) is a high-level language for developing user-written line protocols. An NDLII program contains a network description and the tables and code for NSPs and LSPs. You would use NDLII when you want to use a nonstandard protocol on your datacomm network. Once the protocol is defined, you can incorporate it into the network using IDC. You can also use NDLII to generate a full network description and then later use IDC to modify that description. The "Network Definition Language (NDLII) Reference Manual" explains NDLII.

How the Data Communications Software Works

Messages come from the I/O subsystem to the Data Communications Controller (DCC). The DCC is a procedure of the MCP, and one copy of it is initiated for each NSP in the network. The DCC monitors NSP activity, decodes incoming messages and routes them to the appropriate MCS or application program, and formats outgoing messages.

Each DCC copy has information about the NSP it controls and about the devices attached to that NSP. It obtains this information at initialization time through calls to the Data Communications Support Library. The library contains routines to access the DATACOMINFO file, which stores all information about the configuration of the devices in the datacomm network.

The DCC routes any information from the datacomm system, such as status information, to the applicable MCS. The MCS that receives the message is a special purpose program, usually written in DCALGOL. DCALGOL is an extension of ALGOL that contains statements for creating messages, moving them in and out of queues, and creating, combining, or changing queues. It is a powerful language that also contains features for the development of internal system software. See the "DCALGOL Reference Manual" for information about the language and about writing an MCS.

Data Communications Utilities

Four utility programs aid in monitoring and analyzing a datacomm network. The programs and their functions are as follows.

o DCAUDITOR	This utility prints monitored NSP traffic. It can print all traffic or selected classes, such as messages, errors, or trace messages.
o NDLIIANALYZER	This utility aids in the writing and debugging of NDLII algorithms and editors. It combines information from the Network Information File and an NSP dumpfile to produce information concerning the status of an NDLII process. The "Network Definition Language II (NDLII) Reference Manual" describes the operation of this utility.
o NSPDUMPANALYZER	This utility produces formatted listings of NSP memory dumps.
o DCSTATUS	This utility tells you the current status of the datacomm network by producing an analysis of the state of the datacomm tables maintained by the MCP and the datacomm subsystem. The "System Software Support Reference Manual" describes this utility.

BURROUGHS NETWORK ARCHITECTURE (BNA)

Burroughs Network Architecture (BNA) allows the connection of multiple, independent Burroughs computer systems into a network. The purpose of this network is to give the user in a distributed processing environment the same type of access to remote resources he or she has to local resources.

BNA gives the user, including programs, the ability to

- o Access files and databases on other hosts
- o Transfer files between systems
- o Share resources on other systems
- o Execute and control jobs on other systems
- o Communicate with users and programs on other systems
- o Transfer datacomm stations to other systems

Although a BNA network can be composed of different types of Burroughs systems, all of the networks have the following characteristics:

- o All systems in the network are cooperating peers regardless of size.
- o Control is distributed evenly among all systems.
- o The routing algorithm responds automatically to changes in the network configuration and characteristics.
- o The user interfaces are a simple extension of normal operations.

How It Works

This explanation assumes you have at least an elementary knowledge of data communications.

In a BNA network, each system, called a host, supports its own users and one or more connections to other hosts. Each connection point is known as a node and is identified by a unique node address. Nodes within the same site can be connected through direct hardware links. More distant nodes are connected by dedicated (leased) or switched point-to-point telephone lines or by public data networks that support the X.25 interface.

Nodes directly connected to each other are known as neighbor nodes. A node communicates with its neighbor through a station, which is the equipment and programs that send, receive, and control the messages and message flow. BNA supports four types of stations:

1. Burroughs Data Link Control (BDLC), for switched and dedicated lines

2. X.25, the CCITT standard interface to a Public Data Network

3. Inter-System Control (ISC), for direct hardware connections

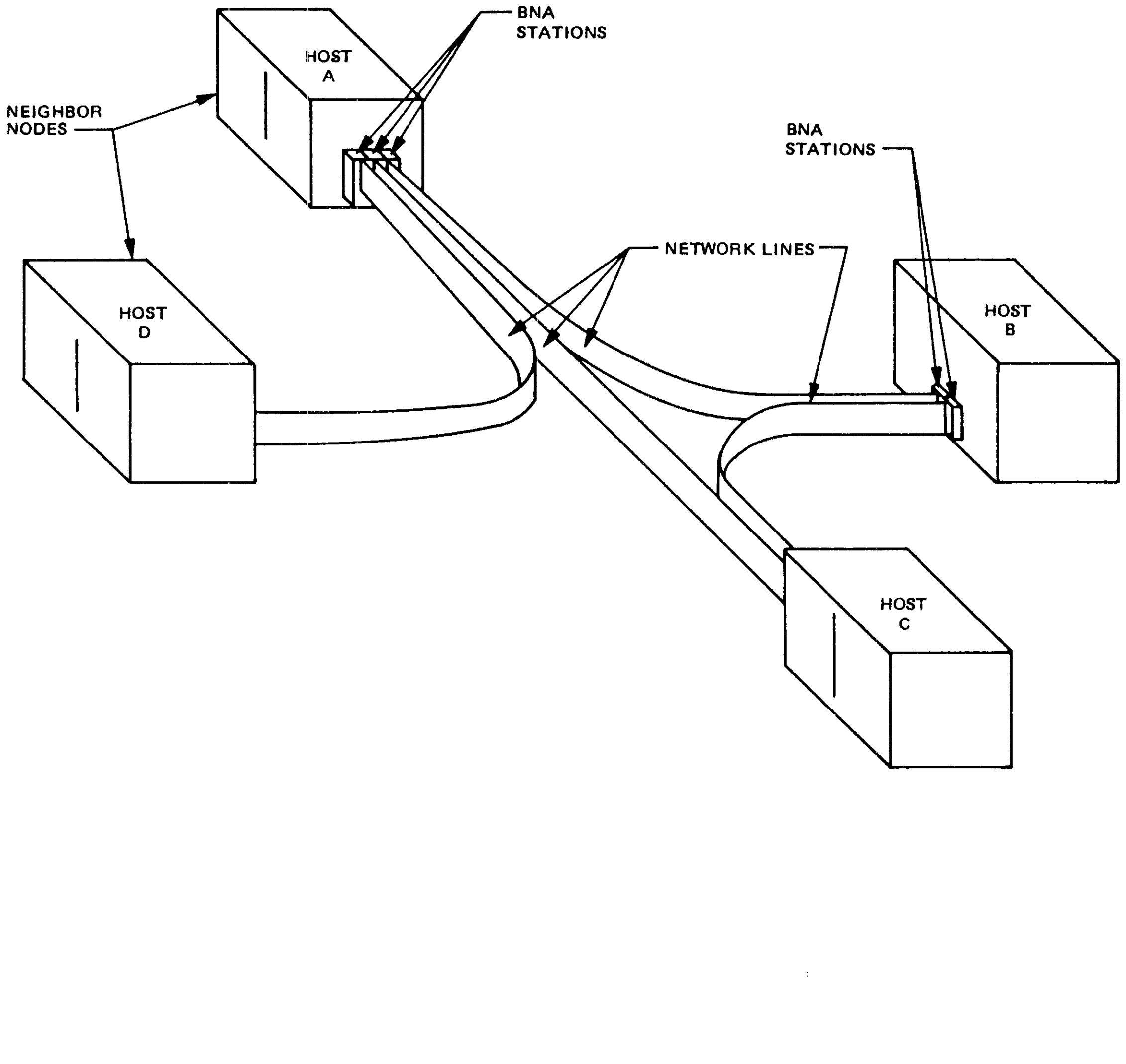
4. Global Memory, for pre-A Series systems with *GLOBAL tm Memory

Non-neighbor nodes can communicate because their messages are routed from neighbor to neighbor through the network. Routing tables at each node keep track of the most efficient route to any other node. These tables change dynamically as nodes or links are added to or deleted from the network, or when the physical characteristics of any node or link change.

* "GLOBAL" is a trademark of Burroughs Corporation.

For control purposes, each host is equal and has equal responsibility for running the network. As each node becomes active in the network, it establishes physical connections with neighboring nodes, then logical connections to all nodes, through a series of greeting messages and routing information exchanges.

A host has complete control over which of its resources are available to network users, and can selectively allow or deny resource use to individual hosts, programs, or users. There are two types of access control: node and host, and individual user. Node and host access control is achieved through validation and authentication. The validation process controls which hosts and nodes are allowed access, and authentication verifies that a host or node is what it claims to be.



Individual user access control is through use of a hostname/usercode pair for log-on to a remote host. Once the hostname/usercode pair is verified, the remote host drops the hostname portion and treats the usercode as if it were a local usercode. This treatment requires coordination of usercodes across the network to avoid duplication on any host. However, a preassigned local alias for the usercode can be used after verification to remove the necessity for this coordination.

BNA provides the option of logging specific events for such purposes as billing, scheduling, reporting, performance monitoring, testing, and maintenance. This logging is in addition to any other system logging. The types and frequency of logging are specified by the user.

BNA is functionally divided into two major levels: Host Services and Network Services. Host Services is the "upper" level and provides the services associated with a host, such as access to remote hosts and file transfer. Host Services can be viewed as an extension of the operating system.

Network Services is the "lower" level and provides the physical and logical connections between nodes for transporting data and messages. Network Services can be thought of as the transporting mechanism for Host Services.

Using BNA

Existing application programs usually require no modification to run on BNA. Such programs can access remote resources through the use of file equation statements at run time. In addition, there are BNA extensions to high-level languages to allow interprogram communication between different systems. BNA task and file attributes, control statements, and Operator Display Terminal (ODT) commands are used to control the network and to request services at remote hosts in a manner similar to local service requests.

The manuals that describe BNA are the following:

- o Burroughs Network Architecture (BNA) User's Guide
- o Burroughs Network Architecture (BNA) Program Agent User's Guide

OPERATIONAL INTERFACE

On a mainframe computer system, an operator must be able to

- o Display information concerning the system
- o Initialize the running environment, including data communications, networks, and so on
- o Start utility programs
- o Execute controlled dumps
- o Perform many other essential system functions

On A Series systems, these functions are easy and efficient to perform. Because of careful ergonomic design, the operating system supports operation of the system by people with varying levels of expertise. For an expert in certain types of operations, a fast, efficient command interface, the Operator Display Terminal (ODT), is provided. For those less expert in any area, a menu-driven interface called Menu-Assisted Resource Control (MARC) is provided. An operator can use either interface interchangeably and concurrently.

Within MARC there are two ways to communicate: through the menus and forms and through direct entry. MARC menus and forms guide the user through command entry, making it unnecessary to learn system commands. Direct entry requires some knowledge of operational commands, and is accomplished through ODT mode on an ODT or through MARC command mode on either an ODT or an end-user terminal.

Direct Entry of System Commands

You can enter system commands directly through an ODT or through MARC on either an ODT or an end-user terminal. A few system commands are privileged and can be entered only through an ODT. However, all commands entered through MARC are subjected to security checking.

In normal ODT mode, an ODT is a privileged device, meaning that commands entered through it have privileged status. In addition, anyone using an ODT can enter commands without logging on to the system and without entering them under a usercode.

Other features of ODT mode include direct entry of Work Flow Language (WFL) commands and immediate execution of certain of those commands. The "Operator Display Terminal (ODT) Reference Manual" describes the syntax and use of system commands.

<u>SECURITY</u>

Security for a computer system comes through controlling the access to that system--to the system as a whole and to its component parts. Access control can be physical, as in computer room, tape library, and terminal access, or can be through software. Control of physical access is determined for an individual site by that site's management, preferably during the initial planning stage. Control through software is optional. Burroughs supplies the means of software control, but each system administrator can choose whether to implement all of it, selected parts, or none.

<u>Software</u> <u>Access</u> <u>Control</u>

For A Series systems, there are three types of software access control: user, system, and file.

User Access

User access is controlled through usercodes, passwords, and access codes. A unique usercode can be used to identify each user who is allowed access to the system. Because the usercode is at times displayed on various output media, it is public. However, a password, known only to the individual person, can be used to validate the identity of the user and prevent unauthorized use of the usercode.

The usercode has two purposes:

1. To identify the user for system access and for logging and accounting purposes

2. To identify the user's files for security control and for normal owner access

A usercode can have up to 15 passwords associated with it. This allows several users to have the same usercode yet use different passwords to gain access to the system. This is helpful when a group of people, such as an accounting department, need to access the same secured files. A password common to all can quickly become common knowledge outside the department through inadvertent slips. Also, when one person leaves the department and his password must be changed for security reasons, the other department members need not learn a new password.

A User's View of System Functions

In order to perform certain necessary system functions, such as archiving files on a daily basis, someone must be able to access files that are normally secured and access privileged system functions. To allow this access, the system administrator can designate a few selected users as being "privileged." A privileged user can access any files and run any programs on the system, regardless of the file security designation. For accountability purposes, it is strongly recommended that the smallest possible number of persons be given privileged status.

The Menu-Assisted Resource Control (MARC) interface has two levels of privileged status, Systemuser and Privilegeduser. A Systemuser has operator privileges and capabilities and has access to functions that affect system operation. A Privilegeduser has access to privileged functions and secured files. Having Systemuser status does not make a usercode privileged, and vice versa, but a usercode can be designated as both Privilegeduser and Systemuser.

Programs also can be marked as privileged, meaning that the programs have the same status as a privileged user and can access all files and privileged system functions. The PP (Privileged Program) ODT command is used to give a program privileged status. Nonprivileged users can run these programs if they are specifically given permission to do so through file security.

Accesscodes provide an additional level of user security. They are similar to usercodes, have one associated password, and are used as a second log-on procedure after the usercode/password log-on. Accesscodes are also used for additional file security.

Usercodes, passwords, and accesscodes are created through the MAKEUSER utility, which builds a minidatabase, the USERDATAFILE. This file contains all the needed information about system users. Only privileged users can access the utility to create or change the file. The MU (Make User) ODT command is an alternative way of adding or changing usercodes; its use is optional and can be restricted. Information about the MAKEUSER utility is in the "System Software Utilities Reference Manual" and about the MU command is in the "Operator Display Terminal (ODT) Reference Manual."

System Access

System access is controlled through log-on procedures. Logging on involves entering a usercode and, optionally, a password in order to use the system. This may be done interactively or, in batch mode, through a job statement. Whether or not the log-on procedure is required is up to the individual site, and within that site's system, to the message control programs in use. For example, CANDE requires all users to log on, while RJE gives the system operator the option of omitting the log-on procedure.

An important exception to the log-on procedure involves Operator Display Terminals (ODTs). These terminals accept input without requiring a user to log on to the system. For this reason, physical access to an ODT may need to be restricted.

File Access

File access is controlled by file attributes, through which the owner of the file defines the level of security each file is to have. The owner can specify who can access the file and the type of access each user can have. The file security attributes are SECURITYTYPE, SECURITYUSE, SECURITYGUARD, and SENSITIVEDATA.

SECURITYTYPE specifies who, other than the file owner, can access the file. This attribute can have the following values:

o PRIVATE	Only the owner and privileged users can access the file.
o PUBLIC	Any user can access the file.
o GUARDED	Only those users and programs listed in the guard file can access the file. The owner and privileged users are excluded from this restriction. The guard file is explained below.
o CONTROLLED	Only those users listed in the guard file, including the owner if he or she is nonprivileged, can access the file. Users can be identified by accesscode in addition to usercode or program name. Privileged users are exempt from this restriction.

SECURITYUSE defines how a file protected by security can be used. This attribute has the possible values of SECURED, IN, OUT, or IO.

o SECURED	For data files, no access is allowed. Secured code files can be executed only. The file cannot be read or written to.
o IN	The file can only be read (data files) or executed (code files).
o OUT	The file can only be written to.
o IO	Code files can be read and executed. Data files can be read and written to.

SECURITYGUARD names the guard file to be used if the file's security type is GUARDED or CONTROLLED. The guard file contains the names of the users and programs that can access the file and their respective access rights. Users are identified by usercodes and optionally, by accesscodes; programs are identified by file name and family name. Both users and programs can be further restricted to access only when running certain programs or when running under certain usercodes, respectively.

The MCP checks the guard file whenever a nonprivileged user or program attempts to open the file. To create a guard file, use the GUARDFILE utility, described in the "System Software Utilities Reference Manual."

SENSITIVEDATA provides access security after a file is removed from a disk pack. When the value of this attribute is TRUE, the MCP overwrites the removed file's disk area with an arbitrary pattern before the disk area is returned to the system for reallocation. This prevents subsequent users of that area from reading any of the file's data that might have been left in that space. The SENSITIVEDATA attribute should be reserved for high security files because it can slow down I/O operations.

Network Security

Burroughs Network Architecture (BNA) has a distributed approach to security. In a BNA network, each host (Burroughs computer system) operates independently of the other hosts and controls its own resources. A host governs remote user access by maintaining a list of allowable remote usercode/hostname pairs in its USERDATAFILE. Each request for service from a remote user is checked against this list and subjected to the same degree of security checking as local requests.

In addition to user, system, and file access control, the host also controls access to such resources as processor time, peripheral use, file storage space, and access to system functions. Changes to access privileges can be expanded or revoked at any time without notifying other hosts in the network.

Compiler Security

Three of the specialized languages offered by Burroughs--DCALGOL, DMALGOL, and NEWP--contain language extensions that allow a program to access internal system functions and data files. These extensions, essential because of the specialized uses of the languages, mean the compilers for these languages might be a problem for system security.

A user does not need privileged status to use these compilers, unless the compilers have been secured at that site. Programs generated by the compilers and not given privileged status will run as long as they do not try to access privileged system functions or files. However, if they are given privileged status, the programs can access some privileged MCP functions.

The NEWP compiler checks the "safety" to the system of program statements as it generates object code. If it finds unsafe statements, it compiles the program but marks it as unsafe. This in turn causes the system to refuse to execute the program, unless the program is one installed by a system operator, such as a new MCP, system library, or a standalone program such as the LOADER. Unsafe statements are those that, if incorrectly used, might cause a program to access memory areas outside the program's normal addressing environment or that would cause the system to halt. The MCP and certain system libraries are the only routinely executed programs marked unsafe by NEWP.

To prevent any security problems from arising, you may want to limit access to the DMALGOL compiler. This can be accomplished in several ways:

1. By limiting the number of privileged users on the system to a very small, controlled group and by physically removing the compiler from the system, returning it only long enough to perform scheduled compilations, and then removing it again

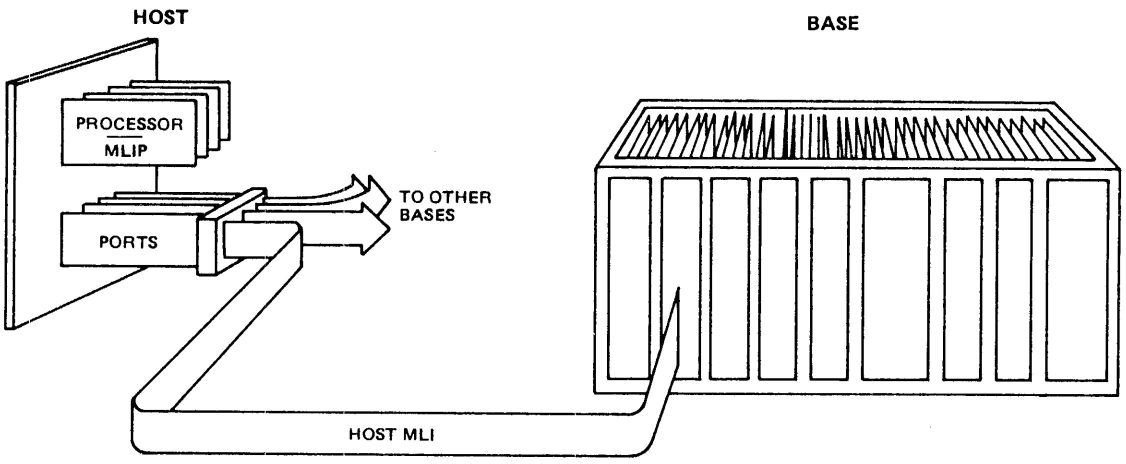
2. By designating the compiler files as GUARDED or CONTROLLED and using a guard file to list the persons who can access them

INPUT/OUTPUT SUBSYSTEM

The input/output (I/O) subsystem manages all transfers of information between system and application software and peripheral devices. The flow of information through the subsystem can be viewed on two levels: the physical and the logical. The physical level handles the movement of data and control information to and from the peripheral devices and includes all physical devices involved in that movement. The logical level structures the data for transmission and manipulates the structure so that data is delivered in the desired quantities. The following pages discuss I/O first from a physical, then a logical viewpoint.

Physical I/O

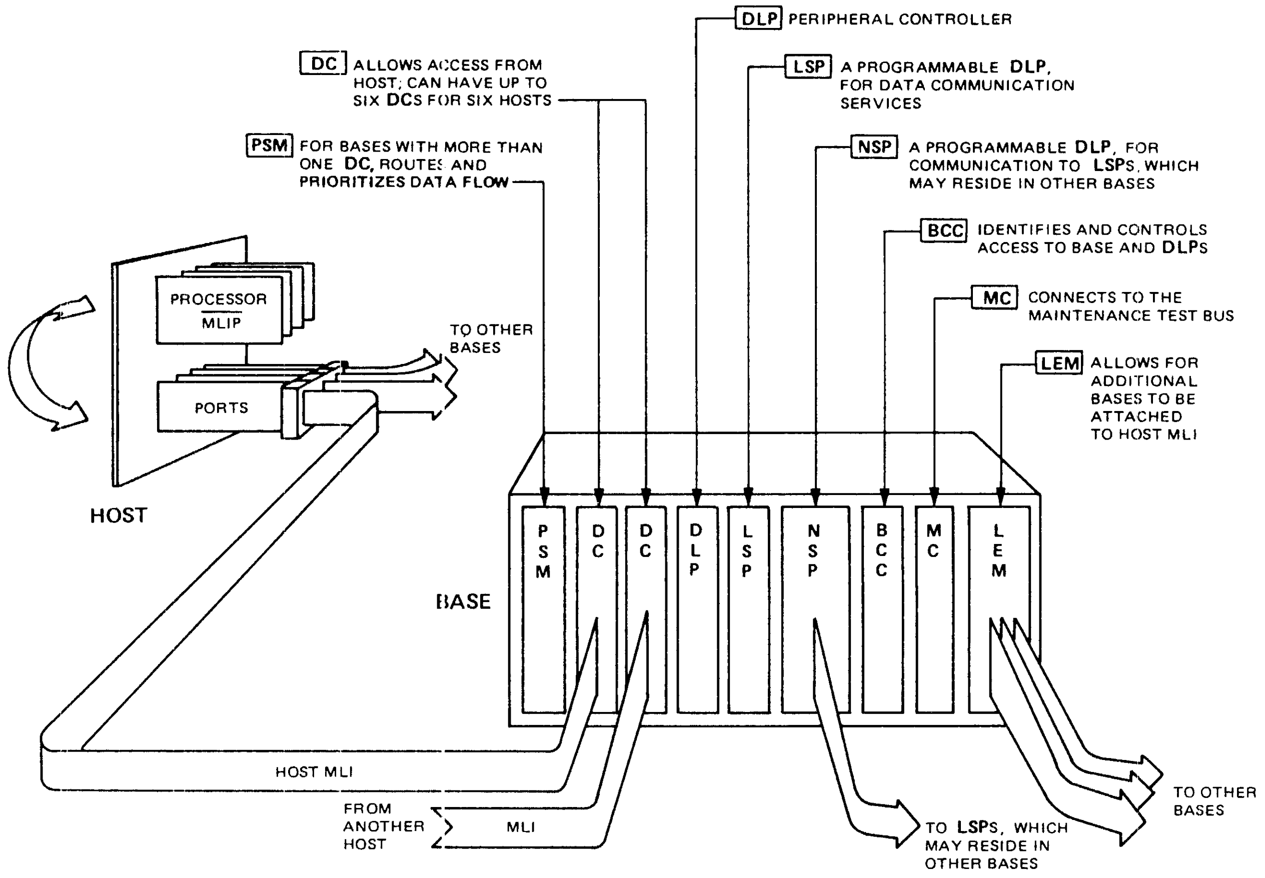
On Burroughs A Series systems, each central processor has an associated Message-Level Interface Processor (MLIP). The MLIP handles I/O operations for the MCP, accepting I/O requests from it and returning the resulting information to it. The MLIP communicates with the I/O subsystem through a Message-Level Interface (MLI), which is connected to an I/O base, as shown in the following diagram. The connection between the MLIP and MLI is made through one or more MLIP ports.



Each I/O base contains one or more Data Link Processors (DLPs), which function as peripheral controllers. PT-DLP handles both the printer and the tape drive; disks have different DLPs, depending on the type of disk used. Network Support Processors (NSPs) and Line Support Processors (LSPs) are programmable DLPs that provide data communications services. A DLP can be connected to a peripheral directly, to a peripheral controller, or, if an NSP, to another base.

A base is connected to an MLI through a Distribution Card (DC). There can be as many as six of these cards, allowing access from six different hosts. A host is any module that has an MLI port, such as an A 9 processor or an NSP. Additional bases can be attached to a host MLI port through use of a Line Expansion Module (LEM).

Bases with more than one Distribution Card must have a Path Selection Module (PSM) to resolve priority and route data from each DLP to the appropriate host. A base also contains a Base Control Card (BCC), to identify and control access to the base and to the DLPs in the base, and a Maintenance Card (MC), to connect the base to the maintenance test bus.


DLP PERIPHERAL CONTROLLER
DC ALLOWS ACCESS FROM HOST; CAN HAVE UP TO SIX DCs FOR SIX HOSTS
LSP A PROGRAMMABLE DLP, FOR DATA COMMUNICATION SERVICES
PSM FOR BASES WITH MORE THAN ONE DC, ROUTES AND PRIORITIZES DATA FLOW
NSP A PROGRAMMABLE DLP, FOR COMMUNICATION TO LSPs, WHICH MAY RESIDE IN OTHER BASES
BCC IDENTIFIES AND CONTROLS ACCESS TO BASE AND DLPs
MC CONNECTS TO THE MAINTENANCE TEST BUS
LEM ALLOWS FOR ADDITIONAL BASES TO BE ATTACHED TO HOST MLI
PROCESSOR
MLIP
PORTS
TO OTHER BASES
HOST
BASE
PSM
DC
DC
DLP
LSP
NSP
BCC
MC
LEM
HOST MLI
FROM ANOTHER HOST
MLI
TO LSPs, WHICH MAY RESIDE IN OTHER BASES
TO OTHER BASES

Logical I/O

The logical I/O subsystem is concerned with structured collections of information called files that can be manipulated by means of I/O statements. The subsystem includes parts of the MCP, formatting libraries, and other parts of the software that deal with logical I/O operations.

The logical I/O subsystem presents data at the record level and makes operational details about the file system and the I/O subsystem transparent to the user program. For certain peripherals such as printers, a program can treat the peripheral as a logical file that is almost independent of the kind of printer used. The program then performs I/O operations to the file rather than to a specific device. Details of device control are handled by the physical and logical I/O subsystems, and peripheral assignments can be delayed until run time. Because of this device independence, changes can be made to the system configuration without changing existing programs.

Input/Output Operations

The I/O subsystem transfers data formatted into two types of structures: records and files. A record is an organized and identifiable set of data containing strings of characters, groups of binary words, or both. The subsystem moves data in blocks between primary storage and memory, but passes data to and from programs one record at a time, moving records from memory to the user work area and back.

To expedite the flow of I/O operations, the I/O subsystem makes use of buffers. Buffers are intermediate storage areas controlled by the I/O subsystem and used to store data when it is in transit between the physical file and the user work area.

A file is a group of related, ordered records. Most of the communication between programs and the subsystem concerns files. Files are both physical and logical. A physical file is the file as it is stored on a recording medium, such as a disk or tape. It generally has a specific structure influenced by the type of medium. A logical file is the file as seen by a program. It exists only within that program and has no inherent properties until it is associated with a physical file or is described by file attributes. The logical file, through file attributes, defines the properties of the physical file it creates. Multiple logical files can be associated with one physical file, as in a database, even when the attributes of the logical files are not identical.

Both the logical file and the physical file have file names. The physical (external) file name is recorded in a directory for disk files and in a tape label for tape files. The combined logical and physical I/O subsystem maintains the disk directory and tape labels, updating them as necessary. The logical (internal) file name is a file variable declared within a program and known only within that program or within programs to which it has been passed as a formal parameter.

The I/O subsystem acts as an intermediary between a logical file and the associated physical file, establishing a connection between the two as necessary. When a new logical file is created, the subsystem finds and provides storage space for the associated physical file or provides access to a requested kind of peripheral.

A file as seen by a program has a particular structure that depends on the format of the records and the way they are accessed. Records can be fixed or variable in length. The I/O subsystem must know the length of a record in order to pass one record at a time to a program. For fixed-length records, the program directly or indirectly declares the length. For variable-length records, the length is usually stored in some location in the record or specified in the I/O statement.

Access to files can be sequential or random. During sequential access, records are processed in consecutive order. Random access allows direct access to a record without first accessing the preceding records. To access a fixed-length record randomly, the logical I/O subsystem calculates the position of the new record and moves to it. To access a variable-length record randomly, the subsystem can use a "key," which is a field in the record that can be used to identify that record. A separate index associates each key with a pointer to the record's actual location.

COBOL74, RPG, and Pascal files with records sequenced by keys can be accessed in both sequential and random modes through use of the system library SYSTEM/KEYEDIO. KEYEDIO uses a hierarchy of indexes to locate records for both types of access. In KEYEDIO files, sequential records are not necessarily stored next to each other. Instead the keys are indexed in sequential order and used in that order for sequential access of the records. A further discussion of KEYEDIO can be found in the "System Software Support Reference Manual."

File Attributes

In a multilanguage, multiprogram environment, most files cannot be viewed as the simple property of a single program. Because of data communications, time-sharing, and database management applications, files have become system components often accessed by a wide variety of programs. Therefore, it is necessary to have some method of managing files that is system wide and language independent. File attributes is the method used on Burroughs systems.

File attributes are control parameters that contain all the information the I/O subsystem needs to connect the correct physical file to a logical file and to process the file after the connection has been made. They are used for the following purposes:

- o To identify a file
- o To indicate file structure
- o To control file access
- o To give the current status of a file
- o To automatically translate the characters in a file
- o To return diagnostic information about attribute consistency and about physical I/O operations

Values for file attributes can be specified in a number of ways. Programming languages provide explicit methods through file declarations. The I/O subsystem allows dynamic specification of attributes when the program is compiled or executed through use of Work Flow Language (WFL) file equation statements. In addition, some attributes can be changed during program execution either through ODT commands or through statements in the program.

For further information on the I/O subsystem, see the "I/O Subsystem Reference Manual" and the "Physical I/O Overview" manual.

DISK SUBSYSTEM

The disk subsystem is one of the most extensively used subsystems. It is responsible for the movement of data to and from disks and the structure and the organization of information on those disks. The subsystem consists of the following components:

o The disk or disk drive media

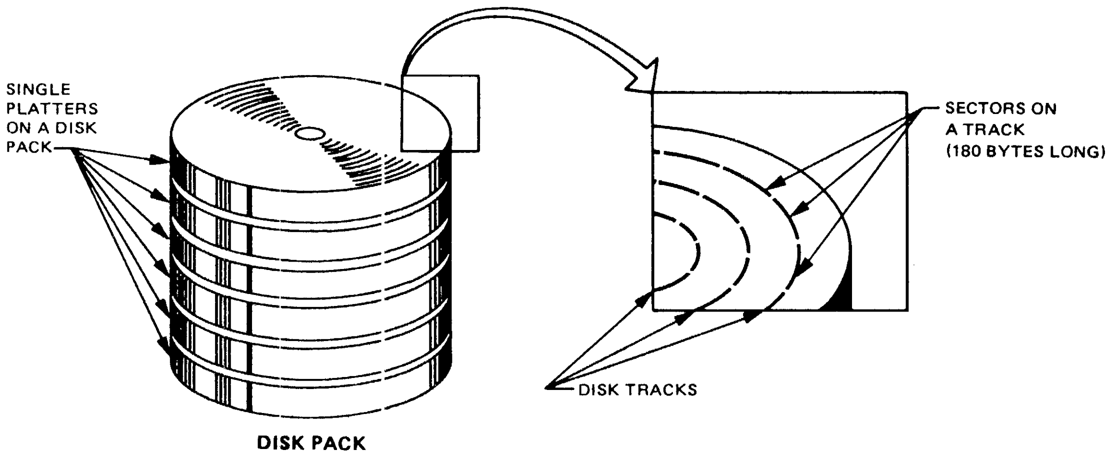
o The disk drive units

o The disk drive controller that controls the disk drive units and transfers information between the host system and disk drive units

o The I/O controller that provides the interface between the host system and the disk drive controller

o The software that controls the structure and operation of the disk subsystem and provides the structure for the information stored on the disks

A disk volume or disk pack is a data storage medium that consists of a circular platter (or circular platters stacked vertically on a central spindle) containing magnetic bits of data stored in concentric circles called tracks. A disk pack consists of multiple platters stacked vertically on a central spindle. For this discussion, the term "disk" includes disk packs.


SINGLE PLATTERS ON A DISK PACK
SECTORS ON A TRACK (180 BYTES LONG)
DISK TRACKS
DISK PACK

Data is written to and read from disk volumes on a disk drive that has one or more read/write heads. These heads are positioned over the desired tracks of the spinning disk so that information can be stored and retrieved from those tracks.

Disk tracks are divided into sectors that are 30 words (180 bytes) long. When you create a disk file or add to it, the data is stored in one or more areas. An area is a contiguous group of sectors, allocated for that file when it is created or expanded. An area is also referred to as a row by the MCP. Areas are defined in terms of logical records, while rows are defined in terms of sectors. You can control the size of an area through the AREALENGTH file attribute and can control the number of areas that can be allocated through the AREAS and FLEXIBLE file attributes. The system allocates physical disk space for areas when they are needed to hold data records written to the file.

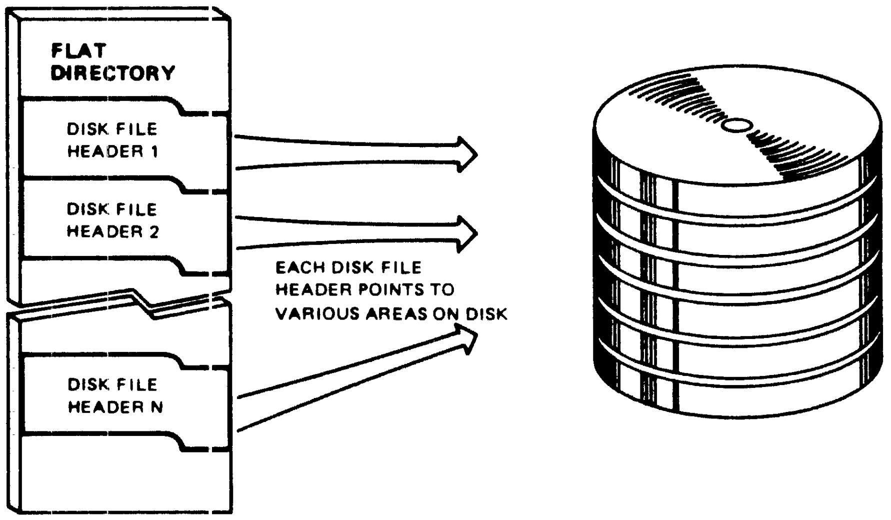
Files on a disk are organized through a flat directory, which is a table of contents for the disk that contains critical information about each file. The flat directory is also referred to as the system directory. Individual file information is stored in disk file headers in the flat directory, one for each file. The headers contain the file name, dates of creation and modification, the kind of file, access security level, file size, record size and structure, and physical location on the disk of the areas for the file, plus housekeeping information.

Disks are organized logically into families. Each family is a disk or a collection of disks with a common name and a common file directory. The system treats each family as a single entity and can spread files destined for that family across all the members of the family.

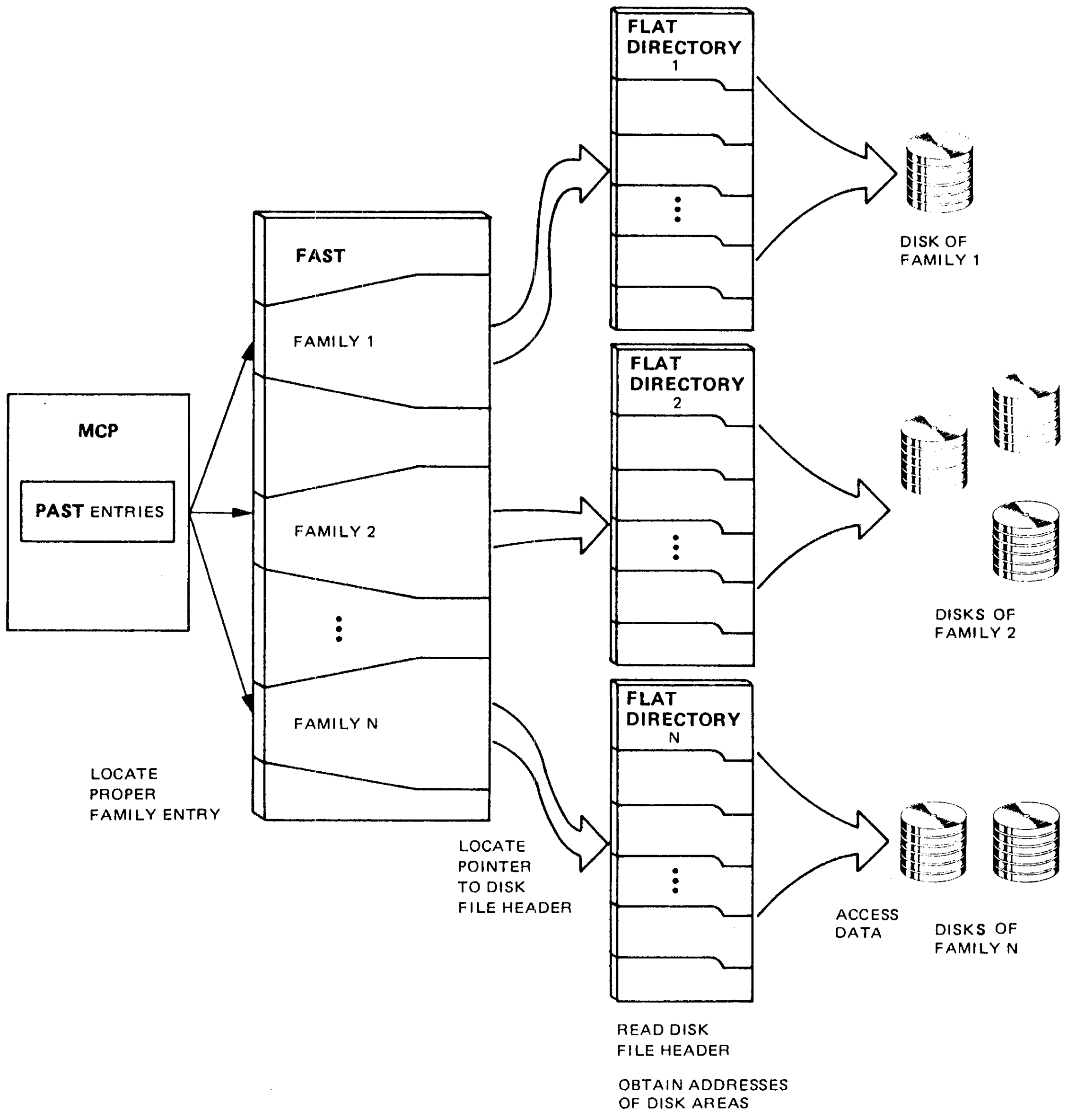
However, the flat directory cannot be spread over the family. The family member holding the directory is called the base pack, and additional disks added to the family are known as continuation packs. Directories can be duplicated on other members of the family, and in this case, any of these members can be the base pack. Both types of packs are added to the configuration through use of the RC (Reconfigure Disk) ODT command.

The system assigns each member of the family a number, the family index, at the time the member is added. If there is a single directory, the base pack is assigned family index 001, the first continuation pack is 002, and so on. If there are duplicate directories, the first family member with a directory that the system encounters is identified as the base pack. A program can use the FAMILYINDEX file attribute to distribute files or areas of files to particular disks in the family.

Access to files in the disk subsystem is through a flat directory for each family and through a special access structure for the entire system. The flat directory, shown in the following figure, resides on the base pack and contains disk file headers for each permanent file on the family. However, these disk file headers are not ordered by name. Therefore, to locate files quickly, the system uses the special access structure.



The access structure consists of two parts: the Pack Access Structure Table (PAST) and the File Access Structure Table (FAST), shown in the following figure. The PAST contains pointers to the FAST. These pointers indicate where in the FAST the entry for a family's files are stored. The FAST contains a pointer to each disk file's header in the flat directory for that file's family. The system obtains the physical address of the file from the disk file header. Using this access structure to access files increases system efficiency.



At system initialization time, the system compares the contents of FAST with the disk families that are on-line. Any on-line disk packs missing from the FAST are automatically added to the FAST by the system. Any unvolumed families on the FAST that are no longer on-line are deleted from the FAST. Subsequently, when a family is brought on-line or taken off-line from the system, the MCP adds or deletes the corresponding file entries to or from the access structure.

Before a disk volume can be used for the first time, two procedures must be used to prepare it: the Initialize, Verify, and Relocate (IVR) procedure, and the Reconfigure Disk (RC) procedure. Before a disk is delivered to a customer, Burroughs uses the IVR procedure to write sector boundaries and a label on the disk. At customer sites, the IVR procedure should never be used except under the direction of a Burroughs field engineer.

Before your installation uses a disk volume, you must use the RC (Reconfigure Disk) ODT command to prepare the disk for system use and to assign a family name and serial number to the disk. The RC procedure erases all the files on a disk and creates a new label and a new directory.

If you want to change the name or serial number of a disk without erasing the data on it, use the LB (Relabel Pack) ODT command.

The system constantly monitors the directories for discrepancies and automatically attempts to eliminate any problems it finds. The MCP maintains data integrity by testing all directory records before they are used through tests such as checksum words. If the system detects a directory error, it reports the problem with one or more messages and then usually attempts some type of automatic error recovery. To prevent any disruption that a directory error might cause, you can set up a duplicate directory on another disk in the family. These duplicate directories require only a modest amount of extra disk space. Directory handling may be slowed slightly because more than one copy of the directory must be updated.

Mirrored Disk

On large systems with Data Link Processor (DLP)-based I/O, the Mirrored Disk feature allows the parallel functioning of two to four disk packs so that they are a "mirrored set," that is, exact copies of each other.

By creating and maintaining duplicate copies of important packs, the Mirrored Disk feature increases both system availability and data integrity. In the event of an error on one mirror, application programs using the mirror set proceed normally, and the operator is notified of the error. Disk mirroring is totally transparent to application programs.

Mirroring disks also improves I/O throughput on disk subsystems that experience a high ratio of read operations versus write operations. Mirrors of existing disks can be created without bringing disks off-line or interrupting use by the system.

The Mirrored Disk feature applies to all supported disk types, except head-per-track disks, and can be used with both Cataloging and non-Cataloging file systems. "Cataloging" in this section explains the Cataloging feature. The "Disk Subsystem Software Overview Manual" provides more detailed information.

No special hardware is required for disk mirroring. Disk units must be available, however, to allow for the redundancy. To create mirrored disks, you specify which disks are to be mirrored, the number of disks in a set, and the number of units on which the mirrored disks are to reside.

Once created, mirrored sets are maintained automatically across system interruption. Operator action is required only in certain exception conditions.

You can remove mirrors from any mirrored set, create new mirrors for any set while the system is in operation, or move mirrored sets between systems and within systems.

If you select the Mirrored Disk feature for some or all of the disks in a system, you must set the MIRRORING option. This done by entering the "OP+MIRRORING" command, then Halt/Loading the system to create the internal structures needed for mirroring. Thereafter, you can create mirrors of existing disks using the "MIRROR CREATE" ODT command. Other MIRROR commands will also be valid.

I/O operations for mirrored disks are handled differently from those for nonmirrored disks. Read operations are issued to only one disk within a mirrored set. Write operations are issued to all disks within a mirrored set.

For more information on the preceding topics and on the disk subsystem, see the "Disk Subsystem Software Overview."

PRINT SYSTEM

The Print System, known as PrintS, is that part of the Master Control Program (MCP) and related system software that deals with the output of files to peripheral devices such as printers or punches. PrintS has an optional part, Remote Print System (ReprintS), that deals with remote (datacomm) printers.

PrintS can be used by users, programmers, operators, and system administrators. By using file and task attributes through WFL statements, or by entering commands, PrintS allows you to control the printer to which the file will be sent, the number of printed copies, file security, the backup file title, and other file characteristics.

ReprintS extends the features of PrintS to include printers at remote destinations that are connected to the host computer through datacomm lines. All PrintS commands can be used on a remote device in the same way they are used on an on-site device. The remote destination must be a datacomm station controlled by MARC/COMS. Note that to use ReprintS, you must have either COMS or COMS (Entry).

When you run a program that generates output, the output (or printer file) can be either printed on-line or spooled for later printing. The file directly routed to the printer, called direct printer file, is printed as your program writes each line of output.

The spooled output is first sent to a disk or tape, creating a backup file. Spooling dissociates the creation of output files from the delivery of the files, allowing each to proceed at a different pace. When spooling is in effect, both the processor and peripherals are used more efficiently, because spooling ensures that a peripheral is not tied to a job for the job's duration, and a job does not have to wait for a peripheral to finish its task.

A system administrator implements spooling by enabling two system options, LPBDONLY and CPBDONLY, that place the system in automatic backup mode. When in this mode, the system automatically changes printer or punch files to backup files and optionally generates file names with a prefix of "BD" for printer files or "BP" for punch files.

Spooling is controlled by PrintS, which consists of utility programs and internal routines controlled through ODT (Operator Display Terminal) and WFL (Work Flow Language) commands and through file and task attributes. In addition, the PrintS commands and functions are available through MARC (Menu-Assisted Resource Control) menus. The number of functions available to an individual user depends on the privilege level of that person's usercode. PrintS is concerned only with spooled output. Direct printing or punching bypasess PrintS.

Once the backup files are created, the Print System routes the backup file to a printing device such as a line printer or an image printer. Files to be printed can be printed either automatically or manually. Automatic printing is done through file attributes, which programmers can declare in programs or WFL jobs. Jobs are printed either in numerical order by job number or in order of size, with the smallest being printed first. You control which order is in effect through the BACKUPBYJOBNR system option, set through the OP (Options) ODT command.

Manual printing is done in three ways by means of the PRINT statement: users specify the PRINT statement interactively through MARC or CANDE, operators use the PRINT statement on the ODT, and programmers use it through a WFL job. The PRINT statement can override any previously declared file attributes.

For more information, see the "Printing Utilities User's Guide."

Controlling the Print System

In addition to the commands previously discussed, there are several other ODT commands and a utility program used to control the Printing System. The commands allow you to stop printing of a job, to change the order in which jobs print, to substitute backup media, and to force printing or punching of a backup file. The utility allows you to create custom character set tables for printers.

You can alter the choice of backup media through three commands.

1. The SB (Substitute Backup) ODT command changes the backup media originally specified by the file attribute BACKUPKIND. The SB command allows you to adjust workload distribution to accommodate current conditions. For instance, if the print queue is unusually long, you could send some of the large backup files to tape and print them at a later time or on an off-line printer.

2. The OU (Output Unit) ODT command forces a direct printer file to a backup file. It is used to respond to an MCP "RSVP" message.

3. The DL (Disk Location) ODT command is used in conjunction with the SB command to direct a backup file to a particular disk family. The SB command must list DLBACKUP as the destination if the DL command is to be effective.

A trainid is the name of a character set used by a printer. For a printer to use a particular character set, a train table or a translate table for that set must reside on the system. The train table equates each character in a set to a hexadecimal number, and the translate table equates each character sent to the printer to a hexadecimal number. In both tables, the hexadecimal number indicates a position on the printing device that corresponds to the desired character.

In most cases the standard character sets supplied with the system are all that a site will need. If nonstandard sets are needed, you use the LTTABLEGEN utility to generate custom train or translate tables for those sets. To inform the system which trainid to use for a particular job, you use the TRAINID backup file attribute. The "System Software Site Management Reference Manual" discusses the LTTABLEGEN utility.

For more details about Print System features and commands, refer to the "Print System (PrintS/ReprintS) User's Guide." The "Operator Display Terminal (ODT) Reference Manual" provides information on the ODT Print System commands.

INTRINSICS AND LIBRARIES

Intrinsics are software routines that perform functions essential to the execution of a program. They can be physically separate from a program but need to be present in order for that program to execute. Procedures that format data or derive square roots are examples of intrinsics. Intrinsics were formerly part of the MCP but now, with a few exceptions, are grouped into related sets of functions known as system libraries.

System libraries can be accessed by both system and application programs. This run-time access of a library can be thought of as a temporary binding of the library to the program.

Libraries provide a very powerful tool for the development of application systems. More than one program can share a given library, and a library can call other libraries. An advantage of libraries is that program creation and maintenance is simpler, because the program structure is more visible.

Libraries need not be on the same pack as the MCP but can reside on any pack in the system, making it easier to distribute the I/O load throughout the disks on the system. You are not limited to Burroughs-supplied libraries; you can create and use your own libraries of procedures.

A site can create and install additional intrinsic functions and their associated libraries. Function names are associated with library file names through the SL (System Library) ODT command. You can also use this command to remove or display existing functions.

Burroughs supplies a General Support library with the system. It contains commonly used mathematical and I/O formatting intrinsics that can be used by programs written in any Burroughs language. This library must be present on the system.

Two other types of system libraries provided by Burroughs are language and communication libraries. Individual language libraries contain special routines commonly needed by programs in that language. Examples are file and string manipulation, array packing, and run-time error handling procedures. Their use enables the compiler to generate shorter code for programs. The PL/I Support library must always be present because it is used by the General Support library. The libraries for other languages must be present when programs in those languages are run.

Communication libraries contain supporting routines for datacomm and for products such as BNA, IDC, and COMS and are supplied with these programs. These libraries must be present when the associated products are used.

The "System Software Utilities Reference Manual" contains additional information on libraries and how to use them.

SORT Intrinsic

The SORT intrinsic is a procedure in the MCP that sorts a file or a set of records into a single file of ordered records, or merges a set of presorted files into a single ordered file. SORT can be called from ALGOL, COBOL, COBOL74, and PL/I programs or from the Sort language. The Sort language exists to allow you to sort or merge files through direct access to the MCP SORT procedure. Sort programs can be created through CANDE or WFL, and can be stored for later use. The "System Software Utilities Reference Manual" explains the SORT intrinsic, and the "Sort Language Reference Manual" explains the Sort language and programs.

SYSTEM PERFORMANCE MONITORING

Because your installation represents a major investment for your organization, you need to determine how your system is used, how it is performing, and where changes, if necessary, can be made. This type of information can be obtained by monitoring the system and system performance. The knowledge gained by monitoring also allows you to increase or fine-tune performance for maximum efficiency and to detect any degradation in performance caused by procedure changes or hardware problems.

On large computer systems like the A Series systems, many factors affect system performance. Effective manual monitoring of all of these factors would be impossible, so Burroughs provides software and Operator Display Terminal (ODT) commands to log, retrieve, display, and print information about pertinent events. These tools consist of the following:

- The System Management Facility II (SMFII)
- The BARS utility
- The LOGGER and LOGANALYZER utilities
- The SUSPENDER utility
- The U (Utilization) ODT command
- Other ODT commands
- The internal routines GETSTATUS, SETSTATUS, and SYSTEMSTATUS

System Management Facility II (SMFII)

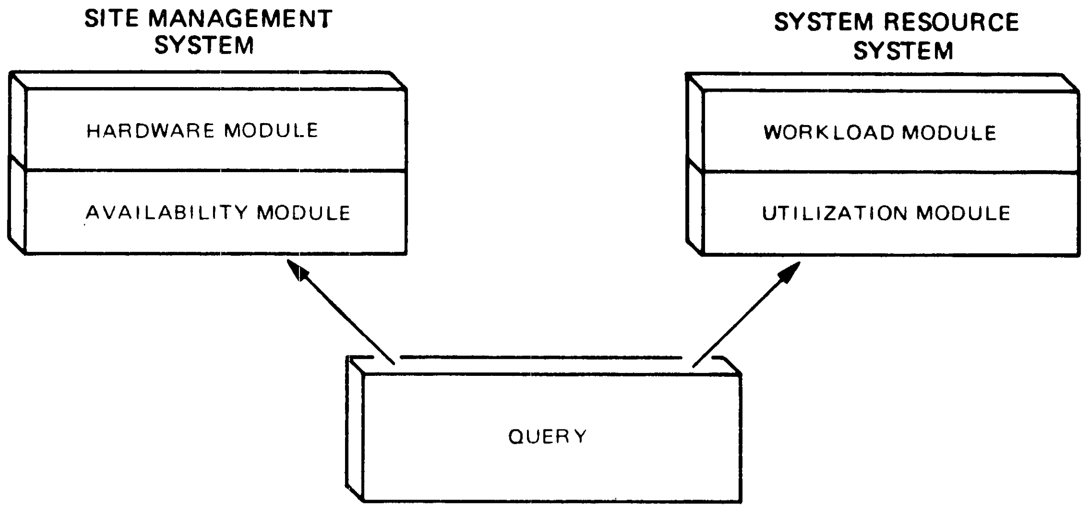
The System Management Facility II (SMFII) is a software system that monitors and provides data on four areas of system performance:

1. Hardware performance
2. Software performance
3. Workload characterization
4. System utilization

A User's View of System Functions

SMFII creates and maintains a database composed of statistics on the four areas of performance. The collected data is accessed, and reports formatted and generated, through the QUERY program.

SMFII consists of three major components, as shown in the following diagram.



1. The Site Management System is used for monitoring the operating condition of the system's hardware and software components. This component contains two modules:

   a. Hardware module--contains information about the occurrence and nature of system-detected faults in mainframe and peripheral devices detected during normal operation, such as I/O errors

   b. Availability module--contains information about the occurrence, nature, extent, and resolution of events that interfere with total system use, such as equipment malfunctions and operations problems

2. The System Resource System is used for measuring the system workload and utilization of system resources. This component also contains two modules:

   a. Workload module--contains data about the programs run on the system and about user resource demands, such as program file access

   b. Utilization module--contains information about program use of hardware and software resources, such as memory management and data communication statistics

3. SMFII/QUERY is used for the analysis of the collected data and generation of reports. QUERY accepts report specifications from the user, analyzes data, and produces graphic and statistical reports in a variety of formats. QUERY can be used interactively or in batch mode.

The Site Management System and SMFII/QUERY are released to all customer sites as a part of the standard System Software Facility, while the System Resource System is a separately priced item.

Three programs build and maintain the SMFII database:

1. LOGCONSOLIDATOR, which extracts data from the Sumlog files for the Hardware and Workload modules

2. SAMPLER, which collects real-time performance information from the MCP about the use of hardware and software resources. This information is for the Utilization module or for direct graphic display.

3. SITEINPUT, which collects data from site personnel for the Availability module

You use SMFII by running these data collection programs to build a database, then running QUERY to examine and analyze the data.

Information about SMFII can be found in the "SMFII System Resource Management Manual" and information about QUERY can be found in the "SMFII Query Program Manual."

The BARS Utility

BARS is a real-time utility program that monitors the system's performance and displays performance statistics as numeric values and bar graphs. The items displayed are those concerning current Central Processor Unit (CPU), memory, I/O, and disk pack use. The display is updated periodically and automatically contains only those items that apply to the system on which BARS is running.

You can select the items to be monitored from a comprehensive list, or use the default list supplied with the program. Newly created lists can be saved for later reuse.

You would use BARS when you want to monitor what is happening at the moment rather than viewing system history, as with SMFII. You can use BARS interactively or in batch mode. In batch mode, the utility writes the performance data to the file specified when the utility is started. You then provide an application program to view the file contents. BARS can be accessed through MARC menus or through CANDE.

Additional information on BARS can be found in the "System Software Site Management Reference Manual."

LOGGER and LOGANALYZER

LOGGER and LOGANALYZER are utility programs for examining system log files. LOGANALYZER displays all or selected parts of the Sumlog or other log files. It does not manipulate the data in any way other than formatting it. You would use this utility when you want information on a specific job or peripheral unit, or if you want to find out what was happening during a specific time period on the system.

The nature of the system, whether Actual Segment Descriptor (ASD) or non-ASD, is stored in the Halt/Load record of the Sumlog. LOGANALYZER uses this record to inform the user if the Sumlog came from an ASD system or not. Likewise, the Print System also uses this record to print this information on the banner page of every printout.

LOGGER generates reports to aid in the analysis of system performance and utilization. It extracts user-selected data from the Sumlog files and can summarize, total, average, and sort the data to produce a report. LOGGER is less powerful and flexible than SMFII but is useful for producing charging and billing information. Both LOGGER and LOGANALYZER are described in the "System Software Site Management Reference Manual."

The SUSPENDER Utility

The SUSPENDER utility program prevents the system from becoming overloaded by running too many non-swappable tasks simultaneously. SWAPPER is a facility that allows installations to service users in a time-sharing environment. Many tasks compete for a relatively small amount of memory in a time-sharing environment. To service all users adequately, tasks are swapped from memory to disk when they can no longer run or have exceeded a specified time-slice.

To prevent the system from becoming overloaded, the SUSPENDER utility monitors system performance and uses that information to keep running statistics on the use of the processor. The utility suspends tasks when the statistics indicate that the processor has too much work and resumes the tasks when the processor idle time increases. SUSPENDER must be run under a privileged usercode, and should not be used if the MCP option OLAYGOAL is set to a value greater than 0.

The Utilization Command

The U (Utilization) ODT command displays current statistics on how the system is being used. The display is in two parts. The first part, Processing Utilization, lists the percentage of time the central processor spent performing each of eight different categories of tasks during the last time interval. The length of the time interval is controlled through the SBP (System Balancing Parameters) ODT command.

The second part, Input/Output Utilization, lists the number and rate of user, MCP, datacomm, and total I/O operations, plus the average number of I/O interrupts, that occurred during the last time interval.

ODT Commands

You can use certain ODT commands to monitor system performance. Commands such as ADM (Automatic Display Mode), CU (Core Usage), and PER (Peripheral Status) provide real-time information about the system. The "Operator Display Terminal (ODT) Reference Manual" lists the applicable commands in the "Functional Command Groupings" appendix under "Automatic Display Mode & ODT Control Commands" and "Job Queue Commands."

GETSTATUS, SETSTATUS, and SYSTEMSTATUS

GETSTATUS and SETSTATUS are intrinsic routines within the MCP. GETSTATUS retrieves information about the job or task mix, peripheral and disk unit status, MCP and configuration status, and the files in the disk directories. SETSTATUS provides the system interface for an object program that controls MCP mix, unit, and operational functions.

Both routines can be called only by a DCALGOL MCS or a DCALGOL user program executing under a privileged usercode or with privileged program status. An exception to this is the GETSTATUS directory query function. It can be used by a nonprivileged user to examine data about files accessible to that user.

GETSTATUS and SETSTATUS are used by performance-monitoring utility programs such as BARS and are available for use by sites that want to write their own utility programs. The "DCALGOL Reference Manual" contains information on these two intrinsics.

SYSTEMSTATUS is an intrinsic MCP routine that gathers many different types of information concerning the activity and environment of the system. SYSTEMSTATUS returns groups of related information, locking the system while it extracts the data. You would use SYSTEMSTATUS when you want comprehensive information concerning a particular area of the system, such as queue and swapper or hardware configuration information. This routine can be called only from a DCALGOL program with privileged program status. The "Systemstatus Reference Manual" describes this intrinsic routine and explains how to use it.

The MARC Menu System

Many of the performance utility programs and functions, including ODT command functions, are available through the MARC menu system. The actual functions available depend on the security status of the usercode under which you log on. Usercodes with both Systemuser and Privilegeduser status have the most functions available.

SOFT RECONFIGURATION

The hardware resources used by any one system are collectively known as a configuration, and to configure a system means to allocate these resources. Soft reconfiguration refers to the ability to allocate and later reallocate hardware resources to the configuration through software.

For A Series systems, soft reconfiguration is concerned with memory partitioning and sharing of I/O subsystems. Information concerning these areas is placed in a configuration file, which is a dynamic record of resource allocation. The system then uses this file to modify system tables at initialization or after a Halt/Load operation.

The following paragraphs describe soft reconfiguration at an overview level. For detailed information concerning soft reconfiguration, see the "Softconfiguration Document" on the Documents tape.

ASD Extended Memory

The Actual Segment Descriptor (ASD)-based memory subsystem is implemented by the MCP/AS operating system and supports a single large memory structure that requires no partitioning. ASD extended memory architecture permits the memory resource to be increased to as much as 4 billion words (24 billion bytes) on most systems.

Memory in the ASD architectural design is organized as a monolithic storage area. Since the memory structure requires no partitioning, programs have full visibility to each other while executing (they are not placed into closed partitions of memory), and the MCP can address very large amounts of memory through a table that points to all locations in memory, virtual or physical.

All allocated memory areas are accessed and controlled through a central structure called the ASD table. This table, which can contain up to one million entries, is a central source of information that defines all memory in use by the system.

ASD memory management is based on a distinction between an "actual segment descriptor" and a "virtual segment descriptor." In the context of ASD architecture, the term "descriptor" refers to both the data descriptor and the segment descriptor. On an ASD system, the descriptor is a virtual descriptor because it merely points to the ASD for the memory segment it represents. It is the ASD, a multiword structure, that contains all information regarding the memory segment, such as its memory address, length in words, and the location of its original descriptor for the area. Instead of an address, the descriptor on an ASD machine contains an ASD number. This number is the index into the ASD table on which the ASD for the memory segment resides.

For more information, refer to the "Memory Subsystem Overview."

ASD systems support all user software developed for earlier Burroughs large systems. Thus, all programs compiled to run on an A Series system with a Mark 3.4.1-or-later compiler can run on an ASD system without recompilation or program change. Databases must be compiled with a Mark 3.5-or-later release of DMSII software.

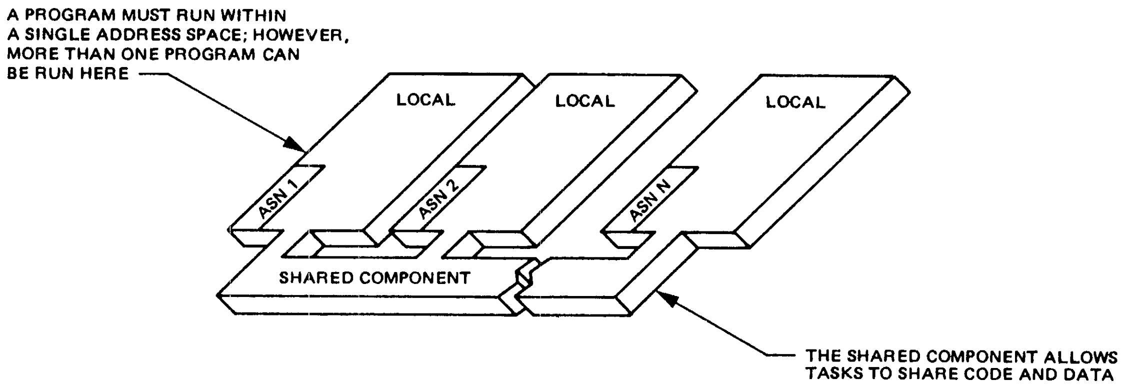
Burroughs A Series systems using Address Space Number (ASN) architecture can be upgraded to ASD architecture through the installation of the Master Control Program/Advanced Systems (MCP/AS) operating system and associated microcode. Some systems may require some hardware changes.

Memory Partitioning

The addressing mechanism in A Series systems with the standard MCP can address a maximum of 1024K words, but the systems can hold many times this amount of memory. To be able to access all of memory, the MCP divides memory into address spaces of 1024K words each and assigns each address space a unique Address Space Number (ASN). This number allows the MCP to address memory locations relative to the address space number and thus access all of memory. A program must run entirely within a single address space, but more than one program can occupy an address space at any given time.

Each address space consists of a shared and a local component. The shared component is common to all address spaces and exists so that tasks can share code and data. The local component is unique to the address space and is identified by the ASN. The sizes of both the shared and the local components are dynamically controlled through statements in the configuration file (described later) and through ODT commands.

The size of the shared component is important, because the MCP and most MCP functions run in the shared component. If the shared component is too small, the MCP will run slowly. If it is too large, the local component will be small, possibly causing programs in a local component to run slowly.


A PROGRAM MUST RUN WITHIN A SINGLE ADDRESS SPACE; HOWEVER, MORE THAN ONE PROGRAM CAN BE RUN HERE
LOCAL
LOCAL
LOCAL
ASN 1
ASN 2
ASN N
SHARED COMPONENT
THE SHARED COMPONENT ALLOWS TASKS TO SHARE CODE AND DATA

Sharing of I/O Subsystems

I/O subsystems can be shared by several systems through the sharing of I/O base modules and peripherals. An I/O base module is the interface between peripheral/datacomm devices and one or more host systems. It contains the Data Link Processors (DLPs), which connect to and control the peripheral devices used by the systems, plus one or more distribution cards to connect the hosts to the base.

The Configuration File

The configuration file is a description of the hardware resources that make up a configuration. In this file, you predefine the different hardware configurations that your system will need. For example, if your system is sharing peripherals with another system, you might define several configurations one with all of the peripherals and others with only selected peripherals.

The system manager builds a configuration file by creating a file and using the GROUP statement to place the configuration definitions in the file. The GROUP statement is divided into several sections, two of which pertain to A Series systems: the global memory (shared memory) section and the peripherals section. The global memory section applies to non-MCP/AS systems only.

The global memory section specifies the size of the shared (global) portion of memory on systems with more than 1024K words of memory. This is stated as the number of 128K pages of memory to use. An address space consists of eight pages, so to find the size of the local component of each address space, the system subtracts the number of global pages from 8.

To find the number of address spaces the system will have, you subtract the number of global pages from the total number of pages of memory, and divide the result by the number of pages in the local component.

For example, suppose the total amount of memory in a system is 4096K words, which is 32 pages. If you specify a global size of 5 pages, the local size is

    8 - 5 = 3 pages

The total number of pages is 32, so the number of pages left for local memory is

    32 - 5 = 27 pages

Divide 27 pages by 3 pages per local to find the number of address spaces.

    27 pages / 3 pages per local = 9 locals or 9 address spaces

If the number of pages available for local memory had not been evenly divisible by 3, the last address space would have been smaller than the others. This size difference can cause operational inconsistencies if a program that normally runs in the larger areas is initiated in the smaller area.

The peripherals section of the configuration file specifies the peripherals that can be used by the system. A system will not attempt to use any peripheral that is not on the list, even though a physical connection exists to that peripheral. This restriction allows physical system-to-peripheral connections to remain in place even when an I/O base module is being used by another host.

The Configuration Utility

The Configuration utility uses the configuration file source code to produce the object file that the system uses. This is analogous to compiling an application source file to generate an executable object file. You specify the titles of both the input and the output configuration files when invoking the utility.

Reconfiguring the System

Once the configuration file object code file exists, you must inform the MCP of the title. You do this by issuing a CF (Configuration File) ODT command and including the object file title in the command.

To reconfigure the system once the CF command has been issued, issue a RECONFIGURE (Reconfigure System) ODT command and specify the particular configuration wanted. The system will use the information from the new configuration file to build or modify the system information tables.

The configuration can also be modified through the use of the FREE (Free Resource) and ACQUIRE (Acquire Resource) ODT commands. These commands change the current configuration of a system but do not alter the defined configuration in the configuration file. The changes from these commands remain in effect after a Halt/Load but not after execution of a RECONFIGURE command. The FREE and ACQUIRE commands are a more flexible way to temporarily reconfigure the peripherals on a system, because peripherals not listed in the configuration file can be added to the system.

4 PLANNING FOR EFFECTIVE OPERATIONS

This section discusses operational procedures and issues that you need to be aware of and plan for before your system arrives. It is meant to be read before you order your system, especially by first-time users of A Series systems. The entire section should be read before you start planning for your system.

The section is directed to system administrators and users who will be performing the functions of system administrators. The information presented here is not meant to be a "how to" but is intended to help you plan for the initial, day-to-day, and occasional operations performed on A Series systems.

4.1 INITIAL PLANNING

Before you order your system, you need to consider and plan for one and possibly two important areas: the configuration of your system, and progression from your existing Burroughs system or conversion from a non-Burroughs system.

SYSTEM CONFIGURATION

Configuring your system involves more than calculating the number of devices you need for your anticipated workload. It also involves planning for fail-safe operation if this is needed, for emergency situations, for equipment maintenance time, and for possible equipment malfunctions.

If fail-safe operation, where the system must run nonstop, is a requirement, then you may want to duplicate certain parts of the system, such as I/O bases or peripheral equipment. Both the hardware and software in A Series systems are designed to support redundant operation for all data communication and I/O subsystem components. For example, I/O bases can be permanently connected to several systems at one time, as can datacomm Network Support Processors. Although only one system at a time can use them, these devices can be reconfigured through software to serve as alternate communications paths for the other systems as needed. Use of these devices by several systems is also a way to achieve redundancy and keep costs low.

If a certain type of device is needed at all times while the system is running and cannot be spared even for preventive maintenance, you should consider ordering a backup device for it. Any device critical to the functioning of the system at your site is a candidate for duplication.

Physical access to the system should be considered, as well as security. If secured access to the Operator Display Terminal (ODT) is desired, you may need to place the ODTs in a secured room, away from the rest of the system. In that case, you might want to order extra terminals to place next to the system. Conversely, you may want to make a terminal and printer available for general use, in addition to the assigned terminals. You should also consider emergency power supplies if you need uninterrupted operation at your site. The emergency system should include automatic startup and transfer to the emergency power source.

PROGRESSION AND CONVERSION

If your site is converting from a non-Burroughs system, you can contract for services to assist you in the process. Conversion services are determined on an individual basis, depending on the type of system you are converting from and the amount of assistance you need. See your Burroughs representative for information on conversion.

Whether your site is progressing from a Burroughs B 1000 or a Burroughs B 2000/B 3000/B 4000 Series system, or is converting from a non-Burroughs system, you need to plan how to manage your archived files. You can follow either of two methods:

1. Convert the full-backup, archived files to the new system.

2. Make new, complete backups of your converted current files before you start daily processing on your new system. These backups then become the basis of your archiving system.

If you choose to do the latter, you can convert any prior-format files on an as-needed basis.

If at all possible, you should plan to run the old and new systems in parallel for a short time and compare program results. This can help ensure that all programs are performing as required.

PROGRESSION FROM BURROUGHS SYSTEMS

A wide range of services are available to you when you progress from a Burroughs B 1000 or a B 2000/B 3000/B 4000 Series system to an A Series system. These services include requirements definition, education courses, software tools, and programming and system consultation. You can choose selected portions of these services or contract for the total progression effort.

To help you save time and minimize costs if your site is handling most of the effort, progression software aids are available on order. See the "User Guide to B 1000 Progression Aids," obtainable through your Burroughs representative, for the software aids available and information on how to use them. These aids are for both small and medium systems and include file transfer utilities for transferring tapes between systems.

You should also order the "B 1000 Series to A Series Progression Guide." This guide discusses the differences between small and A Series systems, lists the steps in progression, describes common progression problems and solutions, tells you how to transfer files, and describes the steps to take after using the software progression tools.

4.2 PLANNING FOR BASIC SYSTEM OPERATION

Once you have planned your configuration, you need to plan how to start up your system, what system options you are going to use, and how you are going to set up security and data communications, manage job flow, and distribute file storage.

BASIC SYSTEM INITIALIZATION

System initialization is a process used to bring a system to a normal operating condition and make it available for its intended use. When a system is first powered up, the process consists of powering on the hardware, including the peripheral devices; loading the microcode, controlware, and firmware; loading the Master Control Program (MCP) code file and other system software into memory; identifying all resources and initializing them; and transferring control to the MCP.

There are three methods used to initialize the system: cold start, cool start, and Halt/Load. During all three types of system initialization, the system must have access to a copy of the MCP code file to be used, either from a system tape or from a disk on the system. The cold and cool start procedures load that MCP code file onto a disk, designating it as the Halt/Load unit, while the Halt/Load procedure uses the MCP that is already on the Halt/Load unit.

The Halt/Load unit is prepared through the CM (Change MCP) ODT command when the system is running under control of the MCP, and through the SYSTEM/LOADER program when it is not. The Halt/Load unit has special label information in segments 0 to 27 and contains a pointer to locate the MCP tables for the system. It must contain, at a minimum, a disk label and directory, the MCP code files, bootstrap code, and MCP parameter and configuration tables. See the System Software Installation Guide for your type of A Series system for detailed procedures on creating a Halt/Load unit.

A cold start is the most comprehensive form of system initialization. It may be performed by the field engineer when the system is first powered up, depending on the type of system. The system does the following during a cold start:

1. Creates a new disk directory on the Halt/Load unit

2. Loads a copy of the MCP code file onto the Halt/Load unit

3. Creates new MCP tables on the Halt/Load unit

4. Performs a Halt/Load using the new MCP

Because a cold start creates a new disk directory on the Halt/Load unit, all existing files on that disk are erased and must be reloaded if needed. In addition, all system information, including all system options, reserved units, and configuration information, is erased.

A cool start causes the system to load a new MCP code file onto the Halt/Load unit, update existing MCP tables, and perform a Halt/Load using the new MCP. During a cool start, the disk directory is not rebuilt and files on the Halt/Load unit are left intact. A cool start is used to load a new MCP code file or replace a corrupted MCP code file.

A Halt/Load stops all processing in the system and reloads an image of the MCP code file to main memory from the Halt/Load unit. The disk directory remains intact. During a Halt/Load, the system does the following:

1. Terminates all jobs and sessions and stores the information necessary to automatically restart Work Flow Language (WFL) jobs and some programs

2. Reloads an image of the MCP code file from the Halt/Load unit to main memory

3. Reinitializes MCP tables with the exception of job management, restart, configuration, and parameter information

Situations that require a Halt/Load are: changing to a new edition of the MCP, software and hardware problems that leave the system in a "hung" condition, and the occurrence of a fatal memory dump.

AFTER COLD START

Once initialization is complete, there are several tasks you should perform before making the system available for general use. The following list contains the most commonly needed tasks, but because the requirements of each site vary, you may need to add others to the list.

1. Copy the system files to disk and designate the system libraries. See your Software Installation Guide for information on how to do these tasks.

2. Initialize the Menu-Assisted Resource Control (MARC) system. See your System Software Installation Guide for details.

3. Create a USERDATAFILE using the MAKEUSER utility. Be sure to assign privileged usercode status to at least one usercode. See the "System Software Site Management Reference Manual" for the MAKEUSER procedures.

4. Use the Interactive Datacomm Configurator (IDC) to modify the SYSTEM/DATACOMINFO file to reflect your site's datacomm configuration. First read your System Software Installation Guide, then the "Interactive Datacomm Configurator (IDC) User's Guide" for instructions.

5. Set the system time and date through the TR (Time Reset) and DR (Date Reset) ODT commands.

6. Enable or disable system options to customize the system to your site. See the OP (Options) ODT command for the list of options and how to enable or disable them.

7. Make a backup Halt/Load unit, using the method described in the "System Software Installation Guide" for your system. In addition, make copies on tape of the USERDATAFILE, the source for the configuration file, and the DATACOMINFO file. All of these copies are for recovery in the event of system failure.

See the "System Software Installation Guide" for your system for a more detailed explanation of the initialization procedure.

SOFTWARE CONFIGURATION

The next step after planning initialization and the associated tasks is to plan the software settings that affect the efficient running of your system. These include settings for memory factors, system options, security, distribution of files, supervisor programs, job queues, data communications, and multilingual capabilities. The MCP and Master Control Program/Advanced Systems (MCP/AS) operating systems automatically provide complete management of all system resources and tasks.

ASD Systems Memory Management

Memory management techniques vary depending on the memory architecture of the system. Actual Segment Descriptor (ASD) systems manage memory areas automatically. System factors can be adjusted, along with the size of the increments of the ASD table.

The ASD table is the single most important feature of an ASD memory subsystem for a user. The number and sizes of ASDs on a system are assigned through this table. When the number of ASDs on a system is changed, the Memory Management Module automatically adjusts the amount of memory included in each ASD.

The "ASD" ODT command allows you to specify the number of ASDs you want in the ASD table. The "Operator Display Terminal (ODT) Reference Manual" provides information on the syntax for the ASD command.

ASN and GLOBAL/Tightly-Coupled Systems

Each of the three methods of memory management for ASN and GLOBAL/Tightly-Coupled systems--on demand, working set, and SWAPPER--support virtual memory. The SF (Set Factor) ODT command is used to set or modify memory management parameters released to these three methods.

The on-demand method of memory management allocates memory as tasks require space. If memory areas are not available, as a first choice the MCP deallocates and reuses object code areas not currently used. Thus, the area does not have to be written to disk.

The working set method of memory management attempts to anticipate memory allocation needs by maintaining a pool of available memory through aggressive overlay. This is is invoked when you set parameters of the SF (Set Factor) command at the ODT.

SWAPPER is an independent runner of the MCP initiated by the SW (SWapper) ODT command. The SWAPPER method of memory management allows you to set aside a portion of memory called "swap space." Swap space is used for time slicing. Time slicing is a technique where each program in a multiprogramming environment gets control of the MCP for a limited period of time. Refer to the "Operator Display Terminal (ODT) Reference Manual" for a description of the SW command. The section on "SWAPPER" in the "System Software Support Reference Manual" provides a detailed description of SWAPPER.

Setting Memory Factors

The term "memory factors" refers to memory utilization settings that control the way the MCP handles virtual memory. A summary of the factors is given here. For a detailed discussion, see the "Managing Memory" section of the "Memory Subsystem Overview" manual.

There are four factors that can be set to customize memory use for your site: OLAYGOAL, AVAILMIN, FACTOR, and MEM PRIORITY FACTOR. They manage memory by controlling the overlaying of programs to disk and suspending execution of programs when a percentage of available memory falls below a specified limit.

Factor 1, OLAYGOAL, is the percentage of overlayable memory in the system that is to be overlaid every minute. An ideal environment is one in which the overlay rate is large enough to force out unused segments but not ones being used. On a system where the work load is varied and unpredictable, the optimum rate of overlay can be exceeded when the total requirements for the currently running tasks are larger than the amount of available physical memory. This produces a condition known as "thrashing." When a system is thrashing, it is spending an inordinate amount of time swapping program segments in and out of memory, causing overall performance to degrade to an unacceptable level.

The architecture of ASD systems and the large amounts of memory present makes it less likely that thrashing would cause noticeable problems. But thrashing can be experienced on both GLOBAL and ASN extended memory systems.

To prevent thrashing, you can set the OLAYGOAL factor to force a constant overlay rate. Setting the factor to a value greater than zero initiates an MCP procedure, Working Set Sheriff (WSSHERIFF), that periodically examines memory and the overlay rate of each task. On ASN and GLOBAL/Tightly-Coupled systems, the WSSHERIFF provides the working set requirements for each program execution. On ASD systems, WSSHERIFF has been modified to work automatically within the ASD Memory Management Module.

WSSHERIFF suspends tasks that exceed the overlay rate (lowest priority first) until the overall rate reaches an acceptable level, at which time the tasks are reactivated. When the overlay rate is less than the specified limit, WSSHERIFF overlays areas until the specified per-minute rate is reached.

Factor 2, AVAILMIN, determines the minimum amount of total memory that is to be available at all times. To prevent thrashing, the MCP suspends the lowest priority jobs in the mix when the total amount of available memory drops below one-half the value of AVAILMIN. Control programs and programs that are sorting are not suspended.

Factor 3, FACTOR, is used to determine if enough memory is available for a task to be initiated. The system divides the memory estimate for a task by the value of FACTOR to find the amount of memory that must be available. When FACTOR has a value greater than 100 percent, the MCP assumes it has more memory and thus executes more programs. For example, if the core estimate is 5,000 and the value of FACTOR is 200 percent, the scheduling algorithm uses a core size of 2,500 (5,000 / 2.00) to check if enough memory is available. Assigning too high a value to this factor could lead to thrashing, while assigning too low a value could slow down the rate of job flow through the system.

Factor 4, MEM PRIORITY FACTOR, determines the amount of time that data belonging to a higher priority program remains in memory before being overlaid to make space for a lower priority program. This factor protects the program data/code from being overlaid when they have just been put into memory. The factor is expressed as a percentage of one second per increment of priority difference. For example, if this factor is set to 20 percent and a job of priority 55 acquires an area in memory, a job of priority 30 cannot use that area until 5 seconds ((55-30) x .20) have passed. Assigning a high value to this factor slows down the flow of lower priority jobs through the system.

You set the memory control factors through the SF (Set Factor) ODT command. Because each site has different requirements, it is impossible to give recommended values. However, you can start by assigning Factors 1, 2, and 4 values of 0 percent and Factor 3 a value of 100 percent, and then varying the factors one at a time to judge the effect on the system. Use System Management Facility II (SMFII) to check the effects.

Setting System Options

The MCP contains many options that control various system operations. Some of these options are set when the MCP is compiled, while others are set when the MCP is executing.

The compile-time options are set when the MCP is compiled by Burroughs prior to release of the system software. Only in rare situations would a site need to change these options. You can see which options were enabled when your current MCP was compiled by entering the "?WM" command through CANDE or choosing the "WM" menu selection through MARC.

The most important system option is the DIAGNOSTICS option. When this option is enabled, the MCP runs in a debugging mode and may execute extra code, send a message to the operator, or cause a dump to occur when it encounters preset conditions. Debugging mode requires extra overhead and normally is not used by customers unless an unusual situation occurs, such as a problem with new software. For this reason, there are two versions of the MCP on every system software release tape: a regular version, with DIAGNOSTICS disabled, and a diagnostic version, with DIAGNOSTICS enabled.

The run-time options can be set through the OP (Options) ODT command at any time while the MCP is executing. Some options, when changed, do not take until the next Halt/Load. The option settings are stored on the Halt/Load unit. See the "Operator Display Terminal (ODT) Reference Manual" for definitions of the options and an explanation of the command. Once you have decided which options you want to use, it is a good idea to store this information either in a supervisor program (discussed later in this section) or in a written log. This is a precaution only, because, although cold starts are relatively rare, the option selections on the Halt/Load unit are erased during a cold start.

The following list is a suggested starting point for setting the system options. You naturally will want to customize the list for your own site. The options listed are the ones to be set to "true" or "on." Set the remaining options to "false" or "off."

2	TERMINATE	21	NOSUMMARY
4	LPBDONLY	24	OKTIMEANDDATE
5	AUTORM	25	NEWPERETRY
8	AUTORECOVERY	26	LOGPOSITIONING
12	AUTODC	27	SERIALNUMBER
14	CPBDONLY	29	CONTROLOLDWFL
16	CRUNCH		

Setting Up User Identification

Various types of information can be associated with each user on the system. Users can have a usercode for file ownership, be assigned to a disk family, have a default language, (optionally) have charge codes and accesscodes, and have any other items required by the site, including passwords for security.

To associate this information with a user, you use the MAKEUSER utility. This utility updates a file, the USERDATAFILE, which contains one entry for each usercode. MAKEUSER can create a new USERDATAFILE, reinstate an old one, or copy, examine, or modify the current file while the system is in normal operation. The "System Software Site Management Reference Manual" describes the MAKEUSER utility.

Setting Up Security

If your site chooses to implement security, one of the first things you may want to do once your system is up and running is set up the security system. If you have not already done so, read the discussion of security in the "Functional Areas of the System" section to familiarize yourself with the types and implementation of security available.

You may want to limit physical access to the system or to parts of it, especially to the Operator Display Terminals (ODTs) and to COMS control stations. If you do limit physical access, you can place a terminal and printer for general use outside of the computer room and associate them with each other through the PA (Peripheral Association) ODT command. The printer then receives any output for jobs started from that terminal.

If you are using COMS, you can control access to stations, transaction codes (trancodes), and windows, plus you can implement other forms of security through application programs. See the "Defining and Using the Configuration File" section of the "Communications Management System (COMS) Planning and Installation Manual" for information on setting up security on COMS.

To control user access to the system through MARC, COMS, or CANDE, run the MAKEUSER utility program to create usercodes and passwords and, optionally, accesscodes for all users. You can also specify charge codes for each usercode, so that system use can be logged for accounting purposes. The MAKEUSER utility must be run by a privileged user. It can be run through MARC and is described in the "System Software Site Management Reference Manual."

You can limit access to files through the file attributes SECURITYTYPE, SECURITYUSE, and SECURITYGUARD. You may want to limit access to certain system files, such as the DCALGOL and NEWP compilers. You can designate these files as GUARDED or CONTROLLED and use a guard file to designate the nonprivileged usercodes, passwords, and accesscodes through which they can be accessed. See the "System Software Utilities Reference Manual" for information on the GUARDFILE utility.

Distributing Files on Disk

You can influence system performance by the placement of various system and application files on different families. The MCP allows you to select the storage location for the MCP code file, the swapdisk, system libraries, and system files allocated through the DL (Disk Location) ODT command. Your installation can mix some or all system files with application files or can segregate system files.

These decisions involve tradeoffs. For maximum system performance, the ideal would be to store every system file on a different disk and never put any other files on those disks. Obviously, this is not practical. When deciding where to locate files, you need to consider the following criteria:

- o System performance (speed)
- o Amount of storage space needed
- o Ease of recovery from damaged files or media
- o The probability that the failure of a single disk will cause a service interruption

The following recommendations summarize the "Disk File Allocation" topic in the "Disk Subsystem Software Overview" manual. See that manual for an explanation of the reasons for the recommendations.

1. Segregate static (rarely changed) system and application files from dynamic application files by putting them on separate families. This groups files that need frequent backups and separates them from those that do not, thereby simplifying the process.

2. Group the Halt/Load files and the overlay files together and segregate them from all other files.

3. The Halt/Load family should be a single disk family and should not be named DISK or PACK. Reserve the Halt/Load family for the essential files: the MCP code file, the SYSTEM/TRAINTABLES file, and the overlay files. If your site is short of disk space, then place the other system files on this family also. Avoid placing application files on this family.

4. Place the printer or punch backup files on families named DISK or PACK.

Setting Up Supervisor Programs

A supervisor program is a specially designated code file that is run immediately after a Halt/Load. The system automatically enters it in the job mix and runs it before all other jobs. You can use a supervisor program to automatically perform chores that an operator otherwise would have to do. Examples of these chores are checking that system options are set correctly, starting any necessary site-specific programs, and setting up job queues. A supervisor program typically calls other programs to perform these chores and checks that they have been performed correctly.

You designate a code file as a supervisor program through the CS (Change Supervisor) ODT command. See the "Operator Display Terminal (ODT) Reference Manual" for the format of the command.

Setting Up Job Queues

Job queues are waiting lists from which the MCP selects jobs to enter into the mix or the schedule queue. The mix is the list of active jobs in the system, and the schedule queue is a list of jobs waiting for system resources. Jobs in the schedule queue enter the mix as soon as the resources become available.

You define job queues through the MQ (Make Queue) ODT command. When planning job queues for your system, keep in mind the amount of time a job will take. For quick servicing of short jobs, define a queue with a relatively high mix limit, high priority, and restricted use of system resources. Several such queues would greatly speed the turnaround time for these jobs. Batch job queues can have a low mix limit and low priority, allowing the jobs from those queues to be serviced as resources permit.

Before you set up your job queues, read "Managing Jobs" in this section.

Setting Up Data Communications

Configuring the data communications portion of a system involves determining what physical equipment, such as terminals, modems, and Network Support Processors (NSPs), your site needs and then using the Interactive Datacomm Configurator (IDC) to describe the configuration to the system. This procedure is normally handled by the field engineers who install the system.

Start with determining how many Line Support Processors (LSPs) are needed. This then determines the number of NSPs that are needed. In determining these numbers, ensure that the configuration does not have too many lines per NSP and that the bandpass of the NSP is not exceeded. The Data Communications Data Link Processor (DC-DLP), which combines the functions of an NSP and an LSP, supports up to four lines of communication. One logical NSP and one logical LSP must exist in the configuration to represent a DC-DLP. When the field engineer assigns a unit number to a DC-DLP, that number is also given to the NSP. The LSP is given an unused LSP number.

The steps in configuring the physical system are the following:

1. Determine the number of lines you need, and from this determine the number of LSPs.
2. Determine the number of NSPs.
3. Check the number of lines per NSP.
4. Check the bandpass per NSP.

These steps are explained in more detail in the following text.

1. Determine the number of lines and LSPs.

   First determine the number of lines you need and speed of each line. The number of lines an LSP can support is determined by the line speed.

LINE SPEED	NUMBER OF CONTINUOUS POLL LINES PER LSP	NUMBER OF POLL CONTENTION LINES PER LSP
19,200	1	2
9,600	4	12
4,800	8	16
2,400	16	16

   Poll contention means that the LSP waits until a terminal requests polling before it initiates the poll sequence.

2. Determine the number of NSPs (for non-DC-DLP systems).

   Assign up to eight LSPs to each NSP. See steps 3 and 4 for line and bandpass limitations.

3. Check the number of lines per NSP.

   The total number of lines an NSP can support is determined by the amount of memory in the NSP and whether or not the lines are multidrop lines.

NSP MEMORY SIZE	MAXIMUM MULTIDROP LINES	MAXIMUM LINES WITH 1 TERMINAL PER LINE
256K	24	48
512K	64	128

4. Check the bandpass per NSP.

   The NSP is unaffected by line speeds but is affected by the number of messages per second and the total number of characters per second in those messages. Estimate the maximum message traffic and the maximum message size and compare that to the NSP bandpass.

   NSP BANDPASS

   25 messages per second
   25,000 characters per second

When you use DC-DLPs, perform the following steps when configuring the physical system:

1. Determine the number of DC-DLPs needed from the number of lines and line speed.

2. Check the bandpass per DC-DLP.

These steps are explained in more detail in the following text.

1. Determine the number of lines needed and the speed of each line. The number of lines a DC-DLP can support is determined by the line speed.

LINE SPEED	NUMBER OF CONTINUOUS POLL OR POLL CONTENTION LINES PER DC-DLP
38,400	2
19,200	4
9,600	4
4,800	4
2,400	4

2. The DC-DLP is affected by the characters per second. Estimate the messages traffic in characters per second and compare that to the DC-DLP bandpass.

   DC-DLP BANDPASS

   3,000 characters per second

Once you have determined the configuration for datacomm, the next step is to use the Interactive Datacomm Configurator (IDC) to inform the system of the configuration. This requires the hardware addresses for the components and normally is taken care of by the field engineers who install the system. See the "Interactive Datacomm Configurator (IDC) User's Guide" for an explanation of the use of IDC.

Setting Up a Multilingual System

The MultiLingual System (MLS) makes it possible to display output messages, on-line help text, and menu screens in multiple natural languages, such as French, English, and Spanish. Each user on the system can view the messages in a different language.

To give computer sites multilingual capability, MLS uses three utilities--the Message Translation Utility, the Interactive Menugraph Generator (IMG), and the Help Utility.

The Message Translation Utility translates ALGOL output messages from one natural language to other natural languages. These translated output messages can then be added to the ALGOL program containing the original output messages.

IMG provides the tools to customize screens for Menu-Assisted Resource Control (MARC) and can be used to translate MARC screens into other languages.

The Help Utility processes on-line help text for several utilities and products.

MARC, although not a part of MLS, is a product that looks for a specified language in which to display MARC screens. MARC is the only product that integrates all the multilingual features of MLS so that the screens, on-line help text, and messages of MARC can be translated and retrieved in a specified language.

For instructions on how to set up a multilingual system, see "The System Manager's Guide to the Multilingual System" section of the "MultiLingual System (MLS) User's Guide."

4.3 PLANNING DAY-TO-DAY OPERATIONS

Day-to-day operations need as much attention and planning as do startup tasks. These operations include keeping records, monitoring system performance, managing jobs, maintaining files, and managing backups.

KEEPING RECORDS

A site should maintain records of system activity, performance, use, and maintenance. These records are necessary to distribute resources effectively, to evaluate system performance, to plan for efficient use, both present and future, to bill accounts, and to prevent downtime. Records should be kept of the following items:

1. Log files or extracts for maintenance records
2. CPU utilization, on-line response time, and transaction counts
3. I/O traffic, preferably by unit
4. Available disk space
5. Batch throughput
6. Development activity
7. Resource use by the major categories of users
8. Overhead, which is computing activity that produces no visible end result but supports the operational environment. Creating backup files is one example of overhead.
9. Equipment faults
10. Any stoppage, such as Halt/Loads, and the reasons for each
11. Usage time and downtime, with program rerun time recorded where meaningful
12. Trouble reports submitted, changes received and applied, any operational detours
13. Memory dumps

The system logs summary information about system activity in a protected disk file called SUMLOG. This file is created during a cold start and is maintained by the MCP, which categorizes information by both major and minor type and stores it in the appropriate area of the file.

In addition to the defined major types, a major type is reserved for definition and use by each installation. Operators or privileged users can enter a comment in the SUMLOG file to indicate the start or end of an event, such as a special test. The comment is entered through use of the LC (Log Comment) ODT command or the "LC" MARC menu selection.

The information in the SUMLOG file is used by the System Management Facility II (SMFII) to generate and maintain a statistical database on system performance. You generate reports from this database through the SMFII QUERY program. You can also use the LOGGER and LOGANALYZER utilities to view SUMLOG file information and to generate reports on selected data. See "System Performance Monitoring" in the section "A User's View of System Functions" for information on these programs and other performance measuring tools. The "System Software Support Reference Manual" contains detailed information about SUMLOG.

The system automatically starts a new SUMLOG file when the old one is 95 percent filled. On a busy system, this can take from three to six 24-hour days.

NOTE

Be aware that the old log files are not removed by the system and continue to accumulate unless specifically removed by an operator.

Therefore, removal of log files is an issue that should be planned before your system arrives.

You may want to set up an archiving system for log files and remove the files on a periodic basis. The archiving system can be as complex as necessary. However, a simple method is to enter information about the files in a log and remove them daily, using a set of tapes, numbered 1 to 10, that are rotated. The daily log entry contains the tape number, the date, and any unusual events that occurred during the day. You may also want to print a daily copy of the log, which can done through the LOGANALYZER utility. A new log file can be started manually through the LR (Log Release) or TL (Transfer Log) ODT commands.

You can use two system procedures as models for any special log analysis programs your site may need: SYSTEM/JOBFORMATTER and SYSTEM/LOGANALYZER. Both programs contain log format defines that can be used by an ALGOL or DCALGOL program. However, it is possible to report on most log items using SMFII.

MONITORING SYSTEM PERFORMANCE

Monitoring performance gives you the information you need to determine how your system is actually used, how it is performing, and where changes, if necessary, can be made. "System Performance Monitoring" in the section "A User's View of System Functions" explains the tools available for monitoring the performance of the system.

It is strongly recommended that system monitoring be performed on a regular, periodic basis, with a minimum of several times a week but preferably daily. Start by monitoring daily, then gradually increase the time between monitorings as system use becomes more efficient and balanced. However, any time you change the status of the system, for example, by adding new application programs or changing to a new MCP version, you should increase the frequency of monitoring.

MANAGING JOBS

Managing the flow of jobs is one of the most important aspects of operating a computer system and has a large effect on efficient use of the system. If the flow of jobs on your system is predictable and can follow a regular timetable, scheduling jobs to gain optimal performance from your system is relatively easy. But if the flow of work is random and fairly unpredictable, as it typically is for program development, you must do more planning to prevent overloading or underutilizing your system.

If you have not read it already, read the discussion of "Job and Task Management" in the section "A User's View of System Functions" to gain an overview of job management. Managing the flow of jobs on the system is accomplished through the Work Flow Management system (optional) and through the use of job queues.

Job queues are a practical way to make large numbers of jobs visible for selection by the MCP. You can define multiple job queues for different classes of service and improve turnaround time for short jobs by grouping them in the same queue. A job queue has attributes that describe the maximum resources that a job from that queue can use: priority, I/O time, process time, subspaces, tapes allowed, lines printed, wait limit for an event, disk space limit, and maximum active time.

The MCP routes jobs to the appropriate job queues. A job can specify a particular queue, and the MCP will put it into that queue. When a job specifies resource restrictions, the MCP puts it into the highest-numbered queue possible. When no restrictions are specified, the MCP puts it into the default job queue and assigns it the default restrictions for that queue. If a job does not fit the requirements of any job queue, the system terminates the job.

Jobs are selected from job queues on the basis of the priority and the mix limit of the queue. The mix limit is the number of jobs from that queue that can be executing at any given time. The system initially provides one job queue, queue 0, which has the lowest priority. You can modify or delete this queue and add other queues. Note, however, that no jobs can be started if there are no job queues.

For random flow, queues are one of the best ways to prevent overload and allocate resources fairly. By using queues effectively, you can group jobs by the demands they make on system resources and control resource allocation. Read "Setting Up Job Queues" earlier in this section for information on creating queues.

The efficient flow of printing jobs is handled through PrintS. Read "The Print System" in the section "A User's View of System Functions" for information on controlling printing jobs.

In addition to these methods of handling jobs, you can schedule certain jobs or types of jobs for specific times of the day or week. This allows you to group jobs that use resources requiring set-up time. One way to schedule these jobs is to create job queues specifically for them and give these queues a mix limit of 0. When you are ready to run the jobs, increase the mix limit to 1 or greater. For example, for nighttime runs, the mix limit could be 0 during the day and 3 at night.

MAINTAINING DISK FILES

Day to day maintenance of disk files involves managing file storage on disk. You manage storage by consolidating files, removing permanent files no longer needed, and redistributing files to improve system performance.

Consolidating files is necessary when available disk storage space has become fragmented into numerous small areas. This condition, known as "checkerboarding," often results from constant creation and removal of files, especially small ones, on a disk. The total number of available sectors may be large, but because the system cannot find free space in contiguous sectors, it may not be able to allocate an area for a file.

You can consolidate files through the SQUASH (Consolidate Disk Allocation) ODT command. The squash operation moves file storage areas, particularly shorter ones, to available areas. It reduces fragmentation but does not completely eliminate it. However, successive squash operations can significantly diminish fragmentation. You may want to plan to "squash" disks on a regular basis.

Permanent files are often created as a byproduct of programs and remain on disk after the need for them has expired. Because these files decrease available disk space, you may want to create a program that searches for and removes the files, and then run the program on a regular basis. You can incorporate a call to the FILECOPY utility in your program to do the actual search and removal. See the "System Software Utilities Reference Manual" for information on this utility.

Any damaged areas on disk can be removed from the systems table of available areas through use of the RES (Reserve) ODT command. You can use this command to copy data or code from the damaged area even if errors occur during the copy operation. This allows recovery of as much information as possible.

The FILEDATA utility produces a variety of reports that you can use to help you plan and implement file maintenance. The reports can give you a hierarchical list of files, a map of file storage layout, a "checkerboard" report showing file locations and the space around them, the attributes of files, catalog information about files, and a list of disk files on a library maintenance tape. In addition, FILEDATA can generate a file that can be used for periodic file maintenance. The "System Software Utilities Reference Manual" explains how to use FILEDATA.

MANAGING BACKUPS

Files require some form of backup to guard against loss for any reason. At one extreme, a backup consists of a direct audit of every change to a file. The other extreme is just to make an occasional safety copy of the file.

A tradeoff must be made between the cost of backing up a file and the cost of recovering the information via audit or other means. Usually only updated files are backed up on a daily basis. On a weekly or monthly basis, categories of files are backed up to serve as a base for recovery. Copies of these weekly or monthly backups often are stored in secure locations away from the site to prevent loss from fire or other accidents.

The management of backups is a process unique to each site. The procedure depends on the types of files, how often they are changed, and how critical the information is in each file. A site should plan in advance how they are going to keep track of backup copies, where the copies are to be stored, who is to do the backup operations, and when. Backups should be scheduled during periods of light system use. As an aid to the backup process, A Series systems can be run with Cataloging. See "File Management" in the section "A User's View of System Functions" for a summary of Cataloging, and see the "Disk Subsystem Software Overview" manual for a detailed explanation of it.

Source code files MUST BE RECOMPILED at least every third major release. This is necessary to allow the compiler/MCP interface to evolve and change with each successive release. Therefore, your site should make provisions to back up application program source code files and store them in a secure place.

Other files that should be backed up are data files, log files (optional), changed application program files, and changeable system files, such as the USERDATAFILE, DATACOMINFO file, and the source for the configuration file.

4.4 PLANNING FOR EXCEPTIONAL OPERATIONS

Exceptional operations are those that occur infrequently, often on an unscheduled basis. You can and should plan for their occurrence so that they can be handled as quickly as possible and with minimal impact on system operation.

MEMORY DUMPS

If the MCP detects a serious problem, it tries to dump an image of the contents of memory to either disk or tape. A memory dump can be fatal or nonfatal. A fatal dump results in a Halt/Load, while a nonfatal dump does not.

The MCP automatically initiates the memory dump without any operator intervention, dumping to the disk file designated through the DN (Dump Name) ODT command. If no file is designated, it dumps to tape. If the dump went to disk, the MCP attempts to free the disk dump file for another dump as quickly as possible by renaming the filled file or dumping the file contents to a tape file. The MCP procedure DUMPDISKMASTER handles this procedure and issues an RSVP message to the operator, who responds with instructions on file disposition. After the dump, the MCP resumes normal operations.

The contents of the dump file are translated into a readable format by the DUMPANALYZER utility. The report from the utility is a diagnostic tool that can be used for on-site analysis or sent to Burroughs with the Field Communication Form (FCF) you use to report the problem. Field Communication Forms are explained in the section "If You Have a Problem." If you send the dump to Burroughs for analysis, you first must run DUMPANALYZER on the file, because the utility must use the MCP code file that was running when the dump occurred to correctly convert the file contents.

You should keep a record of all memory dumps, recording when and why they occurred. In addition, you may want to keep the dump files, in which case you might add them to your file backup system. One method of storing dump files is to use a log book to record all dumps and a set of numbered tapes to store the dumps for a preset length of time. The tapes are rotated and are identified by a simple numbering system.

A site can control whether or not the system takes memory dumps through use of the system run-time option NODUMP, set through the OP (Options) ODT command. The site also specifies the name of the disk file to hold the dump through the DN command.

You can stop a memory dump while it is in progress through the DS (Discontinue) ODT command and can force a memory dump through the DUMP (Dump Memory) ODT command or the "??MEMDP" primitive command. See the DN, DF (Empty Dumpdisk File), and the CM (Change MCP) ODT commands in the "Operator Display Terminal (ODT) Reference Manual" for more information on handling memory dumps.

RECONFIGURING THE SYSTEM

If your system has any redundancy in the I/O or datacomm components, you need to plan alternative configurations of these components to cope with any abnormal situations, such as downtime for equipment maintenance or malfunction. In addition, if your system shares I/O components with another system, you should plan the different configurations possible with and without each component.

You can place these alternative configurations in the configuration file ahead of time and reconfigure the system through software when necessary. See "Soft Reconfiguration" in the section "A User's View of System Functions" for information on creating configurations and placing them in the configuration file.

UPDATING THE SOFTWARE

On a periodic basis, you will receive updated versions of system software. You need to plan ahead of time how you are going to integrate this software into your system and start using it. You may want to run the new software at night or on weekends until it is fully integrated.

For Mark releases, the software arrives with the "System Software Installation Guide" and the "Software Information" documents. The "System Software Installation Guide" describes the software and the major features of the system, and explains how to install the system software. "Software Information " contains warnings and precautions concerning the system software.

For support releases, the software arrives with the "Support Release Document." This document describes the software, any enhancements or changes to the old version, and precautions to take when installing the new release. The document also includes packaging, hardware, firmware, and documentation information. It is, therefore, very important that you read these documents thoroughly before you start integrating the new software into your system.

Of special note is the hardware level information. In order for the new version of the system software to run correctly, the hardware must be at the correct release level. Check to be sure your site has implemented all hardware engineering changes that have been received to date.

PTD CONFIDENCE AND DIAGNOSTIC TESTING

The Peripheral Test Driver (PTD) is an MCP procedure used to run confidence and diagnostic tests on Data Link Processors (DLPs), peripheral devices, and bad disk files. PTD uses a series of code files containing operation codes, one file for each type of DLP and peripheral that Burroughs supplies. Within each code file are numerous individual tests, each of which performs an operation appropriate for that device. When you use PTD, you can specify the tests you want performed as well as which device to test.

PTD is intended for use by Burroughs Field Engineers, but you might use PTD under the following circumstances:

- o If you buy a disk pack from a supplier other than Burroughs. These packs must be prepared for use by running the Initialize, Verify, and Relocate (IVR) program through PTD.
- o For confidence testing. Confidence tests are run to verify correct operation. You typically would run these tests after maintenance or repairs are performed on a device or as a regular part of a preventive maintenance program. Some of the tests can be performed while the device is in use.
- o For diagnostic testing. Diagnostic tests are run to detect a problem and to find the cause. You would run these tests when an intermittent problem (such as a parity error on a tape read operation) occurs, and you need to ascertain if the problem is in the media or in the device, and just what the problem is.

The PTD tests are available on a separate tape, PTDTESTS, that also includes the user's guide. The tape is available from your local Field Engineer. In addition, there is a supplemental document for A Series, B 5900, and B 6900 systems, DOCUMENT/PTD, on the DOCUMENTS tape that is a part of the system software release package.

5 PROGRAMMING ON A SERIES SYSTEMS

This section discusses topics that programmers should be aware of and keep in mind when writing programs to run on A Series systems. It is directed to both experienced and inexperienced programmers and explains issues that influence the efficiency of programs running on A Series systems. The discussion assumes that the reader already knows how to program and therefore only needs to learn how to program on A Series systems.

5.1 PROGRESSION AND CONVERSION

Progression and conversion involves defining the differences, if any, between the programs and their operational environments on the old and the new systems, running programs through filter programs to translate language differences, moving programs to the new system, and refining the programs to achieve greater efficiency.

AVAILABLE TOOLS

When you progress from a Burroughs B 1000 or B 2000/B 3000/B 4000 Series system to an A Series system, a range of services are available to you: requirements definition, education courses, software tools, and programming and system consultation. You can choose some of these services or contract for the total effort.

If your site decides to handle most of the effort, you can order progression software aids to help you save time and minimize costs. See the "User Guide to B 1000 Progression Aids," obtainable through your Burroughs representative, for the software aids available and information on using them. These aids are for both small and medium systems and include file transfer utilities for transferring tapes between systems.

In addition, you should order the "B 1000 Series to A Series Progression Guide." This guide discusses the differences between small and A Series systems, lists the steps in progression, describes common progression problems and solutions, tells you how to transfer files, and describes the steps to take after using the software progression tools.

For conversion from non-Burroughs systems, see your Burroughs representative for the available conversion aids.

CONVERSION OF PROGRAMS

There are several tasks you can start well ahead of time, before you receive any conversion documents: locate the source code for your application programs, and identify and flag any machine-dependent or nonstandard code in the programs. This code must be changed either before or after you run the program through the translation tools. On future changes to programs, make it a practice to stay as close as possible to the standard language. Anything nonstandard can be placed into a library or a separate module.

For existing programs, transfer the source code only, make any needed changes to the source, then recompile the program. This ensures that the object code will run on the current MCP.

Some routines, especially small, frequently used ones, can be placed in general or specialized libraries for use by all programs on the system. This can reduce the size of existing and future application programs and thereby reduce required disk storage space.

If at all possible, you should plan to run the old and new systems in parallel for a short time and compare program results. This can help ensure that all programs are performing as required.

If some of the programs you are converting are old and have been modified numerous times, this may be a good time to rewrite them. The result can be easier maintenance and greater efficiency when running the program. However, unless the rewrite is well managed, it can add risk to the conversion schedule. The tradeoffs should be carefully considered.

CONVERSION OF DATA

When you convert or progress your application software to A Series systems, pay special attention to the format of the data. The format affects the amount of overhead required to handle the data on the system. For example, on numeric string data transferred from non-Burroughs systems, if the sign field is not properly positioned, more operations may be required to interpret it. (See the reference manual for the programming language you are using for information on placement of the sign field.)

There are three levels of data to consider:

1. The physical level, which is the actual representation of the data and includes character sets and character sizes. At this level, you need to be sure that the system recognizes the format used.

2. The logical format of the file. The file format on your old system may differ from that on your new system. If so, you may need to write an extraction program that removes the file from the tape record by record and reformats it as it writes it to disk.

3. The format of the transfer tape. It is possible for a file to be successfully loaded onto an A Series system yet not be usable by that system. This occurs because the file information can be read but not interpreted by the system. To avoid this problem on transfers between Burroughs systems, use the file transfer utility programs supplied by Burroughs. (See the section "Transferring Files" in the "B 1000 Series to A Series Progression Guide" for specific information on the utility programs.) For transfers from non-Burroughs systems, use a copy program such as DUMPALL that does not copy the file header with the file. DUMPALL creates a new header for the file as it is placed on disk.

5.2 PROGRAM INITIALIZATION, RECOVERY, AND RESTART

Because the mechanisms for starting and restarting programs are very flexible, you need tc plan which of the available methods you are going to use. In addition to the Work Flow Language (WFL), the startup mechanisms also include supervisor programs that can start other programs after a Halt/Load.

APPLICATION PROGRAM INITIALIZATION

The execution of batch application programs on an A Series system is normally controlled through the Work Flow Management system. Through WFL statements issued to start the job, you can

- o Control multiple programs running asynchronously
- o Define and change the parameters of the program or programs that make up the job

This allows you to use an application as a generic program and customize it to fit changing conditions when it is run.

There are three kinds of WFL specifications you can use to affect the program: task attributes, file equations, and data specifications. Task attributes have diverse functions, such as specifying the priority of the program, the usercode it is to run under, and in what memory subsystem it is to run. File equations are used to substitute different files for the files named in the program, such as input and output files. Data specifications supply input to those programs that expect input from a card reader during execution (that is, programs that declare a file with the file attribute KIND equal to READER).

For more information on the use of WFL specifications for program initialization, see the "Work Flow Language (WFL) Reference Manual."

RECOVERY AND RESTART

An important consideration when you are writing an application program is how the program is going to recover and restart if it is interrupted. The kind of recovery needed depends on the importance of the transactions it is performing and whether it is answering inquiries or updating files.

Interactive Programs

For interactive programs, the type of recovery you build into your program depends on the type of transaction it is processing.

For simple inquiries, sending a message to the user may be sufficient. This message could simply tell the user to repeat the last inquiry.

For straight data entry programs, you can keep an audit trail or tell the users what the last recorded transaction was and direct them to re-enter all transactions since that one.

For programs that randomly update files, you need audit trails containing before and after records of each update. However, if you are using DMSII, you can have it audit the transactions, sparing you the necessity of writing the extra audit code. Recovery from interruptions is automatic, so that re-entry of data is unnecessary.

The "Communications Management System (COMS) Programmer's Guide" contains information on synchronizing programs with DMSII.

Batch Programs and Jobs

The system automatically restarts batch jobs if they are interrupted. However, through Work Flow Language (WFL) statements, you can control how and where in the program they restart.

Some application languages permit stops in a program, called checkpoints, for the system to save information regarding the state of the program at that point. This information consists of everything the system needs to know to resume execution of the program. If the program is interrupted and must be restarted, the system can restart it at the last checkpoint instead of at the beginning. WFL provides the RERUN statement to direct the system to restart the program at a particular checkpoint.

SUPERVISOR PROGRAMS

A supervisor program is a specially designated code file that is run immediately after a Halt/Load. The system automatically enters it in the job mix and runs it before all other jobs. You can use a supervisor program to automatically perform chores that an operator would otherwise have to do when starting up the system, such as setting up special job queues. However, you are not limited to this type of use and can have the supervisor program perform any functions wanted and needed by your site.

A supervisor program can send Operator Display Terminal (ODT) commands to the MCP and receive the corresponding responses through the DCALGOL "DCKEYIN" statement. Any program using this statement must be privileged, meaning the program must be run under a privileged usercode or be designated a privileged program.

You designate a program as a supervisor program through the CS (Change Supervisor) ODT command. See the "Operator Display Terminal (ODT) Reference Manual" for the format of the command.

5.3 ONGOING CONSIDERATIONS

DESIGN OF PROGRAMS

With the availability of program generators such as the Logic and Information Compiler (LINC), the most important aspect of program design has become the evaluation of the relationships between the data. Good program design requires that thought be given to the effect of file, data, and program structure on the efficiency, maintainability, and integrity of the program.

File and Data Structure

File and data structure is concerned with record, block, and area sizes. The sizes of these items affect the speed and performance of programs and of the system, and the efficient use of storage.

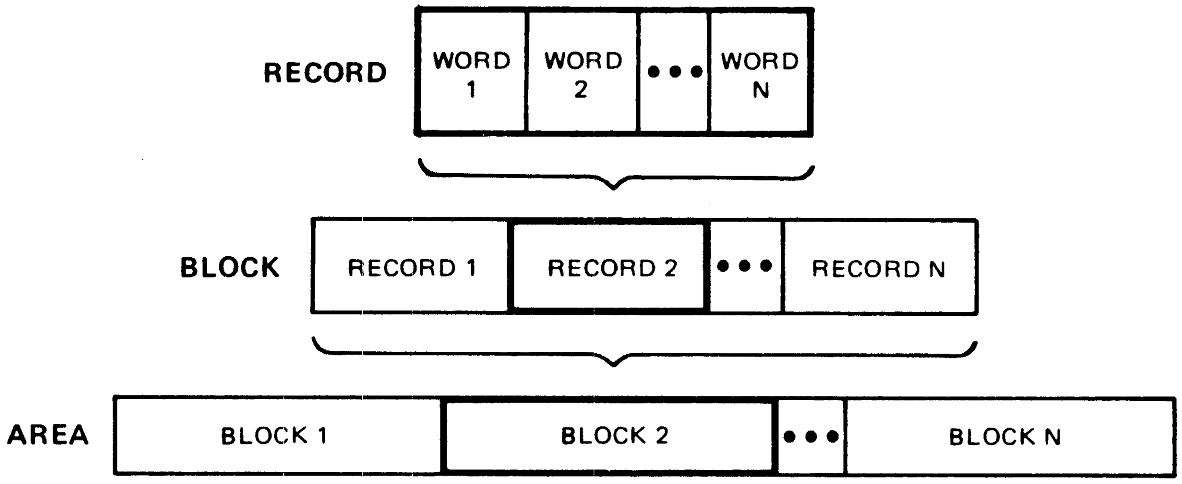
For the greatest central processor efficiency, the record size should be a multiple of the word size, six bytes, because some of the operations are more efficient on words than characters. A useful technique is to make the record size larger than initially needed, to allow for future expansion. If you have a choice, fixed-length records are normally more efficient than variable-length records, because the system can more easily manage data storage and access.

The system transfers records to disk in blocks made up of a number of records. However, data is stored on disk in segments of 180 bytes, the size of one sector. Therefore, for minimum disk space waste, the block size should be a multiple of both 180 bytes and of the record size. For example, if the record size is 90 bytes, the lowest block size would be 180 bytes, or if the record size is 150 bytes, the lowest block size would be 900 bytes.

When a file is read, the system transfers a block from disk into buffers in memory. From there, the MCP gives the program a particular record. If the next record is in the buffer, transfer time is saved. If not, another block must be transferred. Therefore, for serially accessed files, where records are used in the order in which they are stored, the block size should be large in order to minimize the number of transfers. For randomly accessed files, where the next record needed probably is not in the buffer, the block size should be small to minimize the amount of time spent in transferring a block.

For files accessed both sequentially and randomly by different programs, choose a record size such that a small multiple of it is a multiple of 180. Then use a small block size for random access and a large block size for sequential use. For example, a record size of 270 bytes allows a small block size of 540 bytes and a large block size of 10,800 bytes.

Just as records are accumulated into blocks in memory, so blocks are accumulated into areas on disk. A file can "own" and occupy multiple areas. The file attribute AREALENGTH defines the size of each area, and the attribute AREAS defines the number of areas the program needs. The following diagram shows the relationship between records, blocks, and areas. To simplify the illustration, a smaller than normal record size is used.



It can be helpful to a site to standardize the size of all areas to 504 sectors, the area size of object files produced by the compilers. This can help prevent "checkerboarding" of the disk, where available disk space is fragmented into areas too small for the file areas.

Areas are allocated only when needed. If they are too small, the MCP wastes time keeping track of a large number of areas. If they are too large, space may be wasted, and it will be difficult to find space of the required size.

When a file reaches the largest size anticipated for it, it is a good idea to use the crunch option when the file is made permanent (entered into the disk directory). This option causes unused blocks in the last area of the file to be returned to the available pool for the disk.

If you are writing programs for database applications, there are several sources of information available. See the "Structure Formats and Pragmatics" and the "Designing DMSII Databases" appendixes in the "DMSII DASDL Reference Manual," and see the "Efficiency Considerations" appendix in the "DMSII User Language Interface Software Operation Guide."

The Effect of Program Structure

The program you write to fulfill an application need can be structured as one large or many small programs. A single large program centralizes the application and minimizes the overhead incurred when starting and ending programs, but maintenance of the program may be difficult. With several small programs, each can execute quickly, and faults in one can be isolated from the other programs. Small programs also make it easier to balance the load on the system.

When writing a large program, you have a choice between writing it as one large program or writing it as several smaller programs and binding them into one program. You can also choose to use libraries in lieu of binding some of the small programs.

The advantage of using libraries or binding programs is that complex data-handling routines needed by many programs can be written only once yet used by all programs. The system allows routines written in one language to be used by programs in another language, with some limitations. These routines can be bound to the program or called through library calls during execution. You write a routine in the language most suited to the expression of the function it performs; for example, COBOL for printing reports or FORTRAN for processing numbers.

Whether to use libraries or to bind routines depends on the type of routine and how it is used. Libraries are good for both single- and multifunction procedures, such as currency conversion routines. It is slightly more efficient to bind a routine than to place a library call to it, but if you change the routine, you must change many different programs rather than one library component. In programs containing bound routines, the symbol and object files don't match, and you cannot use the Test and Debug System (TADS) to test the program. However, bound routines can share global data, which must be passed to libraries as parameters.

TADS can be used to test libraries. If a task that calls the library also runs with TADS, then that task must have its own remote station or COMS window for the test session.

For more information on libraries and on binding, see the "Libraries" section of the "System Software Utilities Reference Manual" and see the "Binder Reference Manual."

General Practice

Initialization time for complex databases can be as long as 30 to 40 seconds. When you have a complex database that is used only a few times daily, but the access to it is time critical, consider writing a program to open the database at the start of each processing day. This program would open the database and keep it open by waiting for a user-defined event that never occurs, thus eliminating the initialization for each user.

Small, frequently-used programs, such as a Pascal compiler in a university lab, can be kept resident in memory. This decreases processor and I/O use when the program is run. A program is made resident through the RP (Resident Program) ODT command.

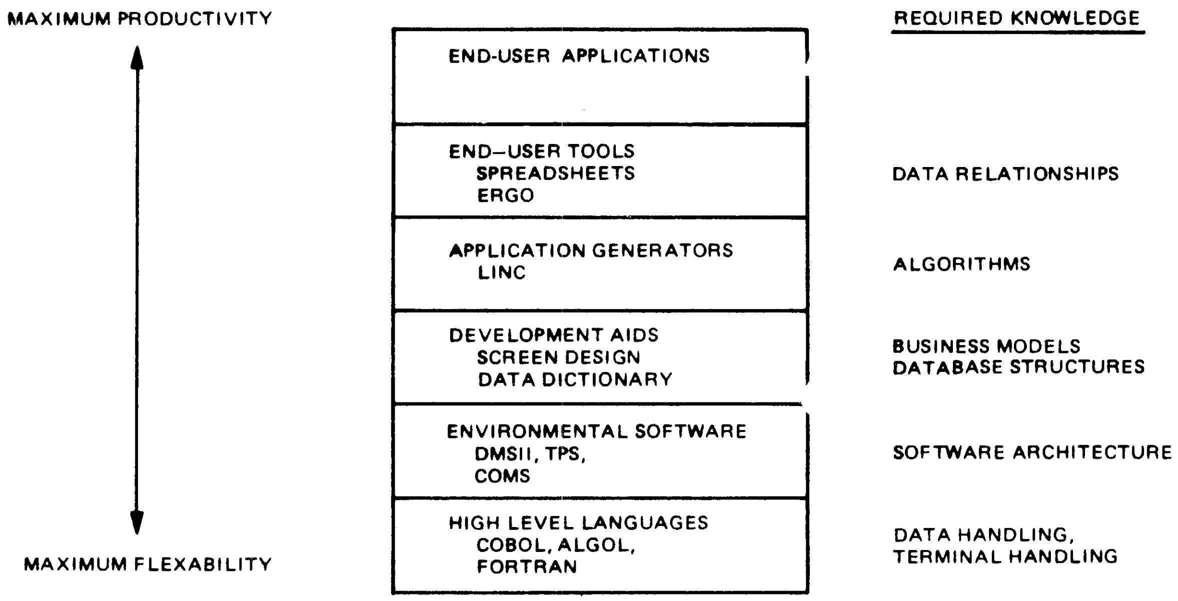
Program Maintenance

Maintenance of programs is an ongoing function that must be planned for ahead of time and designed into a program. Maintenance refers to the use of diagnostic, checking, or test routines to help programmers remove mistakes and errors from a program and produce correct and efficient code.

You can have two versions of programs, a diagnostic version and a regular version, and predefine to the operators the conditions under which each one should be used. Also define what output should be saved, such as job summaries, to give a history of the program execution. The system log files are an excellent source of information, especially for tracking what was happening on the system when the job was running. In addition, you can print the contents of the COMS and DMSII audit files if applicable.

PROGRAM DEVELOPMENT

Burroughs provides a variety of software tools to aid in the development of application programs. These tools are designed to increase productivity and be easy to use. Within these tools, there is a difference based on the amount of work required to produce the finished product and the amount of flexibility the tool allows the programmer. The following diagram shows this tradeoff between flexibility and productivity.



As programmer productivity increases, the knowledg[illegible]quired to use the tools changes. The user is required to have less knowledge of the hardware and software architecture of the system and can concentrate on knowledge of the problem the application is solving. However, there is a decrease in flexibility; the routes available to solve the problem become increasingly restricted.

As an A Series system user, you can choose the level of tools that meet your site's particular needs and objectives. See "Productivity Software" in the section "Virtual Memory, Stacks, and Other System Concepts" for a list of the products available, and see the section "A User's View of System Functions" for a more detailed explanation of the individual products.

APPLICATION PROGRAM MANAGEMENT

For new application programs, major considerations are which language to use, which programs to write in-house, and which programs to purchase. The type of application or the language preferred by the site usually dictates the language used for a program. If you are purchasing general application programs, be sure that the version of the language used is compatible with A Series systems compilers, so that you can maintain the programs. See "Programming Languages" in the section "A User's View of System Functions" for the language compiler versions available.

Programs used frequently by many types of users or for different purposes can be made "generic" by use of Work Flow Language (WFL) file equations. For example, output file names in a program can be represented by a variable that is equated to an actual file name when the program is run. This type of file equation allows a program to serve many users without being changed constantly.

Code File Compatibility

Object code files generated on B 5900 and B 6900 systems are compatible with A Series systems and can execute on them. The B 5900 and B 6900 systems are part of the processor family of machines called Level 0 machines. (In this context, "family" is distinct from a machine series such as B 5000/B 6000/B 7000.) The A Series systems are in the Level 1 family of machines.

The terms "Level 0" and "Level 1" are values for the compiler control option TARGET. Compilers can generate code that is optimized for a specific family of machines and that runs only on that family or a limited set of families. You gain performance improvements when you compile programs with TARGET set for the family on which the program is to run. Therefore, although Level 0 programs execute on Level 1 machines, you may obtain some performance improvements if you recompile the Level 0 programs with TARGET equal to Level 1 when you want to run them on Level 1 machines only. However, the converse is not necessarily true, and code files generated for Level 1 families will not execute at all if run on Level 0 machines.

If you do not specify a value for TARGET, the compiler uses the default specification maintained by the MCP. The system default specification is to compile code for the machine on which the compiler is running, but you can select the MCP default specification through the COMPILERTARGET (Set Default TARGET Value) ODT command.

Distribution of Files

You can influence backup speed for application programs and related data files by the location of these files on disk families. The basic principle is to group files that change frequently and separate them from files that change only occasionally. Depending on the number of disk packs and families at your site, place frequently changed files on a separate family or at least a separate pack. Avoid placing them on families that contain system files, because this can complicate or lengthen the backup process as well as slow execution of programs. In addition, keep overlay packs as clear as possible of application programs. If these packs fill up, the system can slow down drastically or halt.

CONTROLLING PROGRAM CHANGES

You may want to set up procedures to control how changes are made to application programs, especially critical programs. On A Series systems, you have the ability to develop, test, and review changes to a program before actually changing the program. This ability is a part of the process of patching, which is a means of making controlled changes to a program. Patching has the advantage of allowing you to keep a record of changes made to a file.

There are three ways to handle patching:

1. Working in patch mode in the Editor

2. Using the SYSTEM/PATCH utility program

3. Using the "mark" field in CANDE text files

The Editor normally runs in merge mode, meaning that changes made to the file during the Editor session are merged into the source file when the session is ended. In patch mode, you specify two files for an editing session: a patch file and a source file. The source file is the file you want to change, and the patch file is a work file that contains the source file records that are actually changed during the session. During a session in patch mode, you edit a copy of the source file. At the end of the session, the Editor saves the changed records in the patch file, and the source file remains intact.

You can merge the patch and source files to produce a temporary file containing a copy of the source file with the patch applied. This temporary file then can be used for review and testing until you are satisfied with the changes. Merging is accomplished in one of two ways: through the SYSTEM/PATCH utility and through compilers.

The SYSTEM/PATCH utility has the ability to combine multiple patch files into a single file and merge this file with the source file. It also can produce a listing comparing the changed records to the original records. By inspecting this listing, you can see how your patch affects the source file and if there are any conflicts when multiple patches are applied. Once you are satisfied, you can run the utility again and make the patch permanent. See the "PATCH" section of the "System Software Utilities Reference Manual" for complete information on SYSTEM/PATCH.

Compilers also provide a mechanism to apply patches to software and test the resulting code files. Through compiler control options and file equation statements, the compiler can be directed to merge the source and patch files and compile the resultant file.

The mark field of files created through CANDE can be used to track changes to a file. In a file containing 90-character records, the last 10 characters are the mark field. This field can be used to indicate anything you want, such as the date the record was placed in the source file. You can have the Editor automatically change the field through use of the MARK command, or you can manually change the mark field through the Editor CHANGE command.

TESTING

When testing programs that are potentially harmful to the running of the system, such as privileged programs, it may be a good idea to test them under a nonprivileged usercode. If necessary, you can create temporary test files to replace privileged files or functions.

When testing programs that interact with databases, a safe way to proceed is to model the database and use that for testing. You set up another database that is logically identical to the database involved in the test but is physically smaller. The smaller size helps to shorten testing time. See the "Update, Reorganization, and Modeling" section of the "DMSII DASDL Reference Manual" for information on modeling databases.

The same idea can be used for testing other large programs. Use a logical subset of the program, a skeletal version, that contains all the essential functions but not necessarily all the possible data paths. Then add other modules or paths as testing proceeds. Also use copies of production data when running test programs.

When testing ALGOL, COBOL74, or FORTRAN77 programs, use the Test And Debug System (TADS), an interactive tool for testing programs and libraries. Through TADS you can monitor and control the execution of the software and examine or modify the data at any given point during program execution. See the "ALGOL Test And Debug System (TADS) User's Guide," "COBOL74 Test and Debug System (TADS) User's Guide," and "FORTRAN77 Test and Debug System (TADS) User's Guide" for instructions on the use of TADS.

5.4 EXCEPTIONAL SITUATIONS

Exceptional situations are situations that you need to plan for, but that occur infrequently. These include new releases of software, progression, adding new features and deimplementing old ones, and analyzing memory dumps.

NEW RELEASES

On a periodic basis, you will receive updated versions of system software. You need to plan ahead of time how you are going to integrate these programs into your system and start using them.

If the changes to the software you are using are relatively minor or if the software affects only a minor function, you can probably start running it as soon as you load or need it. If the change is substantial or is to software critical to system performance, you may want to run the software after your normal work day or on weekends to ensure that there is no incompatibility with your application programs.

If you have sent a Field Communication Form (FCF) to Burroughs regarding a software problem, and the problem is one that a software patch has been written for, Burroughs will send you the patch and a letter of explanation. The patch usually is to the source code for the affected program and must be compiled into your software. It is a good idea to monitor the patched system software to see if the patch solves the problem. Be sure to report the results to your Burroughs representative, because the results are of interest to many users.

For new releases of the system software, the first thing you should do is read the "System Software Installation Guide" and the "Software Information" documents that come with Mark releases, and the "Support Release Document" that arrives with the support releases. These documents describe the major features of the system software, any enhancements or changes to the old version, and precautions to take when installing the new release.

There are four steps involved in installing a new release of system software on a system. These steps assume that the system is operational and is running on a prior release.

1. Load and initialize the new MCP and the new system libraries.

   NOTE

   It is very important that the MCP and the system libraries be the same release level. This ensures that the new features of the release can run.

2. Verify system operation by running test programs or the current versions of your programs. You may want to use temporary copies of your data as a precaution.

3. Bring up the new compilers.

4. Recompile the programs that need recompilation and verify that the new versions run successfully.

Source code files MUST BE RECOMPILED every third major release. This is necessary to allow the compiler/MCP interface to evolve and change with each successive release. Therefore, you should make provisions to save source code files and set up a schedule for recompilation with each major release. As a bonus, you may find that recompiling programs sooner than is necessary results in additional speed in program execution because of features in the new software or because of compiler improvements.

DEIMPLEMENTATION OF OLD FEATURES

Occasionally old functions and commands are deimplemented by Burroughs. These are planned well in advance, and warnings are given to users in several ways.

1. A section of the "Mark 3.6.0 System Software Installation Guide" accompanying system software releases lists features deimplemented on the present release and features to be deimplemented on future releases. When possible, warnings are given at least two major releases before the actual deimplementation; for example, on Mark 3.6 for Mark 3.8 deimplementations.

2. Compilers give warning messages when they encounter features scheduled for deimplementation.

3. The MCP gives run-time warnings when a program contains features scheduled for deimplementation.

The MCP records any run-time deimplementation warnings in the system log file or the code file, or both if applicable. The system administrator can retrieve all deimplementation warnings by running the LOGANALYZER utility with the DEIMPLEMENTATION option. Individual users can view the list of warnings recorded for a particular file by using the FILEDATA utility with the WARNINGS file attribute or by using the CANDE "LFILES" command.

6 IF YOU HAVE A PROBLEM

Burroughs performs testing of hardware and software products to keep those products at the highest level of reliability. Typically, more than half the total time, effort, and cost of producing a software or hardware product is spent testing and debugging. However, you may occasionally encounter a failure and it is essential that you communicate the problem and the circumstances leading to it to Burroughs.

There also may be times when the system does the unexpected. You might see a repeated, momentary interruption, a sudden slowdown of response, or even a memory dump. If any of these or other unexpected events do occur, there are troubleshooting procedures to follow to find the cause. If, after following these procedures, you cannot find the cause and need to call for help, Burroughs support people are available for assistance.

Burroughs offers different levels of support service, allowing you to select the level appropriate to the availability and expertise of your staff. See the "Customer Guide to Burroughs Services and Support" for detailed information on the available services.

This section discusses how to proceed when the unexpected happens, where to get help, and how to report problems. Even if you yourself would rather not get involved in troubleshooting, you can speed up the process by knowing what information to have ready and what preliminary things to check before calling for help.

The document "A Customer Guide to Software Reporting" contains a more detailed explanation of the topics discussed in this section. See the section "Where to Find More Information" for information on ordering this guide.

DEFINING THE PROBLEM

Communicating information about problems may sound simple, but it can be a difficult task no matter how experienced at it you are. Knowing how to define the problem can ease the process and allow Burroughs to help you quickly and efficiently.

Ideally, to define the problem, you isolate, reproduce, and identify it. If this is not possible, your goal becomes to identify the essential characteristics of the problem. In either case, it is important that your definition be as specific as possible.

While it is tempting to take short cuts in defining the problem, such as using a random approach, experience has shown that a very structured approach works best. This approach can be broken into three phases:

1. Make quick checks.
2. Ask basic questions.
3. Break the problem down.

Make Quick Checks

The purpose of the quick checks is to determine if the problem is being caused by hardware or by software. This is necessary because in today's integrated computer environment, what appears to be a software problem may in fact be a hardware or operational problem. Some of the checks may seem obvious, but if you are pressured for time, you might overlook the obvious.

Quick Hardware Checks

1. Have there been any hardware changes or additions recently?
2. Are all interface cables installed and secure?

Quick Operational Checks

1. Do all units show power on and ready status?
2. Are all peripheral switches in the proper positions for the job/application?
3. Is all necessary software installed?

4. Are all configuration files resident and do they contain correct values?

5. Are all required files resident?

6. Are file, path, volume, and family names correct?

7. Are there sufficient resources to run the job?

8. Is the security level correct?

9. Have operational procedures changed recently?

10. Have new operators been using the system?

Checking these items often solves the problem. However, if this check does not reveal any causes, proceed to phase two.

<u>Ask Basic Questions</u>

At this point, you already should have ruled out hardware and operational problems as the cause of the problem. Phase two involves asking basic questions about what has changed. This can lead directly to the cause of a problem, because problems often appear when change occurs. Ask the following basic questions about change:

1. Has a new maintenance release or patch been installed recently?

2. Were unique types of data being run at the time the problem occurred?

3. Did the volume of data input change prior to the occurrence of the problem?

4. Has anyone recently modified any of the programs that were being run at the time the problem occurred?

5. Have there been any environmental changes recently?

6. Have you recently identified a problem in another software product?

7. Were new functions or features being used at the time the problem occurred?

8. Did input parameters or control files change prior to the occurrence of the problem?

If your answer to any of these questions is yes, the problem may be related to the change the question identified. If you can, test the suspected cause by reversing the change, although this admittedly may not be practical. If you feel confident that you have found the problem, you are ready to report it.

At this point, try to reproduce the problem. If you can reproduce it, try to develop a test case or procedure so that Burroughs service people can also reproduce it.

However, if you still cannot define the problem, proceed to phase three.

Break the Problem Down

Phase three is a bit more complex than phases one and two, because the probable causes are more elusive. To find these causes, you break the problem down progressively. First you identify the essential conditions of the problem, then you break those conditions down into their conditions, and so on, until you can define the problem.

Identifying the Essential Conditions

You identify the essential conditions by stating as much as possible about exactly what happens when the problem occurs. To do this, find answers to the following questions:

1. When does the problem occur?
2. What programs were running when the problem occurred?
3. What did the operator do just before the problem occurred?
4. How many users were on-line and what was each doing just before the problem occurred?

Usually after you have asked these questions, you will find that you have many conditions to work with. If you do not have enough information, try to duplicate the problem.

Now try to narrow down the list of essential conditions by testing the conditions on the list. Try various combinations of conditions until you are satisfied that you have all of, and only, the essential conditions. Start by asking yourself the following questions:

1. Is the problem likely to be related to all the items in the list?

2. If I drop one or two conditions, does the problem still occur?

Breaking Down the Essential Conditions

Once you have reduced the list, break each essential condition down into its essential conditions, using the same process. Once you seem to have the problem defined, test the definition one more time by asking yourself the following questions:

1. Have I considered all the pertinent data?

2. Does the problem always occur when all the essential conditions are present?

When you feel confident the problem is defined, try to reproduce it. If it is reproducible, try to develop a test case so Burroughs can see how you arrived at your definition.

If you can't develop a test case, gather the media you think might help Burroughs to reproduce the problem.

Defining a problem can be a complex process, and you must use your own judgment to determine how far to break down a problem. If you have spent a reasonable amount of time and still do not have the probable cause, stop and define the problem in terms of the essential conditions you have isolated. Then report the problem and let Burroughs take over from there.

WHERE TO GO FOR HELP

Once you have defined the problem to the best of your ability, the next step depends on the type and urgency of the problem. The remainder of this section describes the procedure customers within the United States should follow. International customers should consult their Burroughs representative because the procedures vary from country to country.

Hardware Problems

If you are a domestic customer and the problem is in the hardware, call the Regional Support Center, using the toll-free phone number listed on a sticker on your equipment. Also on the sticker is the serial number, called the RESPOND ID, for your system. Have this ID number, as well as the serial number of any other applicable equipment, ready to give to the person answering the call. If you need the assistance of a field engineer, the center will dispatch one to your site.

If the field engineer needs additional assistance, he or she can request help from a Remote Support Center. From these centers, specially-trained customer support engineers can diagnose and isolate problems at the customer's site. The same capabilities and software interface to the system that are available to the on-site field engineer are available to the customer support engineer. A datacomm connection to the customer's system enables the support engineer to run diagnostic programs, take direct system readings, and access system files such as system logs.

You can control the amount of access the Remote Support Center has to your site through hardware switches, through the datacomm connection, and through the software security system. Each time the Center accesses your system, the engineer must follow a log-on procedure. In addition, your site must initiate the datacomm connection. See your field engineer for more information on controlling access.

Software Problems

If the problem is in the software, you can report these problems to the Software Support Center by phone or through Field Communications Forms (FCFs). The method you use depends on the urgency of the problem. The FCFs are explained in the topic "Reporting System Problems and Suggestions" in this section.

If the problem is one that needs speedy resolution, contact your Software Support Center by phone or Telex. Call the number listed in your service contract with Burroughs or given to you by your Burroughs representative. When calling the center, have the following information available:

- The system serial number (RESPOND ID) and the serial numbers of any other applicable equipment
- The customer name
- The system software release level
- The name of the product having the problem
- A description of the problem
- The frequency of occurrence
- The impact of the problem
- The name and phone number of the person to contact at your site for follow-up
- A listing of the program and other supporting documentation, if applicable

The Support Center will frequently request that you submit an FCF to document the problem if it has not been answered or resolved by phone or Telex. When you have completed the FCF, forward it to your local support organization. The local organization will acknowledge receipt of the FCF, and, where possible, propose a detour. It will then forward the FCF to the applicable in-house support organization, which will assume responsibility for finding a solution.

User Groups

User groups are a good source of specialized information concerning the use of computer systems. There are several groups available worldwide for Burroughs system users.

CUBE

CUBE, Incorporated (Cooperating Users of Burroughs Equipment) is an independent association composed of Burroughs customers. The group operates as a customer clearinghouse for the exchange of information among themselves and with Burroughs. The type of information exchanged concerns use of Burroughs equipment, software products, and services.

Burroughs supports and cooperates with CUBE but does not run the organization. The group conducts its own activities with the assistance of a full-time executive secretary who, as required by CUBE by-laws, is a Burroughs employee. The other officers are volunteers elected annually from the membership.

The principal activity of CUBE is semiannual conferences for Burroughs worldwide customers. These meetings are three to four days long and are usually held in April and October in major cities in North America. The conferences consist of over 120 separate sessions for presentation of papers and for discussions on effective computer management and technical hardware and software topics. The speakers may be other customers, Burroughs representatives, or outside experts.

In addition, there are displays featuring new product offerings from CUBE members and from Burroughs Corporation. Members have the opportunity to meet and talk with other users with the same kind of Burroughs equipment and from the same or a related line of business. Members can also meet both technical and management employees of Burroughs Corporation.

CUBE membership is open to employees of any customer or of any organization that uses a Burroughs computer system. The system must have been identified by the CUBE executive board as an eligible system. Eligible systems currently range from the B 20 to A Series systems. To become a member, send your name, company name and mailing address, and the name of the Burroughs equipment installed or purchased to:

CUBE Secretary
P.O. Box 33053
Detroit, MI 48232

When you are a member, you automatically receive notices of CUBE conferences, meeting reports, and education schedules. You maintain your membership by attending a conference or by notifying the CUBE secretary of your desire to continue.

There are no annual dues or fee schedules. CUBE finances all its activities from the registration fees charged each participant at the conferences.

International User Groups

There are many other user groups for international customers of Burroughs. For example, ABCU is the European association of Burroughs computer users and operates similarly to CUBE. See your Burroughs representative for information about joining the user group in your area.

REPORTING SYSTEM PROBLEMS AND SUGGESTIONS

The Field Communication Form (FCF) is the means by which Burroughs customers report problems in and suggest new features and improvements to software, hardware, and documentation. The FCF is a six-part form, with a key to filling in the form on the back of the last page.

Each site should maintain a log of FCFs submitted to Burroughs. This log serves several purposes:

1. Shows that a problem has already been submitted, preventing duplication

2. Identifies FCFs still outstanding

3. Assists in compiling the history of a particular problem

When your support center receives an FCF, the coordinator assigns a unique number to it, places the number on the FCF, and returns a copy to the originator. You should use this number to identify the FCF both in your log book and in any inquiries regarding the FCF.

An FCF that suggests additional system capabilities or expresses a desire to have the product function differently is called a New Feature Suggestion (NFS).

An FCF that reports a problem is called a Field Trouble Report (FTR). If you submit an FTR, it must be accompanied by all relevant information, such as program dumps and source listings, that will assist Burroughs in investigating the problem. All machine-readable media except cassettes will be returned if you so request. Each piece of media must have a label attached stating that the media is to be returned and listing the return address.

General Problems

All FTRs submitted should include, either on the FCF or as an attachment, the following items:

1. The release levels of the compiler, MCP, and pertinent software

2. An identification of all local and Burroughs-supplied patches in the pertinent configuration

Data Communications Problems

If you submit an FCF concerning a data communications problem, include the following items when available:

1. A description of the hardware configuration, including the processor, Network Support Processors (NSPs), Line Support Processors (LSPs), Data Communications Data Link Processor (DC-DLP), peripherals, and the network. The network description should include a specification for each line, showing line speed and if the line is synchronous, asynchronous, direct, or modem correct. Also indicate types of stations and terminals attached and the make and model of the modem. A Network Definition Language II (NDLII) source listing can be substituted for all the preceding information if it exactly reflects the physical network.

2. A description of the software used, including the name, release level, and version of the network and Message Control System (MCS) and generator, if used

3. An NDLII source listing of the network controller

4. An MCS source listing or generation listing if a generative MCS is used

5. Any applicable application program source listings

6. A dump of the MCS at the appropriate point

7. A trace of the network controller if available

8. A monitor of messages into and out of the MCS or into and out of the line adaptors

9. An exact description of the symptoms of the problem

Data Management Problems

If you submit an FTR concerning a Data Management System II (DMSII) product, include whichever of the following attachments is applicable. In all cases, include any information and material that helps to reproduce the problem.

1. For Data and Structure Definition Language (DASDL) problems: Include a program dump of the compiler and a machine-readable copy of the DASDL source. If the problem occurs during an update or reorganize compile, also include the original DASDL listings, and the description file (before and after the program dump).

2. For database recovery problems: Include a copy of the audit file being used at the time of the problem and a program dump of the recovery program. Also include information on the reason for recovering, such as Master Control Program (MCP) dumps for MCP interrupt recovery and ODT logs pertaining to the system halt or program abort.

3. For unexpected DMSII exceptions (LIMITERRORS, DEADLOCK, etc.): Include a program dump at the point of exception, with the Data Base Stacks (DBS) option set, plus a compiler listing of the application program.

4. For all other DMSII problems: Include MCP dumps and description and control files, DASDL listings, and data files, if appropriate.

Compiler Problems

If you submit an FTR concerning language compilers, include the following items:

1. The source program on tape, plus a listing. In addition, include all copy libraries and included files.

2. The object code on tape.

3. If DMSII is involved, a description file and the DASDL source program that generated this description file.

4. A list of the settings of the compiler control options during the compilation of the source program. If the compiler has been recompiled, include any changes applied and the settings of the compiler control options during the compiler recompilation.

5. The steps to take to reproduce the problem, including input data.

6. A program dump, if the problem results in a fault.

<u>Completing Field Communication Forms (FCFs)</u>

The following pages describe how to fill out an FCF. A copy of the form is included in this section immediately after the directions. Before you start, there are a few general guidelines to keep in mind.

- o Describe the problem concisely and avoid jargon.
- o Report only one problem per FCF.
- o Use the actual FCF form, not a copy or facsimile.
- o Make sure the information on the form is legible (preferably typewritten).
- o Identify all attachments.
- o Include all indicated attachments with the FCF.

In the following paragraphs, the fields of the FCF are described in the order in which you would fill them in. Some of the fields are applicable to all types of FCFs, while others are applicable only to FTRs. To differentiate these fields, those that are applicable to all types of FCFs are marked "All Types", and those that are applicable only to FTRs are marked "FTR".

Support Activity Location

This field is for Burroughs use only and indicates the Support Activity to which the FCF is ultimately forwarded.

Class (All Types)

Class indicates the type of product being discussed. Enter "1" for hardware, "2" for system software, "3" for application software, or "4" for maintenance test routines.

Type (All Types)

Type refers to the kind of communication. Enter "1" if you are reporting a problem, "2" if you are suggesting a new feature or improvement, and "4" if this FCF concerns any other subject.

Reference # (All Types)

When the Burroughs Support Center receives the FCF, it assigns a reference number to it that is cross-referenced to the customer's number. When you submit attachments with an FCF, first obtain a reference number from your assigned Support Center and complete this field, then mark all attachments with this number. The reference number is divided into the following parts:

Dist/Sub/Country

This number identifies the Burroughs district or subsidiary that is submitting the FCF to the Support Activity.

Branch/Location

This number identifies the branch or location within the district or subsidiary (for domestic customers only).

Unique Seq Id

This number is a unique identification assigned by Burroughs for this FCF.

Product (All Types)

If this FCF is for a software product, enter the name of the product in the first set of boxes following the word "Product". Enter only one letter per box. If the FCF is for hardware, enter the top unit number, right justified, of the equipment. For a new feature suggestion, enter the name of the product. For documentation suggestions, enter the form number of the document.

For software products, enter the release level of the product in the second set of boxes on the "Product" line. Right-justify all numbers and include the major release number, the level number within that release, and the four-digit patch level number; for example, 036 130 0123. The four-digit patch level number (which is 0123 in this example) is optional.

MCP Version (All Types)

If applicable, enter the style identification (the name MCP) and the version number of the MCP being used with the product. For example, enter "MCP 036 170 0123".

Firmware/Other (All Types)

Enter the style id (name) and complete release level number of the firmware or other software product used with the product involved in the problem; for example, NSP firmware and level, or compiler and level.

Date Originated (All Types)

Enter today's date using the MMDDYY format.

System Style (All Types)

This field is for the style id of the processor; for example, A9B or A3D.

Unit Style (All Types)

For hardware FTRs, enter the style id of the unit; for example, for the Daisy Wheel printer on the A 9, enter "AP1300". For system or application software, enter the product style id; for example, for ERGO on a B 7000, enter "B7000 ERG".

Unit Serial # (FTR)

This field applies to hardware FTRs only. Enter the serial number of the equipment having the problem.

Concise Description (All Types)

Enter a brief statement of the reason this FCF is being written. This statement should be as clear as possible because it becomes part of the published description of the FCF.

Frequency (FTR)

This field applies only to FTRs. Enter "1" if the problem occurred once, and enter "2" if it occurred more than once.

Product Status (FTR)

Enter the number that corresponds to the current status of the product. This field indicates the seriousness of a problem.

Attachments (FTR)

Check the boxes that indicate the type of attachments you are including with this FCF. If there is more than one of a given type, put the number in the box rather than a check mark; for example, if there are three program listings, put the number "3" in box 7. BE SURE THAT ALL ATTACHMENTS ARE CLEARLY MARKED WITH THE ASSIGNED REFERENCE NUMBER. See "Reference Number" earlier in this description.

Include all information that contributes to the solution of the problem. If the problem is not reproducible, include any observations by site personnel in addition to the system log. If the problem is reproducible, include a test case, or a procedure that demonstrates the problem, or both.

Other types of attachments that you should always consider for inclusion are the following:

1. A full memory dump, taken at the time the problem occurred, that has been saved by DUMPANALYZER and copied to tape

2. A printed trace of the sequence of instructions leading to the problem

3. The system log for the pertinent time period

4. Program object code in machine-readable form

5. A program dump

6. A source listing with cross reference of the programs involved in the problem

7. A copy of the source program in machine-readable form

8. A copy of the input data, in machine-readable form, that leads to the problem

9. A copy of the output data demonstrating the results of the problem

10. Any notes, information, and comments by the individual who analyzed the problem

11. A note of any Burroughs reference material, with section and page numbers, that is inconsistent with the performance of the software

Reproducible (FTR)

If information being submitted with an FTR enables the problem to be reproduced by the support group, enter "1"; otherwise, enter "2".

Installation Impact (FTR)

This field should be a brief statement indicating the effect of the problem on your installation.

Pertinent Configuration (FTR)

If the configuration has any effect on the problem, list the components and pertinent configuration information here. For example: unusual configurations, shared systems, system registers, allocation tables, unusual program mixes, and so on.

System Serial # (FTR)

Enter the serial number of the system.

Full Description (All Types)

For FTRs, this field should begin with a clear statement of the problem, followed by all relevant information, in detail. Describe the circumstances of the problem and the cause, if possible, including any possible external causes such as power surges. If there is not enough room, attach additional sheets and note this.

For new feature suggestions or improvements, use this field to describe the details of the suggestion. Note that Burroughs welcomes suggestions but makes no guarantees concerning their implementation.

Means to Correct/Avoid Problem (FTR)

Identify the current means, such as patches or operational procedures, used to avoid the problem. For hardware FTRs, list the components or parts used to correct the problem.

Reviewed By

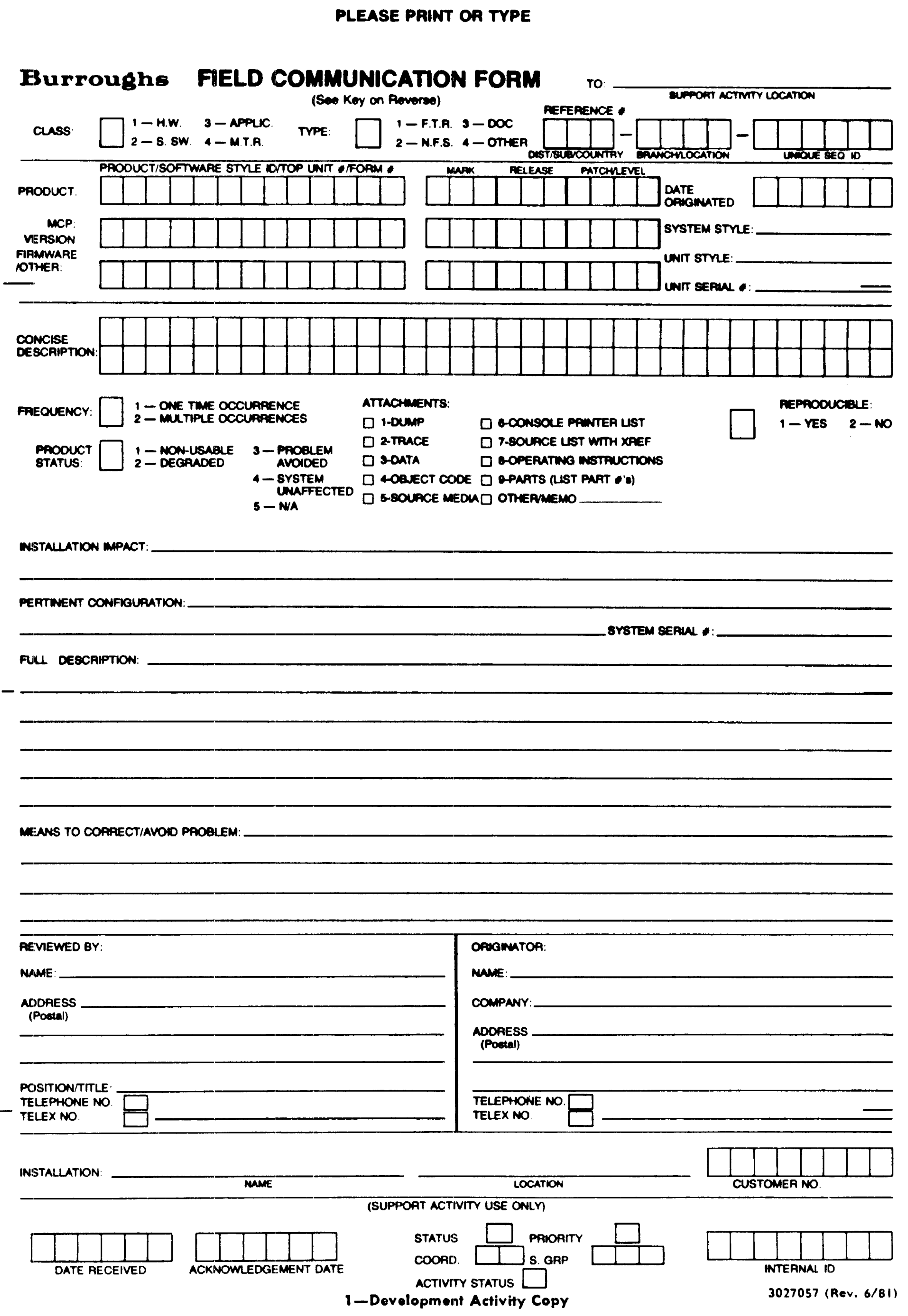
This area is for Burroughs use only. It is completed by the Burroughs Support Center that is forwarding this FCF.

Originator (All Types)

The identity of the person originating this FCF.

Installation (All Types)

The name and location of the customer site (company name), as well as the identifying site number assigned by Burroughs.



PLEASE PRINT OR TYPE

Burroughs FIELD COMMUNICATION FORM
(See Key on Reverse)

TO: ________ SUPPORT ACTIVITY LOCATION

CLASS: ☐ 1 — H.W. 3 — APPLIC. 2 — S. SW. 4 — M.T.R.

TYPE: ☐ 1 — F.T.R. 3 — DOC 2 — N.F.S. 4 — OTHER

REFERENCE #: ☐☐☐ — ☐☐☐☐ — ☐☐☐☐☐☐
DIST/SUB/COUNTRY BRANCH/LOCATION UNIQUE SEQ ID

PRODUCT/SOFTWARE STYLE ID/TOP UNIT #/FORM # — MARK RELEASE PATCH/LEVEL

PRODUCT: DATE ORIGINATED

MCP VERSION: SYSTEM STYLE: ________

FIRMWARE /OTHER: UNIT STYLE: ________

UNIT SERIAL #: ________

CONCISE DESCRIPTION:

FREQUENCY: ☐ 1 — ONE TIME OCCURRENCE 2 — MULTIPLE OCCURRENCES

PRODUCT STATUS: ☐ 1 — NON-USABLE 2 — DEGRADED 3 — PROBLEM AVOIDED 4 — SYSTEM UNAFFECTED 5 — N/A

ATTACHMENTS:
☐ 1-DUMP ☐ 6-CONSOLE PRINTER LIST
☐ 2-TRACE ☐ 7-SOURCE LIST WITH XREF
☐ 3-DATA ☐ 8-OPERATING INSTRUCTIONS
☐ 4-OBJECT CODE ☐ 9-PARTS (LIST PART #'s)
☐ 5-SOURCE MEDIA ☐ OTHER/MEMO ________

REPRODUCIBLE: ☐ 1 — YES 2 — NO

INSTALLATION IMPACT: ________

PERTINENT CONFIGURATION: ________

SYSTEM SERIAL #: ________

FULL DESCRIPTION: ________

MEANS TO CORRECT/AVOID PROBLEM: ________

REVIEWED BY:	ORIGINATOR:
NAME: ________	NAME: ________
ADDRESS (Postal) ________	COMPANY: ________
	ADDRESS (Postal) ________
POSITION/TITLE: ________	
TELEPHONE NO. ☐	TELEPHONE NO. ☐
TELEX NO. ☐ ________	TELEX NO. ☐ ________

INSTALLATION: ________ NAME ________ LOCATION ☐☐☐☐☐☐☐☐ CUSTOMER NO.

(SUPPORT ACTIVITY USE ONLY)

DATE RECEIVED ACKNOWLEDGEMENT DATE STATUS ☐ PRIORITY ☐ COORD. ☐☐ S. GRP ☐☐☐ ACTIVITY STATUS ☐ INTERNAL ID

1—Development Activity Copy

3027057 (Rev. 6/81)

7 WHERE TO FIND MORE INFORMATION

You can find more information on the subjects discussed in this book in the A Series software documentation published by Burroughs. In addition, on-line documentation is available when you are using MARC (Menu-Assisted Resource Control) to access the system, and when you are using software products accessed through standard MARC menus.

This section lists the available A Series documentation, first by subject and then alphabetically. The alphabetical listing includes the form number and a brief description of the manual contents. The section next explains how to order documentation and the conventions followed by A Series documents.

DOCUMENTATION LISTED BY SUBJECT

In this part of the section, the manuals are sorted by subject. When manuals cover more than one subject, they are listed under the two most applicable.

Database Management

Advanced Data Dictionary System (ADDS) Capabilities Manual
Advanced Data Dictionary System (ADDS) Installation and Operations Manual
Advanced Data Dictionary System (ADDS) User's Guide
DMSII DataAid User's Guide *
DMSII Data and Structure Definition Language (DASDL) Reference Manual
DMSII Data Base Certification Software Operation Guide
DMSII Data Dictionary Reference Manual
DMSII Inquiry Software Operation Guide
DMSII Interpretive Interface User's Manual
DMSII Transaction Processing System (TPS) Programmer's Manual
DMSII User Language Interface Software Operation Guide
DMSII Utilities and Operations Guide
Extended Retrieval with Graphic Output (ERGO) Capabilities Manual
Extended Retrieval with Graphic Output (ERGO) User's Guide

*Note: DMSII = Data Management System II

Data Communications

Burroughs Network Architecture (BNA) Architectural Description Reference Manual
Burroughs Network Architecture (BNA) Network Control Reference Manual
Burroughs Network Architecture (BNA) Program Agent User's Guide
Burroughs Network Architecture (BNA) User's Guide
DataComm Processor (DCP) to Network Support Processor (NSP) Configuration Conversion Aid
Interactive Datacomm Configurator (IDC) User's Guide
Network Definition Language (NDL) Reference Manual
Network Definition Language II (NDLII) Reference Manual

Languages

ALGOL Reference Manual
APL/700 Installation Manual
APL/700 User's Reference Manual
APL/700 Utilities Manual
APLB User Reference Manual
BASIC Language Reference Manual
Binder Reference Manual
COBOL Reference Manual
COBOL ANSI-74 Reference Manual
DCALGOL Reference Manual
DOCUMENT/DMALGOL
FORTRAN Reference Manual
FORTRAN77 Reference Manual
DOCUMENT/NEWP
Pascal Reference Manual
PL/I Reference Manual
Report Program Generator (RPG) Debugging Template
Report Program Generator (RPG) Reference Manual
Sort Language Reference Manual
Work Flow Language (WFL) Reference Manual

Message Control Systems

CANDE Operations Manual
CANDE Reference Manual
Communications Management System (COMS) Capabilities Manual
Communications Management System (COMS) Operator's Guide
Communications Management System (COMS) Planning and Installation Manual
Communications Management System (COMS) Programmer's Guide
Communications Management System (COMS) Migration Guide
DiagnosticMCS Reference Manual
Menu-Assisted Resource Control (MARC) User's Guide
Remote Job Entry (RJE) Reference Manual

Where to Find More Information

Multilingual Capabilities Management

Interactive Menugraph Generator (IMG) User's Guide
Message Translation Utility User's Guide
MultiLingual System (MLS) User's Guide

Operations

A 3 System Software Installation Guide
A 9 System Software Installation Guide
A 10 System Software Installation Guide
B 7900 Systems Capabilities and Features Manual
B 7900 System Operator's Guide
Menu-Assisted Resource Control (MARC) User's Guide
Operator Display Terminal (ODT) Reference Manual
Print System (PrintS/ReprintS) User's Guide
Printing Utilities User's Guide
Mark 3.6.0 System Software Installation Guide
Work Flow Language (WFL) Reference Manual

Performance Monitoring

SMFII Capabilities Manual *
SMFII Query Program Manual
SMFII Site Management Manual
SMFII System Resource Management Manual
Systemstatus Reference Manual

*Note: SMFII = System Management Facility II

Peripherals

INFOVIEW (tm) System Software Operation Guide
Intelligent Distributed Editor (IDE) Program Development Guide
Intelligent Laser Printing System Forms Manager User's Guide
Intelligent Laser Printing System (ILPS) User's Guide
Workstation Software Operations Guide

Programming Productivity Aids

ALGOL Test And Debug System (TADS) User's Guide
Binder Reference Manual
COBOL ANSI-74 Test and Debug System (TADS) User's Guide
Editor Capabilities and Features Manual
Editor User's Guide
FORTRAN77 Test and Debug System (TADS) User's Guide
Help Utility User's Guide
Intelligent Distributed Editor (IDE) Program Development Guide
Interactive Menugraph Generator (IMG) User's Guide
Screen Design Facility (SDF) Capabilities Manual
Screen Design Facility (SDF) User's Manual

Progression

B 1000 Series to A Series Progression Guide

Reference Cards

APL/700 Operations Card

System Software

Disk Subsystem Software Overview
I/O Subsystem Reference Manual
Memory Subsystem Overview
Physical I/O Overview
System Configuration Guide
Mark 3.6.0 System Software Installation Guide
System Software Site Management Reference Manual
System Software Support Reference Manual
System Software Utilities Reference Manual

DOCUMENTATION LISTED ALPHABETICALLY

Each entry in this section includes the publication title, the form number, and a brief description of the contents.

A 2/A 3 System Software Installation Guide - 1170081

Describes how to initialize and Halt/Load an A 2/A 3 system.

A 3 Advanced System Software Installation Guide - 1170073

Describes how to initialize and Halt/Load an A 3 Advanced system.

A 3 Dual Processor System Software Installation Guide - 1195096

Describes how to initialize and Halt/Load an A 3 Dual Processor system.

A 3 System Software Installation Guide - 1169679

Describes how to initialize and Halt/Load an A 3 system.

A 9 System Software Installation Guide - 1169695

Describes how to initialize and Halt/Load an A 9 system.

A 10 System Software Installation Guide - 1169935

Describes how to initialize and Halt/Load an A 10 system.

Advanced Data Dictionary System (ADDS) Capabilities Manual - 1180569

Describes the capabilities and benefits of the Advanced Data Dictionary System. ADDS is a tool for the definition, storage, and retrieval of data definitions for databases and application programs.

Advanced Data Dictionary System (ADDS) Installation and Operations Manual - 1143237

Provides instructions for the installation and daily operation of the Advanced Data Dictionary System.

Advanced Data Dictionary System (ADDS) User's Guide - 1180551

Describes how to use the functions of the Advanced Data Dictionary System.

ALGOL Reference Manual - 1169844

Describes the ALGOL language, including Burroughs extensions for communicating between programs and I/O devices, for editing data, and for debugging programs.

ALGOL Test And Debug System (TADS) User's Guide - 1169539

Describes the use of the Test And Debug System for ALGOL programs.

APL/700 Installation Manual - 5000805

Describes the installation of APL/700, a procedure-oriented language, at user sites.

APL/700 Operations Card - 5001928

Describes, in diagram form, the syntax of APL/700 commands.

APL/700 User's Reference Manual - 5000813

Describes the syntax and semantics of APL/700.

APL/700 Utilities Manual - 5001910

Describes the function of and operating procedures for the utilities available to APL/700 users. These utilities give the APL/700 user access to system features not directly available through the language.

APLB User Reference Manual - 1182532

Describes the syntax and semantics of APLB, a language for numerical analysis applications.

B 1000 Series to A Series Progression Guide - 1180595

Provides instruction and tips for progressing from the Burroughs B 1000 Series to the Burroughs A Series and B 5000/B 6000/B 7000 Series systems. This manual discusses the differences encountered when progressing from the small systems to the A Series, describes common progression problems and solutions to these problems, and gives general information for handling the overall progression process.

BASIC Language Reference Manual - 1182433

Describes BASIC, a language developed for novice users, and contains some information on the computer and the compiler.

Binder Reference Manual - 5014582

Provides the Binder syntax and techniques required to bind externally-compiled subprograms to an executable program.

Burroughs Network Architecture (BNA) Architectural Description Reference Manual - 1132172

Describes Burroughs Network Architecture in depth. The manual presents information for network planners and supervisors, applications and network systems programmers, and operators.

Burroughs Network Architecture (BNA) Network Control Reference Manual - 1132180

Describes the Network Control messages used to configure, control, and interrogate BNA Network Services functions. The manual is intended for use by programming and operations personnel.

Burroughs Network Architecture Version 1 (BNA V1) Program Agent User's Guide - 1169653

Describes the BNA Program Agent feature, which allows a user to write a program to issue BNA commands to hosts in a BNA network.

Burroughs Network Architecture Version 1 (BNA V1) User's Guide - 1169687

Provides an overview of BNA, installation instructions for A Series systems, and the syntax and semantics of the network operation commands.

CANDE Operations Manual - 1170065

Contains information about CANDE for those interested in operational control of CANDE and its datacomm network, and those interested in tailoring CANDE to the specific requirements of an installation.

CANDE Reference Manual - 1169869

Describes the commands for CANDE, a time-sharing MCS for communication and program creation and execution.

COBOL ANSI-74 Reference Manual - 1169877

Describes COBOL ANSI-74 as implemented for Burroughs systems.

COBOL ANSI-74 Test and Debug System (TADS) User's Guide - 1169901

Describes the basic steps for using COBOL74 Test and Debug System (TADS), including preparing a program for testing, creating a test environment, initiating a test session, and using the system's interactive commands.

COBOL Reference Manual - 1169786

Describes the COBOL(68) language as implemented for Burroughs systems.

Communications Management System (COMS) Capabilities Manual - 1180494

Provides an overview of the features and capabilities of the Communications Management System. The intended audience is customer executives and data processing administrators.

Communications Management System (COMS) Operator's Guide - 1154523

Describes how to use the Communications Management System, including detailed instructions for controlling sessions, windows, and the network, and procedures for initialization, shutdown, backup, and recovery. The intended audience is computer operators.

Communications Management System (COMS) Planning and Installation Manual - 1185238

Provides information and instructions on planning for and installing the Communications Management System. Included are detailed, step-by-step instructions for planning and implementing the configuration file. The intended audience is analysts who will be installing COMS.

Communications Management System (COMS) Programmer's Guide - 1185303

Describes the Communications Management System and how to use its features when writing applications. Included are details about using the DCI library, controlling messages, using the service functions, and programming for synchronized recovery.

Communications Management System (COMS) Migration Guide - 1185337

Contains information of primary use to system administrators and system programmers responsible for their site's progression to the 3.6 release of COMS. This manual describes the available strategies, tools, and methods of evaluation for progression.

(A) Customer Guide to Burroughs Services and Support - 1147394

Describes Burroughs services available for the support of your systems. It is an overview written for information systems and data processing managers and other management personnel who have responsibility for the planning and overall operations of their data processing systems.

Customer Guide to Software Reporting - 1130838

Describes the procedure Burroughs systems users should follow to resolve problems with software products.

DataComm Processor (DCP) to Network Support Processor (NSP) Configuration Conversion Aid - 1182193

Describes the interactive, menu-driven utility that allows the conversion of the configuration section of a DCP datacomm network to the configuration section of an NSP datacomm network. This manual also describes the instructions and commands for the conversion process.

DCALGOL Reference Manual - 5014574

Describes the Data Communications ALGOL language, an extension of ALGOL designed for the creation and control of data communications systems and Master Control Program (MCP) interfaces.

DiagnosticMCS Reference Manual - 1169596

Describes the functions and commands of SYSTEM/DIAGNOSTICMCS, a DCALGOL program for the verification and testing of data communications subsystems.

Disk Subsystem Software Overview - 1169992

Describes the concepts and operation of the disk subsystem on A Series systems.

DMSII DataAid User's Guide - 1180544

Describes the use of DataAid, an interactive menu system that allows the description, initialization, loading, and accessing of DMSII databases.

DMSII Data and Structure Definition Language (DASDL) Reference Manual - 1163805

Describes DASDL, the language used by a database administrator to describe the physical and logical characteristics of a database.

DMSII Data Base Certification Software Operation Guide - 1164050

Describes the use of the interactive utility program that determines the integrity of a DMSII database.

Where to Find More Information

DMSII Data Dictionary Reference Manual - 1177128

Describes the functions of the Data Dictionary, a DMSII database that contains information about other DMSII databases and transaction bases and the programs that reference them.

DMSII Inquiry Software Operation Guide - 1164035

Describes the use of the DMSII INQUIRY system, which is an interactive tool for examining information in a DMSII database.

DMSII Interpretive Interface User's Manual - 1163813

Describes the Interpretive Interface, which allows dynamic, run-time access to DMSII databases for programming languages that do not have DMSII extensions.

DMSII Transaction Processing System (TPS) Programmer's Manual - 1164043

Describes how to write programs that use the Transaction Processing System to handle transactions and to access a database.

DMSII User Language Interface Software Operation Guide - 1180536

Describes the use of the ALGOL, COBOL, RPG, and PL/I language extensions in accessing a DMSII database.

DMSII Utilities and Operations Guide - 1163839

Describes the function of and operating procedures for utilities available to users of the DMSII software.

Editor Capabilities and Features Manual - 5014400

Provides an overview of general capabilities, specific features, installation information, and reference materials for the Editor.

Editor User's Guide - 1169976

Describes the Editor commands for interactive creation and modification of program and data files.

Extended Retrieval with Graphic Output (ERGO) Capabilities Manual - 1163847

Provides an overview of the features and capabilities of the Extended Retrieval with Graphic Output system, a tool for database access and report generation.

Extended Retrieval with Graphic Output (ERGO) User's Guide - 1164027

Describes the use, output, and commands of the Extended Retrieval with Graphic Output system.

FORTRAN Reference Manual - 1182441

Describes both the programming language accepted by the FORTRAN compiler and the various options and control statements used with this compiler.

FORTRAN77 Reference Manual - 1182458

Describes both the programming language accepted by the FORTRAN77 compiler and the various options and control statements used with this compiler.

FORTRAN77 Test and Debug System (TADS) User's Guide - 1182490

Explains the basic steps for using FORTRAN77 TADS.

Help Utility User's Guide - 1169570

Describes how to use the Help Utility to produce on-line help text for application and system programs.

INFOVIEW (tm) System Software Operation Guide - 1182482

Describes the operation of, and the user interfaces to, the INFOVIEW (tm) workstation environment for ET2000 users.

Intelligent Distributed Editor (IDE) Program Development Guide - 1182474

Describes the commands and user interface of the integrated program and development environment, primarily used for FORTRAN77 program development.

Intelligent Laser Printing System Forms Manager User's Guide - 1169661

Describes the features and operation of the Forms Manager program, which allows the user to define forms for printing on the laser printer.

Intelligent Laser Printing System (ILPS) User's Guide - 1169729

Describes and provides an installation and operation guide for ILPS, an on-line electronic printing subsystem.

Interactive Datacomm Configurator (IDC) User's Guide - 1169810

Describes how to use this menu-driven utility to define and maintain the configuration of a data communications subsystem.

Interactive Menugraph Generator (IMG) User's Guide - 1169893

Describes how to use the Interactive Menugraph Generator, which is used to design and modify screen menus and forms.

I/O Subsystem Reference Manual - 1169984

Describes the Input/Output Subsystem, including file attributes and other system facilities concerned with physical and logical I/O operations.

Memory Subsystem Overview - 1169836

Provides an overview of memory management, including a discussion of relevant concepts and how these concepts are implemented on Burroughs systems.

Menu-Assisted Resource Control (MARC) User's Guide - 1169588

Describes the use of Menu-Assisted Resource Control, which is a menu-driven interface and a transaction processor for users and operators of Burroughs systems.

Message Translation Utility User's Guide - 1169554

Describes the utility for translating system and error messages from one human language (such as English) to another.

MultiLingual System (MLS) User's Guide - 1169646

Explains how output messages, on-line help text, and menu screens can be developed and accessed in multiple human languages, such as French and English.

Network Definition Language (NDL) Reference Manual - 1176690

Explains the language used to describe a data communications network physically, logically, and functionally. (Applies only to B 6800, B 7700, and B 7800 systems.)

Network Definition Language II (NDLII) Reference Manual - 1169604

Describes the NDLII language, which is used to define data communications networks in an MLIP environment. (Applies to B 5900, B 6900, B 7900, and A Series systems.)

Operator Display Terminal (ODT) Reference Manual - 1169612

Describes the commands available to the system operator through the Operator Display Terminal (ODT).

Pascal Reference Manual - 1169851

Describes the Burroughs implementation of Pascal.

Physical I/O Overview - 1169943

Provides a high-level overview of Physical I/O on the MLIP systems that support Universal I/O (B 5900, B 6900, A 3, A 9, and A 10).

PL/I Reference Manual - 1169620

Describes the PL/I compiler and language at an advanced level.

Print System (PrintS/ReprintS) User's Guide - 1169919

Explains the Print System (PrintS) and Remote Printing System (ReprintS) and how to use them.

Printing Utilities User's Guide - 1169950

Describes the two utilities that are part of the print system: the Backup Processor utility and the SYSTEM/BACKUP utility.

Remote Job Entry (RJE) Reference Manual - 1169828

Describes the use of the RJE message control system, which allows remote use of computer systems.

Report Program Generator (RPG) Debugging Template - 5015019

Explains how to use the RPG debugging template, a specification sheet for writing RPG programs.

Report Program Generator (RPG) Reference Manual - 1169760

Describes RPG, a machine-independent language. RPG allows applications programmers with a minimal knowledge of the operating system to write computer programs.

Screen Design Facility (SDF) Capabilities Manual - 1180437

Describes the capabilities and benefits of the Screen Design Facility, a tool for creating screens for on-line, transaction-based application systems.

Screen Design Facility (SDF) User's Guide - 1185295

Provides instructions for using the Screen Design Facility.

SMFII Capabilities Manual - 5012032

Provides an overview of and explains the capabilities of the System Management Facility (SMFII).

SMFII Query Program Manual - 5015696

Describes the functions and use of QUERY, a program used to analyze and report on SMFII data.

SMFII Site Management Manual - 5012016

Describes and explains how to use the Site Management modules of SMFII. These modules monitor the operating condition of the system's hardware and software.

SMFII System Resource Management Manual - 5015688

Describes and explains how to use the System Resource modules of SMFII. These modules measure the system workload and the utilization of system resources.

Sort Language Reference Manual - 1169794

Describes the Sort language and how to use it to sort and merge files. The manual also describes the compatibility between the A Series and B 1000 Series Sort languages.

System Configuration Guide - 1169968

Describes how to control through software the hardware resources connected to an A series or B 7900 system.

Mark 3.6.0 System Software Installation Guide - 1170040

Describes information crucial to the installation of the Mark 3.6 system software release.

System Software Site Management Reference Manual - 1170008

Contains information primarily of use to system administrators and their operations staff. It describes the software features associated with system installation and management.

System Software Support Reference Manual - 1170016

Contains detailed information about important system software and software operations. It is primarily intended for use by software support personnel.

System Software Utilities Reference Manual - 1170024

Contains information primarily of use to systems programmers. It describes utility programs for performing operations such as copying, comparing, and merging files.

Systemstatus Reference Manual - 1169638

Explains the SYSTEMSTATUS intrinsic of the Master Control Program (MCP). This intrinsic reports on the activity and environment of the system.

Work Flow Language (WFL) Reference Manual - 1169802

Describes the syntax and semantics of WFL, a language that provides programmatic control over the running of tasks.

DOCUMENTS Tape

The files listed below contain documentation information. They are released on the DOCUMENTS Tape that comes with your system software and can be printed at your site. The "Mark 3.6.0 System Software Installation Guide" provides procedures on how to print these files.

DOCUMENT/DMALGOL

Describes the DMALGOL language.

DOCUMENT/MARK36/SOFTWARE

Contains the "Mark 3.6.0 System Software Installation Guide," which provides information on the installation of the Mark 3.6 system software release.

DOCUMENT/NEWP

Describes the difference between NEWP and ALGOL.

DOCUMENT/SOFTCONFIGURATION

Describes the soft configuration of A 3, A 9, A 10, B 5900, and B 6900 systems.

HOW TO ORDER DOCUMENTATION

The printed manuals listed above are available for purchase by customers. The ordering procedure for them depends on whether you are a domestic (United States) or international customer.

Domestic (United States) Customers

You can order manuals on an individual order basis and on an ongoing, subscription basis. The individual order basis means you are requesting a one-time-only shipment of the ordered manuals. The subscription basis means you also are requesting automatic updates of the Publications Catalog and of the manuals. You specify which manuals should be automatically updated when you request Customer Subscription Service. Use the Subscription Order Form, No. 3028147, to request this service.

A catalog, "Customer Technical Publications," Form No. 1130010, is available to all customers free of charge from a Burroughs representative or from the Publication Distribution Center. To order any manuals, use Order Form 3020003, also obtainable from the Center or your representative. The catalog contains complete instructions on filling out this form and the Subscription Order Form.

Mail manual and subscription orders to:

Burroughs Corporation
Publication Distribution Center
Building 4
41100 Plymouth Road
Plymouth, MI 48170

International Customers

International customers should place orders with their local Branch, Subsidiary, or Distributor's office.

DOCUMENTATION CONVENTIONS

An important convention used in the software documentation is the one that governs the way commands are presented. This convention is used throughout A Series software documentation.

Commands are made up of a number of components, some required and some optional, which must be entered in a particular order. If the correct order is not followed, the system rejects the command. In order to present the command components in a consistent format and to show their correct order, Burroughs uses syntax diagrams. These diagrams, called railroad diagrams, graphically describe the command and show which items are required and which are optional, the order in which they should appear, how often you can repeat them, and any required punctuation.

The following pages describe the components of railroad diagrams and explain how to read them.

How to Read a Railroad Diagram

Normally, you read a railroad diagram from left to right. However, there are some exceptions; in those cases, arrows indicate a right-to-left direction.

If a diagram is too long to fit on one line and must continue on the next, a right arrow (>) appears at the end of the first line and another at the beginning of the next line, like this:

```
                                                  -------------------->

>-----------
```

The end of a railroad diagram is denoted by a vertical bar (|) or percent sign (%). The vertical bar means the command can be followed by a semicolon and another command. The percent sign means the command must be on a line by itself.

CONSTANTS AND VARIABLES

Consider a hypothetical command for giving instructions to a house painter:

```
-- PAINT ------------- LIVING ROOM ---<color>--|
         |       | |                |
         |- THE -| |- DINING ROOM -|
                   |                |
                   |- BEDROOM -----|
                   |                |
                   |- BATHROOM ----|
                   |                |
                   |- KITCHEN -----|
```

This command tells the painter to paint a designated room in the color you specify.

The example introduces two important features of railroad diagrams:

- Constants
- Variables

Constants

Constants are items that you cannot vary. You must enter a constant as it appears in the diagram, either in full or abbreviated. If you abbreviate a constant, you must enter everything that is underlined in the railroad diagram, optionally followed by one or more of the remaining characters.

You can recognize constants in railroad diagrams by the fact that they are never enclosed in broken brackets.

In the example, the word PAINT is a constant. You could enter PAINT in full or abbreviate it to PAI or PAIN, but not to PA or PAN. If no part of the constant is underlined, you cannot abbreviate it at all.

Variables

Variables are items that you can replace with other data to suit a particular situation; that is, you can vary the information you enter in place of the variable, subject to rules defined for the particular command or statement.

Variables appear in a railroad diagram enclosed in broken brackets (<>).

In the example, <color> is a variable item. If the description of the PAINT command defines the allowable colors as BLUE, GREEN, PINK, and YELLOW, you can enter any one of these in your command.

FOLLOWING THE PATHS OF A RAILROAD DIAGRAM

The paths of a railroad diagram lead you through the diagram from beginning to end. They are represented by horizontal and vertical lines.

A path shows the allowable syntax. Some diagrams have just one path that goes from the beginning to the end of the diagram. Others contain several paths, each covering a part of the diagram. A path shows which items you can include in a command or statement, which you can omit, and the number of times you can include a particular item or group of items.

To follow a path through a railroad diagram, you need to understand the items you may encounter along the way. These items are

- Required items and punctuation
- Optional items
- Loops

A description of each item follows.

Required Items and Punctuation

Required items and punctuation must be entered in the command or statement; you cannot omit them. A required item appears by itself in a path (horizontal line). A required item can be either a constant or a variable. For example, if a railroad diagram indicates

```
-- PAINT -- BEDROOM --<color>--|
```

the words PAINT and BEDROOM are required constants, and <color> is a required variable. You could correctly enter

```
PAINT BEDROOM BLUE
```

but not

```
PAINT BEDROOM
```

because the required item <color> would be missing.

Optional Items

Optional items appear one below another in a vertical list. You can choose any one of the items in the list. If the list also contains an empty path (all dashes), you can omit the item entirely. An optional item can be either a variable or a constant. The PAINT command in the example contains two lists. The first is

```
-------------|
  |       |
  |- THE -|
```

which gives you two options:

- Enter the constant THE

- Omit it (because there is an empty path)

The second list has five optional constants:

```
---- LIVING ROOM ----|
  |
  |- DINING ROOM -
  |
  |- BEDROOM -----
  |
  |- BATHROOM ----
  |
  |- KITCHEN -----
```

You must enter any one of the optional items (LIVING ROOM, DINING ROOM, BEDROOM, BATHROOM, or KITCHEN) because there is no empty path in this list.

Loops

A loop is an item or group of items that you can repeat. The number of repetitions allowed is controlled by an item called the bridge.

A loop can span all or part of a railroad diagram. It always consists of at least two horizontal lines, one below the other, like this:

```
    |<------- <return character> ---------|
    |                                     |
-----<bridge>--<content of the loop>-------
```

or

```
    |<-- <bridge> -- <return character> --|
    |                                     |
--------- <content of the loop> -----------
```

The bridge shows the maximum number of times you can go through the loop. The bridge can precede the contents of the loop, or it can precede the return character on the upper line of the loop to specify the number of times the right-to-left path can be traversed. The bridge is an integer enclosed in sloping lines, / \, for example, /4\. Not all loops have bridges. Those that do not can be repeated any number of times.

The top line is a right-to-left path that contains information about repeating the loop. The return character is the character to use before each repetition of the loop (often, a comma). Not all loops contain a return character; if none is shown, just enter one or more spaces before repeating the loop.

The other lines show the content of the loop (the data you enter each time you go through the loop). This can be any combination of optional items, required items, lists, and even other loops. The content of a loop can range from simple (one item), to very complex (many items, lists, and loops).

Example 1. A Simple Loop

The PAINT command as first shown is of limited usefulness. To tell the painter to do several rooms, you need a separate command for each room. It would be much easier if you could tell him to do several rooms in one command.

You can do that by making the list of rooms into a loop. The command would then look like this:

```
                     |<---------- , ---------|
                     |                       |
-- PAINT ------------/5\--- LIVING ROOM -----<color>--|
          |       |      |              |
          |- THE -|      |- DINING ROOM -|
                         |              |
                         |- BEDROOM -----|
                         |              |
                         |- BATHROOM ----|
                         |              |
                         |- KITCHEN -----|
```

The bridge has a value of 5, so you can go through the loop up to five times, for a total of five rooms. The return character is a comma, which you must enter before repeating the loop content.

You can now enter

    PAINT THE LIVING ROOM, BEDROOM, KITCHEN YELLOW

or

    PAINT DINING ROOM, BEDROOM, BATHROOM BLUE

or

    PAINT BEDROOM PINK

or

```
PAINT BEDROOM, BATHROOM, BEDROOM, BEDROOM BLUE
```

or any other valid combination.

This simple loop makes the PAINT command more versatile, but a significant drawback remains. Although you can include up to five rooms in a command, you cannot specify different colors.

Example 2. A More Complex Loop

If the content of the loop were to include the color, you could specify a different color for each room.

```
                        |<-------------- , -------------|
                        |                               |
-- PAINT -------------/5\--- LIVING ROOM ---<color>----|
         |       |       |              |
         |- THE -|       |- DINING ROOM -|
                         |              |
                         |- BEDROOM -----|
                         |              |
                         |- BATHROOM ----|
                         |              |
                         |- KITCHEN -----|
```

The content of the loop now consists of the

- List of optional constants that indicate rooms
- Required variable <color>

The bridge value is 5, and the return character is a comma. Given this railroad diagram, some valid PAINT commands would be

```
PAINT THE BEDROOM PINK

PAINT THE LIVING ROOM BLUE, DINING ROOM GREEN, KITCHEN YELLOW

PAINT BEDROOM GREEN, KITCHEN BLUE
```

and so on.

Example 3. Another Loop

In some bridges an asterisk follows the number. For example,

```
                         |<-/4*\--------- , ---------|
                         |                            |
    -- PAINT --------------- LIVING ROOM ---<color>----|
               |       |   |             |
               |- THE -|   |- DINING ROOM -|
                           |             |
                           |- BEDROOM -----|
                           |             |
                           |- BATHROOM ----|
                           |             |
                           |- KITCHEN -----|
```

The asterisk means you must take the right-to-left path at least once. You cannot, for example, enter PAINT BEDROOM BLUE; you must tell the painter at least two rooms to paint. The maximum number of rooms to be painted is still five: the first time through the loop with up to four repetitions.

A valid form of the command would be

```
    PAINT BEDROOM BLUE, KITCHEN YELLOW
```

Example 4. Another Use of the Bridge

A bridge can also control the number of times you take an individual path within a loop. For example, another command to the painter might be:

```
    -- WORK -----------------------------|
             |                          |
             | |<-------------------| |
             | |                    | |
             |-----/1\- EVENINGS -----|
                 |                |
                 |-/1\- WEEKENDS -|
                 |                |
                 |-/1\- HOLIDAYS -|
```

Each bridge /1\ indicates you can take that path once or not at all. That is, you can enter each of the items EVENINGS, WEEKENDS, and HOLIDAYS once at most. Some valid commands are

```
WORK EVENINGS WEEKENDS HOLIDAYS

WORK WEEKENDS

WORK HOLIDAYS EVENINGS
```

but

```
WORK EVENINGS EVENINGS
```

is invalid.

A FINAL WORD

To familiarize you with railroad diagrams, this explanation describes the elements of the diagrams and gives a few simplified examples. Some of the actual diagrams you will encounter in a book may be considerably more complex.

However, the principles are the same no matter how complex the diagram. The more you work with railroad notation, the more easily you will understand even the most complex diagrams.

GLOSSARY

accesscode

An identification code that can be required when logging on to a Message Control System (MCS). An associated password is sometimes required with the accesscode. An accesscode is subordinate to a usercode/password combination and is used as a second log-on procedure.

Actual Segment Descriptor (ASD)

A descriptor that points to the location of a data or code item in memory or in disk for an ASD system.

address space

A division of memory containing all the memory (including code and data) accessible by a task. An address space is identified by an Address Space Number (ASN) and is composed of the shared (or global) memory component and a local component.

Address Space Number (ASN)

A term that refers to an address space or a similar type of memory organization.

ADDS

See "Advanced Data Dictionary System."

Advanced Data Dictionary System (ADDS)

A tool for the definition, storage, and retrieval of data definitions for database and application programs.

application software

Programs written to provide specific functions to end-users and to solve specific end-user problems.

ASD

See "Actual Segment Descriptor."

ASD table

A memory-resident table that contains Actual Segment Descriptors (ASDs). One ASD entry exists for each touched descriptor.

ASN

See "Address Space Number."

bit

The most basic unit of data representation. A bit can be in one of two states: OFF, corresponding to a value of binary 0, or ON, corresponding to a value of binary 1.

BNA

See "Burroughs Network Architecture."

Burroughs Network Architecture (BNA)

The network architecture used on A Series systems to connect multiple, independent Burroughs computer systems into a network. BNA gives the user the same type of access to remote resources as he or she has to local resources.

byte

One character, equal to eight consecutive bits.

CANDE

See "Command AND Edit language."

cold start

A procedure used during system initialization that places the Master Control Program (MCP) on disk and in memory. The system configuration is also defined to the MCP at this time. A cold start removes all files residing on disk.

Command AND Edit (CANDE) language

A time-sharing Message Control System (MCS) for communication and program creation and execution.

Communications Management System (COMS)

A general Message Control System (MCS) that supports a network of users and provides them with a consistent, on-line interface to the system.

compiler

A program that translates instructions written in a source language, such as COBOL, into machine-executable object code.

COMS

See "Communications Management System."

configuration file

A file that lists and describes the hardware resources that make up a configuration. The file can contain descriptions of several different hardware configurations for a system.

cool start

Causes the system to load a new Master Control Program (MCP) onto the Halt/Load unit, update existing MCP tables, and perform a Halt/Load using the new MCP.

core-to-core overlay

A process the system uses to free memory space for program execution. To make enough space, the system moves code and data within memory, overlaying existing code as necessary.

data communications (datacomm)

The transfer of data between a data source and a data sink (two computers, or a computer and a terminal) by way of one or more data links, according to appropriate protocols.

Data Communications Controller (DCC)

The subset of the Master Control Program (MCP), operating as a group of independent tasks, each associated with one Network Support Processor (NSP) that is the basic interface between the data communications subsystem and the main system.

Data Communications Data Link Processor (DC-DLP)

A data communications processor that combines the functions of the Network Support Processor (NSP) and Line Support Processor (LSP) into one physical Data Link Processor (DLP).

data descriptor

A one-word item usually located in a stack and is used to point to an array of data.

Data Link Processor (DLP)

A device that controls the flow of data between the system memory and a peripheral or peripheral controller.

Data Management System II (DMSII)

A specialized system software package used to describe and access information in a database and to maintain relationships between data elements.

Data Transfer System (DTS)

A software system that provides for the exchange of application data between B 20 workstations and Burroughs A Series and B 5000/B 6000/B 7000 Series systems.

DataComm Processor (DCP)

On Multiplexor (MPX) and Input/Output Module (IOM) systems (B 6800, B 7700, and B 7800 systems), the processor that executes instructions compiled from a Network Definition Language (NDL) source program to control the data communications network.

DCC

See "Data Communications Controller."

DC-DLP

See "Data Communications Data Link Processor."

DCP

See "DataComm Processor."

directory

A table of contents listing the files contained on a device. The device is usually a disk or a tape.

disk

A data storage device consisting of one or more circular platters that contain magnetic bits of information stored in concentric circles called tracks.

disk drive

The hardware device on which a disk is mounted. The disk drive has read/write heads to access the data on the disk so the data can be used by the system.

disk pack

A disk consisting of multiple platters stacked vertically on a central spindle. Data on a disk pack is accessed by movable read/write heads. Some disk packs are removable.

DLP

See "Data Link Processor."

DMSII

See "Data Management System II."

ERGO

See "Extended Retrieval with Graphic Output."

Extended Retrieval with Graphic Output (ERGO)

A tool for database access and report generation.

family

One or more peripheral devices that are logically grouped together and treated as a single entity by the system.

FAST

See "File Access Structure Table."

FCF

See "Field Communication Form."

field

An item within a larger entity that represents a logical piece of data. Examples are a consecutive group of bytes within a record that contains a person's name or employee number, or a consecutive group of bits within a word.

Field Communication Form (FCF)

A six-part form used by Burroughs customers to report problems and suggest new features and improvements to software, hardware, and documentation.

Field Trouble Report (FTR)

An FCF used by Burroughs customers to report a problem.

File Access Structure Table (FAST)

A special file that is part of the access structure the system uses to locate disk files. The FAST contains a pointer to each disk file's header in the flat directory of each family.

file attributes

Parameters that describe the characteristics of a file and contain all the information the system needs to handle the file. Examples of file attributes are file name, device type, record size, number of areas, and date of creation.

firmware

Programs that are the instruction sets for the computer hardware, especially device controllers.

FTR

See "Field Trouble Report."

gigabyte

One billion bytes of memory.

Halt/Load

A procedure used to momentarily stop the system and reload the Master Control Program (MCP) into memory.

Halt/Load unit

The disk drive that contains the currently used Master Control Program (MCP). The system loads the MCP from this disk drive at the next Halt/Load operation.

hexadecimal digit

A number stored in four consecutive bits.

IDC

See "Interactive Datacomm Configurator."

IMG

See "Interactive Menugraph Generator."

initialization (system)

The process of loading the necessary software and firmware to ready the system for use.

input file

A file used or read by an executing program.

Input/Output Subsystem

The hardware and software that manages all transfers of information between memory and peripheral devices.

Interactive Datacomm Configurator (IDC)

An interactive program for defining and maintaining the configuration of a data communications subsystem.

Interactive Menugraph Generator (IMG)

A software tool for the design and modification of screen menus and forms.

job

A specified task or group of tasks assigned a number and treated as a unit of work by the system.

job queue

A list of jobs or tasks waiting to be processed.

library

A collection of named, related routines, such as data conversion routines, stored in a code file and available for use by programs.

line printer

A peripheral device that prints text one line at a time, in contrast to a page-at-a-time printer. The system printer is usually a line printer.

Line Support Processor (LSP)

On Message-Level Interface Processor (MLIP) systems, the data communications subsystem processor that manages communication with the host and initiates processes that control the input of messages to and output of messages from data communications lines.

log file

A special type of file that contains a record of system activity, including system utilization, messages, and peripheral activity.

logical unit number

An internal index number used by the Master Control Program (MCP) to identify a peripheral device. The number is actually the position of information about the device in the various system tables containing peripheral information. The logical unit number can change from Halt/Load to Halt/Load as the configuration changes and the tables are rebuilt.

LSP

See "Line Support Processor."

Maintenance Subsystem

The software and hardware that serve as the interface between the user and the Master Control Program (MCP) when the hardware is to be initialized, configured, or halted, or when system software is to be initialized for the first time. The subsystem also is the means to diagnose hardware and software problems.

MARC

See "Menu-Assisted Resource Control."

Master Control Program (MCP)

An operating system for Burroughs computers. The MCP controls the operational environment of the system.

Master Control Program/Advanced Systems (MCP/AS)

An operating system for Burroughs A Series systems. MCP/AS can address up to 4 gigawords (24 billion bytes) of memory, as well as provide all the essential features of the standard MCP.

MCP

See "Master Control Program."

MCP/AS

See "Master Control Program/Advanced Systems."

MCS

See "Message Control System."

media

A device used to store data, such as a disk pack or a magnetic tape.

memory

A temporary storage area where data and programs are placed while being processed.

memory dump

A copy of the contents of memory, often referred to as a memory image. A memory dump occurs when there is a problem with the system and is used to analyze the problem.

Memory Subsystem

The subsystem that handles all transfers of data between main memory and the main processor. It consists of one or more memory control units and memory storage units, plus a memory interface, which may be part of the control unit.

Menu-Assisted Resource Control (MARC)

A menu-driven interface and transaction processor for users and operators of Burroughs systems.

Message Control System (MCS)

A software system that controls the flow of messages between terminals, application programs, and the Master Control Program (MCP). In this context, a message is a transmitted series of words or symbols intended to convey information to or from the system.

Message-Level Interface (MLI)

The interface between the host system and the peripheral subsystem.

Mirrored Disk

The parallel functioning of two to four disks packs so that they are exact copies of each other.

mix

The jobs that are currently executing.

mix number

The number by which a task or job is referenced while it is executing.

MLI

See "Message-Level Interface."

MLS

See "MultiLingual System."

modem

A device that converts data from a form compatible with data processing (digital) to a form suitable for transmission over a communications link (analog), and vice versa.

monolithic memory

A memory structure in which memory is seen by hardware and software as a single, unified area.

MultiLingual System (MLS)

A system for developing and accessing output messages, on-line help text, and menu screens in multiple human languages, such as English and Spanish.

multiprocessing

Two or more processors in the same system running under the control of a single Master Control Program (MCP).

multiprogramming

The ability of a system to run many jobs simultaneously.

NDLII

See "Network Definition Language II."

Network Definition Language II (NDLII)

A language used to define data communications networks. NDLII applies to B 5900, B 6900, and A Series systems.

Network Support Processor (NSP)

A data communications subsystem processor that controls the interface between a host system and the datacomm peripherals. It executes the code generated by the Network Definition Language (NDLII) compiler for line control and editor procedures.

New Programming Language (NEWP)

A member of the ALGOL family of languages. NEWP is used for writing the operating system (MCP), some libraries, and utility programs.

NEWP

See "New Programming Language."

NSP

See "Network Support Processor."

object code

Program code that can be executed. Object code is the result of compiling source code.

ODT

See "Operator Display Terminal."

Operator Display Terminal (ODT)

The terminal used to communicate directly with the Master Control Program (MCP) or the maintenance processor.

output file

A file created or written to by an executing program.

Pack Access Structure Table (PAST)

A table used by the system to locate disk families. It contains pointers into the File Access Structure Table (FAST) that indicate where the entries for a family's files are stored.

password

One of a list of names associated with a usercode or accesscode that identifies the user as a valid user. A password may be required when logging on to a Message Control System (MCS).

PAST

See "Pack Access Structure Table."

peripheral device

A hardware device used for input, output, or file storage. Examples are magnetic tape drives, printers, and disk drives.

physical unit number

The permanent identification for a device by which it is known to the system.

Print System (PrintS)

The part of the Master Control Program (MCP) and related system software that deals with the output of files to printing devices such as image printers and line printers.

PrintS

See "Print System."

QLA

See "Quad Line Adaptor."

Quad Line Adaptor (QLA)

A set of four line adaptors on one circuit board.

queue

See "job queue."

record

A group of logically-related items of data within a file that are treated as a unit by the I/O subsystem. A record can also be defined as the amount of data read from or written to a file in one execution of a read or write statement in a program.

remote device

An I/O unit or other piece of equipment that is physically removed from the computer center but connected by a communication line.

remote file

A file that serves as a means of passing information between a program and a remote device, such as a terminal. Most interactive programs use a remote file, writing data to it and receiving data from it as though it were a local peripheral.

Remote Print System (ReprintS)

The part of the Print System (PrintS) that allows integrated control over printing at remote destinations connected to the host computer through datacomm lines.

ReprintS

See "Remote Print System."

schedule queue

A list of all jobs waiting to enter the mix.

Screen Design Facility (SDF)

A tool for creating screens for on-line, transaction-based application systems.

SDF

See "Screen Design Facility."

sector

The physical units that disks are divided into. On A Series systems, a sector is 30 words, or 180 bytes long. Sectors are also referred to as segments.

segment

The physical units that disks are divided into. On A Series systems, a sector is 30 words, or 180 bytes long. Segments are also referred to as sectors.

segment descriptor

A one-word item used to point to a segment of code.

SMFII

See "System Management Facility II."

source code

A program file containing instructions written in a programming language. Source code must be translated (compiled) to object code before the program can be executed.

spooling

The process of indirectly sending output files to relatively slow peripheral devices such as printers. The output file is logically written to the peripheral device, but it actually goes to a tape or disk file known as a backup file. The backup file is sent to the printer when the job finishes. Spooling allows many users to efficiently share peripherals by preventing any one job from monopolizing the peripheral.

SSF

See "System Software Facility."

stack

A set of sequential locations in memory assigned to a task for the duration of its execution. These locations serve as temporary storage for variables, descriptors, and information pertaining to the task execution.

System Management Facility II (SMFII)

A software system that monitors and provides data on four areas of system performance: hardware, software, workload characterization, and system utilization.

system software

The Master Control Program (MCP) and all other files that are necessary for system operation.

System Software Facility (SSF)

The system software essential for the operation of the system.

tape drive

An input/output peripheral device that stores data on reels or cartridges of magnetic tape.

task

A single, complete unit of work performed by the system, such as compiling or executing a program or copying a file from one disk to another. Tasks are initiated by a job, by another task, or directly by a user.

thrashing

The state the system is in when the amount of physical memory required for effective execution of currently running tasks exceeds available memory. Under this condition, the system is constantly servicing presence-bit interrupts, because the tasks are continually referencing nonpresent memory segments. Thrashing is caused by tasks that have inaccurate working-set estimates. (A working set is the amount of physical memory required to run a task effectively.)

track

One of the concentric circles on the surface of a magnetic disk on which data is stored.

unit

An individual peripheral device such as a line printer.

unit number

The three-digit number assigned by an installation to a particular peripheral device and used to identify the device.

usercode

An identification code used to establish user identity, control system and file security, and segregate files. Usercodes can be applied to every task, job, session, and file on the system.

utility program

A part of the system software that performs commonly used functions or basic data handling operations.

virtual memory

A system technique that treats disk storage space as an extension of main memory, giving the appearance of a larger main memory than actually exists.

volume

A disk, disk pack, or tape reel.

volume label

The area on a disk or tape that stores the name and serial number assigned to the volume.

wait queue

A list of the jobs in the mix that are waiting for resources or for an event before proceeding.

WFL

See "Work Flow Language."

word

The basic unit of information addressed on A Series systems. On Burroughs systems, one word contains 6 bytes.

Work Flow Language (WFL)

A language used to control the Work Flow Management system, which provides control over the running of tasks.

Index

ABCU, 173
Actual Segment Descriptor (ASD) Memory Manager module, 110
Actual Segment Descriptor (ASD) memory subsystem, 4
  table, 5
Actual Segment Descriptor (ASD) table, 5
Actual Segment Descriptor extended memory, 109
Address Space Number (ASN) memory, 5, 39
ADDS, See Advanced Data Dictionary System (ADDS)
Advanced Data Dictionary System (ADDS), 64
Application software, 23
Area, 91, 151
  AREAS attribute, 151
  AREASIZE attribute, 151
  effect of size, 152
  size, 152
ASD, See Actual Segment Descriptor (ASD) memory subsystem
ASN memory, See Address Space Number (ASN) memory

Backups
  definition, 138
  managing, 138
BARS utility, 105
Batch programs, 33
Billing, 54
BNA, See Burroughs Network Architecture (BNA)
Burroughs Network Architecture (BNA)
  access control, 78
  capabilities, 75
  characteristics, 75
  how it works, 76
  using BNA, 78

Cache memory, 5
CANDE, See Command AND Edit (CANDE) language
Cataloging, 60
Checkerboarding, 152
Checkpoints, 148
Code file compatibility
  between levels, 157
  TARGET option, 157
Cold start, 119
Command AND Edit (CANDE) language
  capabilities, 46
  recovery, 47
  security, 47
  window, 45
Communications Management System (COMS)
  capabilities, 44
  recovery, 45
  security, 45
  Utility, 46

Communications Management System (COMS) (cont.)
  windows, 44
COMS, See Communications Management System (COMS)
COMS Utility, 46
Configuration, 36
Configuration file, 112
Configuration Utility, 113
Configuring the system, 115
Converting from non-Burroughs systems, 116
Cool start, 119
CUBE, 172

Data communications
  DATACOMINFO file, 72
  hardware
    devices, 68
    message flow, 70
  Interactive Datacomm Configurator (IDC), 72
  Network Definition Language II (NDLII), 73
  setting up a system, 128
  software, 73
  subsystem, 68
  utilities, 74
Data Communications Data Link Processor (DC-DLP), 70
Data Link Processor (DLP), 6
Data Management System II (DMSII)
  Advanced Data Dictionary System (ADDS), 64
  capabilities, 62
  central storage of data, 64
  data examination, 67
  data retrieval, 63
  database certification, 66
  database creation, 62
  database recovery, 63
  database statistics, 66
  DB MONITOR utility, 66
  DBANALYZER, 67
  DBCERTIFICATION, 66
  DM Interpreter, 65
  Extended Retrieval with Graphic Output (ERGO), 63
  INQUIRY utility, 67
  language interface, 65
  report generation, 63
  structure analysis, 67
Database management, 62
DATACOMINFO file, 72
Datacomm, See Data communications
DB MONITOR utility, 66
DBANALYZER utility, 67
DBCERTIFICATION, 66
DC-DLP, See Data Communications Data Link Processor (DC-DLP)
DCAUDITOR utility, 74

DCSTATUS utility, 74
Deimplementation of system features, 163
Descriptors, 32
Disk directory, 91
Disk families, 91
Disk file naming, 56
Disk subsystem
  areas, 91
  base pack, 91
  components, 90
  continuation pack, 91
  disk, 90
  families, 91
  family index, 92
  File Access Structure Table (FAST), 93
  flat directory, 91
  Pack Access Structure Table (PAST), 93
  rows, 91
  sectors, 91
Distributing files on packs, 126
DLP, See Data Link Processor (DLP)
DM Interpreter, 65
DMSII, See Data Management System II (DMSII)
Documentation, 185, See also Manuals
  catalog, 13
  distribution, 13
  how to order, 203
  on-line, 14
  printed, 13
Documentation conventions, 204
DUMPALL utility, 59
DUMPANALYZER utility, 139

ERGO, See Extended Retrieval with Graphic Output (ERGO)
Extended Retrieval with Graphic Output (ERGO), 63

Families, 91
FCF, See Field Communication Form (FCF)
Field Communication Form (FCF), 174
  filling out, 174
File attributes, 89
File equation, 146
File maintenance, 58, 137
  cataloging, 60
  tape files, 61
File management
  DUMPALL utility, 59
  file names, 56
  maintenance, 58
File names
  construction, 56

File names (cont.)
   disk, 56
FILEDATA utility, 137
Files
   access, 88
   attributes, 89
   controlling changes, 159
   crunching, 152
   distributing on disk families, 158
   distributing on packs, 126
   identification, 88
   logical, 87
   maintaining, 137
   naming, 56
   physical, 87
Filling out a Field Communication Form (FCF), 177
Flashes, 13
Flat directory, 91

GEMCOS, See Generalized Message Control System (GEMCOS)
Generalized Message Control System (GEMCOS), 45
   window, 45
GETSTATUS, 108
GUARDFILE utility, 83

Halt/Load, 119
Hardware
   error detection, 4
   firmware, 4
   input/output (I/O) subsystem, 5
   maintenance subsystem, 6
   memory subsystem, 4
   minimum configuration, 3
   support, 15
      how to get help, 170
   updating equipment, 6
Help text, 14
History of Large Systems, 37
How to order manuals, 203
How to use this manual, ii

I/O blocks, 150
   size, 150
IDC, See Interactive Datacomm Configurator (IDC)
Initialization
   program, 146
   system, 118
Input/output (I/O) subsystem
   buffers, 87
   description, 85

Input/output (I/O) subsystem (cont.)
   file attributes, 89
   files, 87
   logical I/O, 87
   operations, 87
   physical I/O
      devices, 85
      I/O base, 85
INQUIRY utility, 67
Integrating patches, 161
Interactive Datacomm Configurator (IDC), 72
   DATACOMMINFO file, 72
Interactive programs, 33
Intrinsics, 100

Job, 25
Job management
   billing, 54
   controlling jobs and tasks, 53
   job creation, 52
   managing job flow, 135
   mix number, 54
   restarting jobs, 55
   tasking, 54
   Work Flow Language (WFL), 52
Job queues
   capabilities, 53
   setting up, 128
JOBFORMATTER, 54

KEYEDIO, 88

LA, See Line Adaptor (LA)
Languages
   human, See MultiLingual System (MLS)
   programming, 24
      availability, 49
      combining, 51
      standards, 49
Libraries, 34
   General Support, 100
   system, 100
   use, 100, 152
Library maintenance, See File maintenance
Line Adaptor (LA), 70
Line Support Processor (LSP), 70
LOGANALYZER utility, 105
LOGCONSOLIDATOR, 104
LOGGER utility, 105
LSP, See Line Support Processor (LSP)

Magnetic tape files, 61
MAKEUSER Utility, 81
Manuals
  by subject, 185
    data communications, 186
    database management, 185
    languages, 186
    Message Control Systems (MCSs), 186
    multilingual capabilities, 187
    operations, 187
    performance monitoring, 187
    peripherals, 187
    programming productivity aids, 188
    progression, 188
    reference cards, 188
    system software, 188
  by title, 189
  catalog, 203
  how to order
    internationally, 203
    within the United States, 203
  subscription service, 203
MARC, See Menu-Assisted Resource Control (MARC)
Mark level numbers, 12
Master Control Program/Advanced Systems (MCP/AS), 4
  operating system, 4
MCP/AS, See Master Control Program/Advanced Systems (MCP/AS)
Memory dumps
  DUMPANALYZER utility, 139
  handling, 139
  types, 139
Memory factors, 122
Memory partitioning, 110
Memory subsystem, 4
Memory utilization settings, 122
Memory, virtual, 31
Menu-Assisted Resource Control (MARC)
  command mode, 42
  functions, 41
  menu mode, 42
  multilingual capabilities, 42
  on-line help, 42
  tasking mode, 42
  window, 45
Message-Level Interface (MLI), 70
Message-Level Interface Processor (MLIP), 85
Mirrored Disk, 94
Mix number, 54
Mixlimit, 53
MLI, See Message-Level Interface (MLI)
MLIP, See Message-Level Interface Processor (MLIP)
MLS, See MultiLingual System (MLS)

MultiLingual System (MLS)
   capabilities, 131
   setting up, 131
Multiprocessing, 28
Multiprogramming, 27

NDLII, See Network Definition Language II (NDLII)
NDLIIANALYZER utility, 74
Network Definition Language II (NDLII), 73
Network Support Processor (NSP), 70
Networks, 75
New Feature Suggestion (NFS), 174
NFS, See New Feature Suggestion (NFS)
NSP, See Network Support Processor (NSP)
NSPDUMPANALYZER utility, 74

ODT, See Operator Display Terminal (ODT)
On-line documentation, 14
Operational interface
   capabilities, 79
   direct command entry, 79
   Menu-Assisted Resource Control (MARC), 79
   ODT commands, 79
Operations
   configuring the system, 115
   converting from non-Burroughs systems, 116
   progressing from Burroughs systems, 117
Operator Display Terminal (ODT), 79
Ordering manuals, 203

PATCH utility, 159
Patches, 12
   integrating, 161
Patching programs
   controlling changes, 159
   Editor patch mode, 159
   mark field, 160
   SYSTEM/PATCH utility, 159
Performance monitoring
   BARS utility, 105
   GETSTATUS, 108
   LOGANALYZER utility, 105
   LOGGER utility, 105
   MARC menu system, 108
   ODT commands, 107
   operating recommendations, 134
   SETSTATUS, 108
   SUSPENDER utility, 106
   System Management Facility II (SMFII), 102
   SYSTEMSTATUS, 108

Performance monitoring (cont.)
   tools, 102
   Utilization command, 106
Peripheral Test Driver (PTD) testing, 142
Planning daily operations, 132
Port files, 58
Print System
   controlling the Print System, 98
   LTTABLEGEN utility, 99
   PrintS, 97
   spooling, 97
   trainid, 99
Print System (PrintS), 97
   ReprintS, 97
Printer files, 57
PrintS, See Print System (PrintS)
Problem solving
   defining the problem, 166
   Field Communication Form (FCF), 174
   obtaining help
      hardware, 170
      software, 170
   reporting problems, 174
      compiler, 176
      data communication, 175
      data management, 176
      filling out an FCF, 177
      general, 174
Process, 29
Product Support Information Manuals, 13
Productivity software, 22
Program
   conversion, 144
   data conversion, 145
   progression, 143
Program segments, 31
Programming, 143
   code compatibility, 157
   controlling changes, 159
   deimplementing features, 163
   design of programs
      effect of structure, 152
      ongoing maintenance, 154
      resident programs, 153
      structure, 150
      using libraries, 153
   development tools, 155
   file distribution on disk, 158
   generic programs, 157
   integrating new releases, 161
   integrating patches, 161
   language choice, 157
   program initialization, 146

Programming (cont.)
  progression
    data conversion, 145
    manuals, 143
    program conversion, 144
    tools, 143
  recompiling source code, 162
  recovery
    batch programs and jobs, 148
    interactive programs, 147
  supervisor programs, 149
  Test And Debug System (TADS), 160
  testing database programs, 160
  testing programs, 160
Progressing from Burroughs systems, 117
PTD tests, See Peripheral Test Driver (PTD) testing

QLA, See Quad Line Adaptor (QLA)
Quad Line Adaptor (QLA), 70
QUERY utility, 104
Queues, See Job queues

Railroad diagrams
  purpose, 204
  reading diagrams, 204
Recompiling source code, 138
Reconfiguration, 36
Reconfiguring the system, 113
Record keeping, 132
Record size, 150
Release level numbers, 11
Remote files, 58
Remote Print System (ReprintS), 97
Remote Support Center, 15, 170
Reporting problems, 174
ReprintS, See Remote Print System (ReprintS)
Restarting jobs, 55
Rows, 91

SAMPLER, 104
Screen Design Facility (SDF), 64
SDF, See Screen Design Facility (SDF)
Sectors, 91
Security
  compiler, 84
  file access, 82
  network, 83
  privileged programs, 81
  setting up security, 125
  system access, 82

Security (cont.)
   types of control, 80
   user access, 80
      accesscodes, 81
      MAKEUSER Utility, 81
      passwords, 80
      privileged users, 81
      Systemuser, 81
      usercodes, 80
Segments, 31
SETSTATUS, 108
Setting memory factors, 122
Setting system options, 124
Setting up data communications, 128
Setting up job queues, 128
Setting up security, 125
Setting up user identification, 125
Shorthelp text, 14
Site Management System, 104
SITEINPUT, 104
SMFII, See System Management Facility II (SMFII)
Soft configuration, 36
Soft reconfiguration
   configuration file, 112
   Configuration Utility, 113
   FREE and ACQUIRE, 113
   I/O subsystem sharing, 111
   memory partitioning, 110
   on A Series systems, 109
   reconfiguring the system, 113
Software
   application, 23
   organization, 19
   Productivity, 22
   support, 16
      how to get help, 170
   system, 20
Software patches, 12
Source code
   need to recompile, 162
Squashing disk space, 137
SSF, See System Software Facility (SSF)
Stacks, 29
Supervisor programs
   creating, 149
   setting up, 127
Support
   hardware, 15
   software, 16
SUSPENDER utility, 106
System initialization, 118
System Management Facility II (SMFII), 102
   QUERY Utility, 104

System Management Facility II (SMFII) (cont.)
   Site Management System, 104
   System Resource System, 104
System operation, 118
System options, 124
System Resource System, 104
System software, 20
   essential, 8
   optional, 9
   updating, 11
System Software Facility (SSF), 8
SYSTEM/PATCH utility, 159
SYSTEMSTATUS, 108

Tape files
   naming, 57
   supported types, 61
Task, 25
Tasking, 54
Technical Information Papers (TIPS), 13
TIPS, See Technical Information Papers (TIPS)
Trainid, 99

Updating software, 141
User groups, 171
   ABCU, 173
   CUBE, 172
Utilization command, 106

Virtual memory, 31

WFL, See Work Flow Language (WFL)
Work Flow Language (WFL), 52
   job management, 52
   purpose, 51