SERIES 60 (LEVEL 6)

GCOS 6 MOD 400 SYSTEM CONCEPTS ADDENDUM C

SUBJECT

Changes and Additions to the Manual

SPECIAL INSTRUCTIONS

This is the third addendum to CB20, Revision 0, dated January 1978. Addendum A, dated June 1978 and Addendum B, dated November 1978 still apply.

Insert attached pages into the manual according to the collating instructions on the back of the cover. New and changed information is indicated by change bars; information that has been deleted is indicated by asterisks.

Note:

Insert this addendum cover behind the manual cover to indicate that the manual is updated with Addendum C.

SOFTWARE SUPPORTED

When the base manual (CB20, Rev. 0) is updated with Addendum A, Addendum B, and Addendum C, it can be used as reference material supporting either Release 0120 or Release 0121 of the MOD 400 Executive. See the Manual Directory supplied with Addendum C for a list of software products and software reference manuals that can be used with Release 0120 or Release 0121 of the MOD 400 Executive.

ORDER NUMBER

CB20-00C, Rev. 0

July 1979

24007
2679
Printed in U.S.A.

Honeywell

COLLATING INSTRUCTIONS

To update the manual, remove old pages and insert new pages as follows:

Remove	Insert
iii through viii	iii through viii
1-1 through 1-11, blank	1-1 through 1-11, blank
2-1, 2-2	2-1, 2-2
2-7, 2-8	2-7, 2-8
5-9, 5-10	5-9 through 5-15, blank
5-10.1, 5-10.2	—
5-11 through 5-13, blank	—
6-3 through 6-6	6-3 through 6-6
—	6-6.1, blank

File No.: 1S13

7/79
CB20-00C

MANUAL DIRECTORY

Listed below are the various software product items and software reference manuals usable in the MOD 400 Release 0120/0121 operating environment.

The list of software product items includes Release 0120 and Release 0121 of the MOD 400 Executive as well as all separately-priced software product items that can run under control of Release 0120 or Release 0121 of the Executive. Along with each software product item, the directory lists the identity of the appropriate supporting software reference manual(s). Thus, to ascertain the documentation required for any installation, the user need only locate the appropriate product items and note the corresponding manuals.

Software Product Item	*Supporting Documentation*		
	Order No.	*Manual Title*	*Revision Number and Required Addenda (if any)*
MOD 400 Executive (Release 0120 or 0121) (Release 0121 enables certain separately priced Components to send/receive data using LHDLC)	CB01 CB01A	GCOS 6 Program Preparation	Revision 2, dated November 1978 Addendum A, dated January 1979
	CB02	GCOS 6 Commands	Revision 2, dated January 1979
	CB03	GCOS 6 Communications Processing	Revision 2, dated January 1979
	CB05 CB05A	GCOS 6 Data File Organizations and Formats	Revision 2, dated November 1978 Addendum A, dated December 1978
	CB06	GCOS 6 System Messages	Revision 1, dated December 1978
	CB20 CB20A CB20B CB20-00C	GCOS 6 MOD 400 Concepts	Revision 0, dated January 1978 Addendum A, dated June 1978 Addendum B, dated November 1978 Addendum C, dated July 1979
	CB21 CB21A CB21B	GCOS 6 MOD 400 Program Execution and Checkout	Revision 0, dated November 1977 Addendum A, dated June 1978 Addendum B, dated November 1978
	CB22 CB22A	GCOS 6 MOD 400 Programmer's Guide	Revision 0, dated January 1978 Addendum A, dated June 1978
	CB23 CB23A CB23-01B	GCOS 6 MOD 400 System Building	Revision 1, dated June 1978 Addendum A, dated November 1978 Addendum B, dated April 1979
	CB24	GCOS 6 MOD 400 Operator's Guide	Revision 1, dated November 1978
	CB27-01	GCOS 6 MOD 400 Programmer's Pocket Guide	Revision 1, dated March 1979
	CB28-01	GCOS 6 MOD 400 Master Index	Revision 1, dated April 1979
	CD46	Display Formatting and Control	Revision 0, dated November 1978
Sort/Merge (Release 0400)	CB04-01	GCOS 6 Sort/Merge	Revision 1, dated January, 1979
Assembler/Macro Preprocessor (Release 0110)	CB07 CB07A	GCOS 6 Assembly Language Reference	Revision 1, dated June 1978 Addendum A, dated November 1978
	CB08	GCOS 6 System Services Macro Calls	Revision 2, dated December 1978
RPG (Release 0301)	CB09 CB09A	GCOS 6 RPG Reference	Revision 0, dated January 1978 Addendum A, dated June 1978

Software Product Item	*Supporting Documentation*		
	Order Number	*Manual Title*	*Revision Number and Required Addenda (if any)*
Intermediate COBOL (Release 0210)	CB10 CB10A	GCOS 6 Intermediate COBOL Reference	Revision 1, dated June 1978 Addendum A, dated November 1978
Entry-Level COBOL (Release 0205)	CB12	GCOS 6 Entry-Level COBOL Reference	Revision 0, dated November 1978
FORTRAN (Release 0200)	CB13	GCOS 6 FORTRAN Reference	Revision 0, dated November 1978
Advanced COBOL Release 0100)	CB14-00	GCOS 6 Advanced COBOL Reference	Revision 0, dated March 1979
	CB15-00	GCOS 6 COBOL Pocket Guide	Revision 0, dated June 1979
Remote Batch Facility/66 (Release 0300) (When used in conjunction with Release 0121 of the Executive, data can be sent/ received using LHDLC)	CB30 CB30A CB30B	Remote Batch Facility User's Guide	Revision 0, dated January 1978 Addendum A, dated June 1978 Addendum B, dated November 1978
Data Entry Facility (Release 0300)	CB31 CB31A CB32	Data Entry Facility User's Guide Data Entry Facility Operator's Quick Reference Guide	Revision 1, dated June 1978 Addendum A, dated November 1978 Revision 1, dated November 1978
File Transmission Facility – Honeywell Host (Release 0300/ 0301) (When Release 0301 of the FTF is used with Release 0121 of the MOD 400 Executive, data can be sent/received using LDHLC)	CB33 CB33A	Level 6/Level 6 File Transmission Facility User's Guide	Revision 1, dated June 1978 Addendum A, dated November 1978
	CB34	Level 6/Level 62 File Transmission Facility User's Guide	Revision 1, dated November 1978
	CB35 CB35A CB35B	Level 6/Level 64 (Native) File Transmission Facility Users Guide	Revision 0, dated January 1978 Addendum A, dated June 1978 Addendum B, dated November 1978
	CB36 CB36A	Level 6/Level 66 File Transmission Facility User's Guide	Revision 1, dated June 1978 Addendum A, dated November 1978
	CB37 CB37A	Level 6/Series 200/ 2000 File Transmission Facility User's Guide	Revision 0, dated January 1978 Addendum A, dated November 1978

Software Product Item	*Supporting Documentation*		
	Order Number	*Manual Title*	*Revision Number and Required Addenda (if any)*
	CB39 CB39A	Level 6/Level 64 (Emulator) File Transmission Facility User's Guide	Revision 0, dated January 1978 Addendum A, dated June 1978
File Transmission Facility – Non-Honeywell Host (Release 0201)	CB38 CB38A	Level 6/BSC 2780/ 3780 File Transmission Facility User's Guide	Revision 1, dated June 1978 Addendum A, dated November 1978
2780/3780 Workstation Facility (Release 0200)	CB40	2780/3780 Workstation Facility User's Guide	Revision 1, dated November 1978
HASP Workstation Facility (Release 0200)	CB41	HASP Workstation Facility User's Guide	Revision 1, dated November 1978
Host Resident Facility (PDS – Release 0200 FORTRAN – Release 0200 COBOL – Release 0205)	CB42	Level 66 Host Resident Facility User's Guide	Revision 1, dated November 1978
Interactive Function (Release 0100/ 0101) (When Release 0101 of the IF is used with Release 0121 of the Executive, data can be sent/ received using LHDLC)	CB44 CB44-00A	Interactive Function User's Guide	Revision 0, dated November 1978 Addendum A, dated May 1979
Remote Batch Facility/64 (Release 0100)	CF11-00	Remote Batch Facility/64 User's Guide	Revision 0, dated March 1979

The following hardware manuals provide supplementary information:

CC71 – Series 60 (Level 6) Minicomputer Systems Handbook

AT97 – Honeywell Level 6 Communications Handbook

Additionally, certain software applications packages – such as INFO 6, TOTAL 6, and TPS 6 – can also run under control of the MOD 400 Executive. See your Honeywell representative concerning the availability of applications software.

CONTENTS

ILLUSTRATIONS

TABLES

SECTION 1
SYSTEM CHARACTERISTICS

GCOS 6 MOD 400 software is a disk-based operating system that supports multitasking, real-time, or data communications applications in one or more online streams. In addition, program development or other batch type applications can be performed concurrently in a single batch stream.

GCOS is a multifunctional system, capable of supporting a variety of processing functions. The user can develop and execute applications software, perform forms data entry, transmit files to other computers, and enter jobs for execution at remote sites. Terminal concentration services are available, as are various facilities for connecting to an IBM host system.

The system can be configured to process different functional applications concurrently. For example, a user can run his own applications, utilize other system functionality such as the data collection capability, and communicate with a host processor, all at the same time.

SOFTWARE FEATURES

The operating system includes Monitor, File System, and data communications facilities as well as an extensive set of program preparation components, utility routines, and debugging aids. Additionally, the operating system supports various software packages for implementing a distributed processing environment. The Monitor supports the execution of user application tasks and provides a set of system services that enables users to control execution of individual tasks and to synchronize multiple tasks with one another and with time-related events. The Monitor controls the loading of user programs and manages requests for available memory. It provides standard system trap handling routines for responding to exception conditions and also allows users to provide their own trap handling routines for user-caused trap conditions.

The File System software provides a multilevel directory tree structure hierarchy as well as an extensive set of logical I/O access methods. It provides device-independent access to any device for sequential files, and direct access to disk files. In addition the File System software automatically manages the space utilization of mounted disk volumes, thus allowing users to create, expand, and release disk files as required by online applications needs.

The operating system offers two levels of communications interface. Remote/local terminals may be accessed through the sequential file interface of file management, or for more direct control of the communications environment, by using the systems physical I/O interface. The communications facility includes line protocol handlers for teleprinter and VIP devices, binary synchronous communications (BSC), and the polled VIP emulator (PVE).

The software includes a powerful and comprehensive set of program preparation components, utilities and debugging aids for applications development, all running under control of the Monitor. Programming languages include assembly language, RPG, FORTRAN, and Entry-Level and Intermediate COBOL. Commercial central processor models and a Scientific Instruction Processor (SIP) or equivalent software simulators are available with the system. The RPG and Intermediate COBOL Compilers generate code for the Commercial Instruction Processors; the FORTRAN Compiler generates SIP code. The Assembler supports both commercial and scientific instructions. A display processing capability is available to Assembly language and Intermediate COBOL programmers. *

[1] A host resident facility is available on Series 60 Level 66 central systems; the host resident facility permits use of Level 66 resources to develop applications for the Level 6.

The system supports various software packages for implementing functional links to other, remote processors. These components interface with the communications software to enable use of the Level 6 in a true distributed processing environment. The various software packages are described below.

- o The file transmission capability supports transmission of files between the Level 6 and Series 60 (Level 6, 62, 64 or 66) or Series 200/2000 processors, or between the Level 6 and non-Honeywell processors.
- o Remote Batch software permits a Level 6 system to be used for job submission and output delivery for one or more Series 60 (Level 66 or 64) or Series 6000 host processors. Local processing (such as program development and user application execution) can occur concurrently with remote batch processing.
- o The Data Entry Facility (DEF) provides a data collection capability that includes creation/modification of forms; formatted data input; validation, extraction and verification of data; and formatted printing of data. The facility supports multiple independent operator display stations that use VIP devices. The interactive function, an optional software component that may be configured with DEF, provides a programmatic capability for interactive communication between multiple terminals and a host processor.
- o A Terminal Concentration Facility permits various types of synchronous and asynchronous terminals to concurrently connect to a Level 6 and have their message traffic concentrated (multiplexed) over one or more links to a host processor. The Terminal Concentration Facility smooths sporadic terminal traffic patterns, reduces the number of modems and cross-country lines required by multi-terminal applications, and improves total reliability.
- o Workstation facility software includes the 2780/3780 Workstation Facility, the 3270 Workstation Facility, and the HASP Workstation Facility. The facilities provide a means of communicating with an IBM 360/370 host system.

OPERATING FEATURES

The operating system supports concurrent execution of multiple tasks running under one or more task groups. Each task group owns the resources necessary for execution of an application program (one or more related tasks). The task group runs independently in its own operating environment while sharing the resources of the operating system.

Multiprogramming can be achieved by defining more than one application task group to be run concurrently. Serial execution of tasks in a task group can be accomplished by stepping through execution of the tasks in sequence; multitasking can be achieved by causing tasks in the group to be executed concurrently.

Multiple online task groups can be run concurrently with a single batch task group. The batch task group (used for program development or a batch-oriented user application) can be rolled out to a disk to obtain additional memory for online applications.

The number of task groups being run is limited only by the amount of memory available. Concurrently executing task groups may occupy independent, dedicated memory areas, or they may contend for space within a memory pool. When one task group is deleted, the released memory is available to other task groups in the same pool. The Monitor allocates memory dynamically from pools and can relocate programs at load time. Once a task group requests execution, it is dispatched according to its assigned priority level. When multiple tasks share a priority level, they are serviced in a round-robin fashion.

Use of the file system by multiple independent users is facilitated by the arrangement of file system entries (directories and files) in a tree-structured hierarchy. Each directory or file is identified by a pathname that indicates the path from the root directory of the hierarchical structure to the particular directory or file. File reference can be simplified through the use of pathnames relative to a working directory that indicates a user's current position in the file system hierarchy. Access to sharable files and devices is controlled by file attributes and concurrency procedures.

SUMMARY OF SYSTEM FEATURES

The GCOS 6 software offers the following capabilities:

- o Provides a multi-user operating system
- o Supports multiple concurrent programming environments, with applications being run in one batch stream and one or more online streams
- o Controls program preparation through the operating system
- o Handles program preparation and execution of user applications concurrently
- o Handles execution of multiple user applications
- o Permits multitask execution within each user application
- o Controls the execution sequence of user applications
- o Supports real-time operations
- o Provides communications support
- o Is time and interrupt driven
- o Allows device independent programming
- o Supports program overlay capability
- o Provides five programming languages: assembly language, FORTRAN, RPG, and two levels of COBOL (entry level and intermediate) *
- o Provides a heirarchical file system with extensive utility support
- o Supports four disk file organizations
- o Supports code sharing via reentrant programs
- o Permits multiple functions that interface with communications facilities to be run concurrently with application development and execution
- o Supports file transmission between the Level 6 and other computers
- o Provides Remote Batch software permitting the Level 6 system to be used for job submission to a host processor
- o Provides the Data Entry Facility, permitting forms creation and data collection
- o Provides a Terminal Concentration Facility, permitting the concentration (multiplexing) of message traffic
- o Provides workstation facilities, permitting communication between the Level 6 and an IBM 360/370 system
- o Supports data sharing via the record locking facility
- o Supports file protection via access control commands
- o Provides a display processing capability

GUIDE TO USING THE MANUAL SET

A guide to the use of the manual set is provided below. Information is tailored for specific classes of users: applications programmers, system programmers, and operators (As used in this guide, the applications programmer writes applications programs; the system programmer configures the system and defines the environment for each application; the operator operates the system from the operator terminal.) Included as a separate subsection is a guide for those who will use the Level 6 in a distributed processing environment.

Applications Programmer's Manual Guide

Figure 1-1 illustrates the suggested sequence for using the manuals. Consult this manual to determine the capabilities and characteristics of the overall system. This manual describes the software facilities, including the file system, and lists the hardware supported. It summarizes system configuration procedures, methods of accessing the system, and the functions of the command environment. The manual directory in the front lists the manuals with their revisions and/or addenda that apply to the specified release of the operating system. To write an application program, begin by using the *Programmer's Guide* manual. It illustrates: (1) various ways to gain access to the system, (2) a sample Editor session, and (3) for application languages, the procedure for performing program preparation and execution. Working

with the small subset of system commands used within examples is a good approach to learning the system command set. This approach for getting started assumes that a system programmer has already configured and started up a suitable application environment.

Through examples, the *Programmer's Guide* illustrates how to use the system facilities. Other manuals provide reference material. The *Program Preparation* manual contains Editor directives (statements) to create and update an application language source unit. For each of the languages, the appropriate language reference manual contains the description of the language statements. If the application uses communications, refer to the *Communications Processing* manual. Read the *Data File Organizations and Formats* manual if you require a better understanding of a language-supported file organization that is to be used in an application or if you must calculate the size of a data file. You can use Monitor macro calls, as described in the *System Service Macro Calls* manual, in assembly language programs. Before your program can be entered for execution, it must be linked as described in the *Program Execution and Checkout* manual.

For program compilation or assembly and execution, the procedures described in the *Programmer's Guide* might be sufficient. To obtain more control over the execution of your program or utilize the system facilities more completely or efficiently, use the commands described in the *Commands* manual. If you wish to use the operator terminal, read the *Operator's Guide* to learn how to use that terminal. In many cases, the description of commands must be supplemented by system concepts described in the *System Concepts* manual. Rather than read all the conceptual material at one time, you may find it more meaningful to refer to it in conjunction with the appropriate reference material. The *Commands* manual also describes the utilities. The Patch, Debug, and Dump utilities are described in the *Program Execution and Checkout* manual. Error messages and return status codes are listed in the *System Messages* manual.

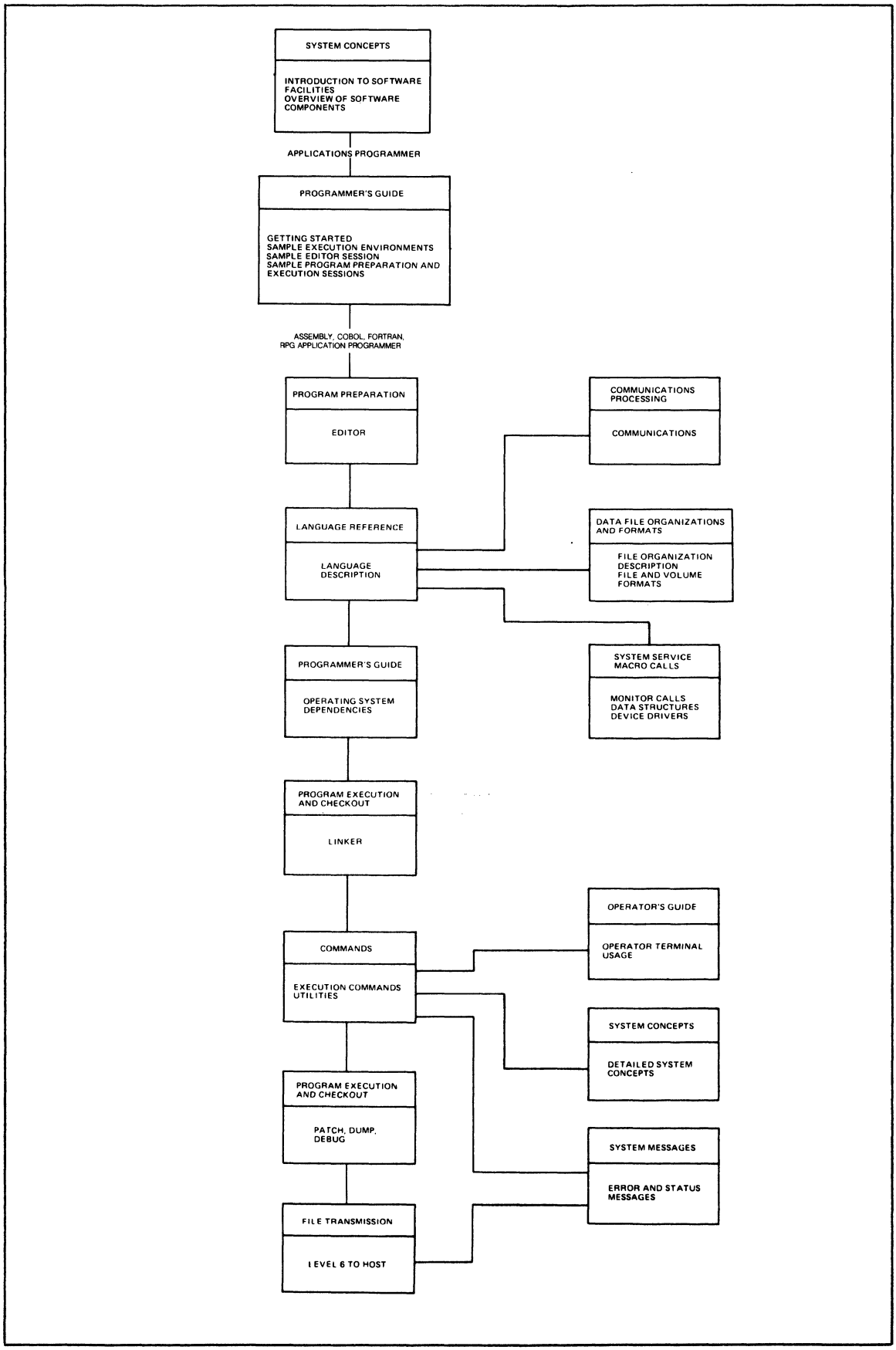

Figure 1-1. Applications Programmer's Guide to Manuals

System Programmer's Manual Guide

Figure 1-2 illustrates the suggested sequence for using the manuals. The *System Building* manual provides you with the configuration directives (statements) and startup procedures to configure and start up a MOD 400 system. You must know the conceptual material in the *System Concepts* manual in order to successfully use the configuration directives. To tailor an applications environment suitable for the intended application, you use operator commands described in the *Operator's Guide* manual. Error messages are listed in the *System Messages* manual.

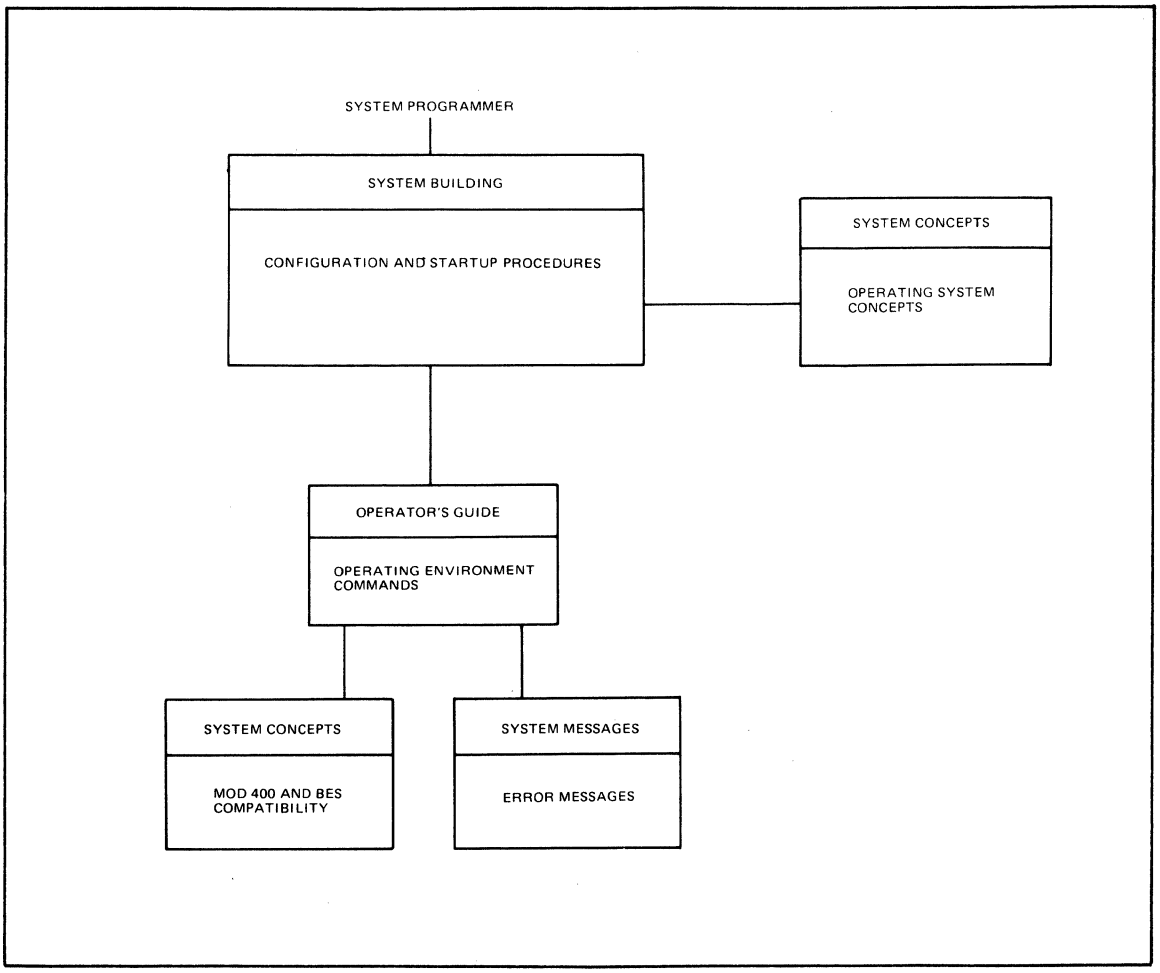

Figure 1-2. System Programmer's Guide to Manuals

Operator's Manual Guide

Figure 1-3 illustrates the suggested sequence for using the manuals. Specific operator job functions must be determined by each installation; a large system might have a person assigned as an operator; a small system might have each programmer also act as an operator. The *Operator's Guide* indicates those system procedures performed through the operator terminal and describes operator commands used in system operation.

The *Programmer's Guide* contains examples using commands (described in the *Commands* manual) that are similar to operator commands. The *System Concepts* manual provides an understanding of the operating system. Note that the *Operator's Guide* describes using the operator terminal for operator functions to enter operator commands to the system task group, or for user functions to enter commands to a user task group. To run the utilities, use the commands (described in the *Commands* manual) entered through the operator terminal functioning as a user terminal. Error messages are listed in the *System Message* manual.

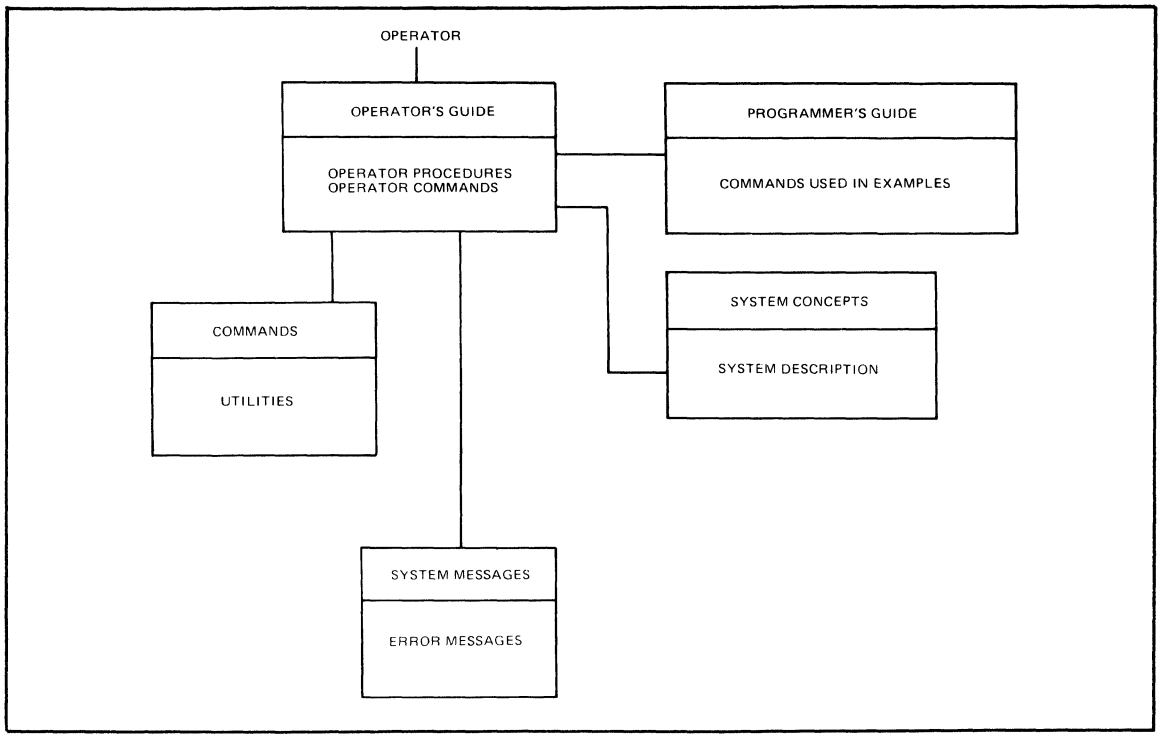

Figure 1-3. Operator's Guide to Manuals

Guide for Using the Manuals in a Distributed Processing Environment

GCOS 6 MOD 400 supports the use of Level 6 in a distributed processing environment. Using Honeywell-supplied software packages, processing capability can be assigned to sites remote to the host computer system. With the functional links provided by Honeywell, a Level 6 can be configured as a host processor and specialized processing (e.g., forms data entry) assigned to remote terminals. Also, the user can develop links with non-Level 6 host processors and distribute the total processing load between the host and Level 6.

The software packages available to the user include the Data Entry Facility, Remote Batch Facility, Terminal Concentration Facility, File Transmission Facility, Host Resident Facility (Level 66), various facilities for connecting to an IBM host system, and a display processing capability. Figure 1-4 indicates the documentation available for the operation and use of such software. Configuration information, if applicable, is contained in the *System Building* manual. Commands and error messages are contained in the *Commands* and *System Messages* manuals respectively.

SOFTWARE DOCUMENT SET

This *System Concepts* manual briefly describes GCOS software, system features, and operating concepts. Most of the background information needed to use the reference material in other manuals of this set is presented in Section 3 through 6 of this manual. Except for summaries, this material is not duplicated in other manuals and covers the following subjects:

- o Task groups and tasking
- o Memory definition and use
- o Operating system features
- o File system and communications concepts
- o Operating environment configuration

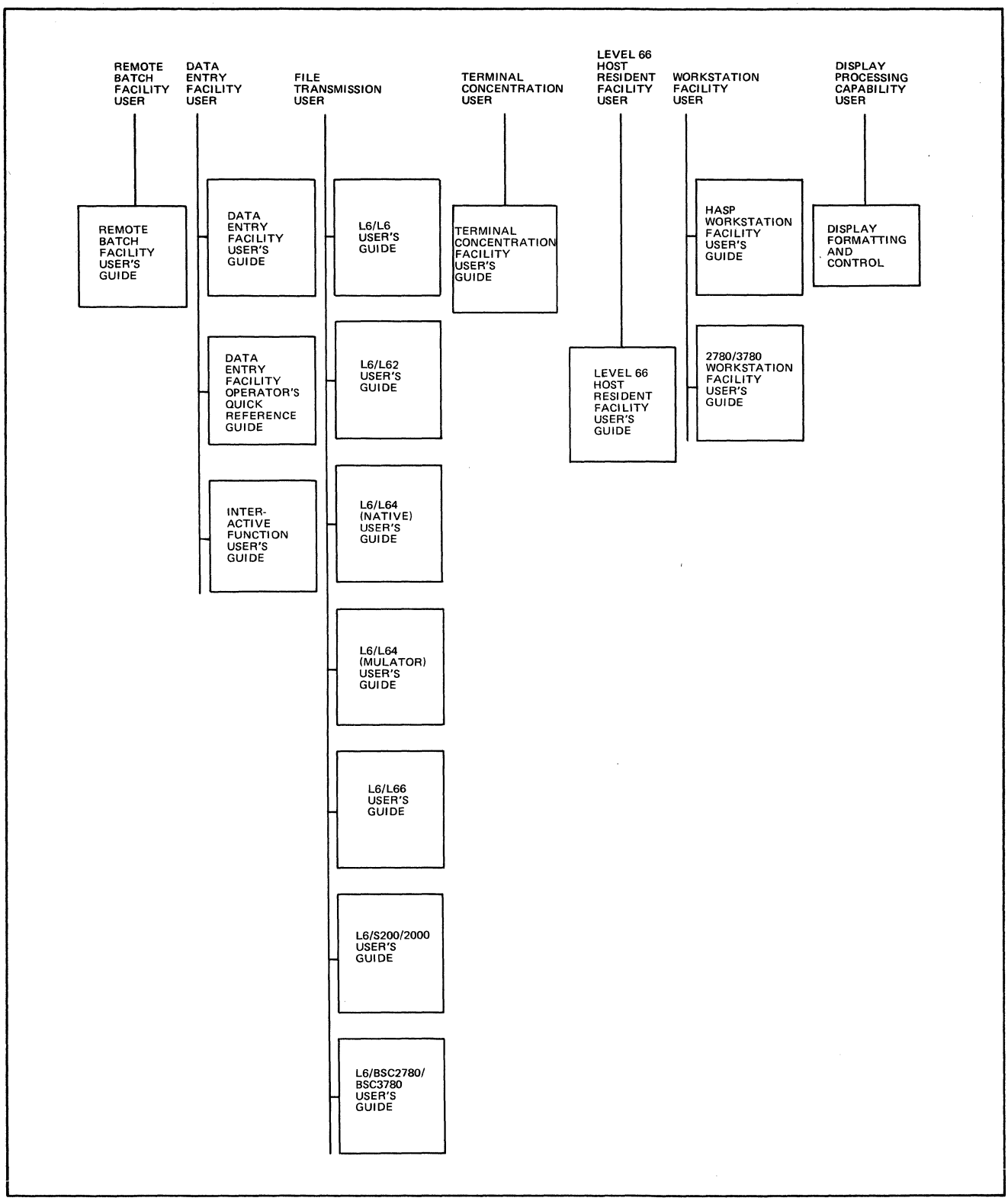

Figure 1-4. Guide for Using the Manuals in a Distributed Processing Environment

Programming conventions are presented in Appendix B of this manual, and a glossary of GCOS 6 MOD 400 terms is in Appendix D.

The contents of other documents in the manual set are summarized briefly below.

- *GCOS 6 Program Preparation*, Order No. CB01 – Overview of the programming steps to prepare a program for execution. Suffix conventions for files used in program preparation. Detailed description of Editor. Rules for writing assembly language programs using SLIC (SAF/LAF independent code).
- *GCOS 6 Commands*, Order No. CB02 – Description of command line format, task interrupt break function, activating an application program, and extending the command set. Detailed description of commands, utilities, and language processor execution. Description of additional command line arguments, terminal characteristics at login, Intersystem Link (ISL) directives, and File Change directives; ASCII and EBCDIC character sets.
- *GCOS 6 Communications Processing*, Order No. CB03 – Introduction to communications software. Description of communications processing through COBOL, assembly language, File System and FORTRAN; sample communications programs; Dump MLCP (DUMCP) utility; TTY, VIP, and BSC control characters; ASCII and EBCDIC character sets.
- *GCOS 6 Sort/Merge*, Order No. CB04 – Description of the Sort and Merge program features, statement formats, and report contents. Includes file and memory requirements, operating procedures, sample programs, using Sort as a subroutine, debug mode execution, and ASCII collating sequence.
- *GCOS 6 Data File Organizations and Formats*, Order No. CB05 – Description of disk and magnetic tape data file organizations support for application programs; disk and magnetic tape record, file, and volume formats; unit record file formats; file and volume headers; ASCII and EBCDIC character sets.
- *GCOS 6 System Messages*, Order No. CB06 – Description of messages reported by system components. Procedure for adding user messages.
- *GCOS 6 Assembly Language Reference*, Order No. CB07 – Complete description of all instructions, instruction formats, control statements, types of data handled, and macro language statements. Description of Scientific Instructions and Commercial Instructions.
- *GCOS 6 System Service Macro Calls*, Order No. CB08 – Description of macro call syntax, register and addressing conventions. Detailed description of system services macro calls for the Monitor and File System and for defining data structures; physical I/O device drivers; Trap Handler; Monitor and File System data structures; writing a user device driver; contents of registers for system service macro calls; ASCII and EBCDIC character set.
- *GCOS 6 RPG Reference*, Order No. CB09 – Complete description of RPG data processing including: a primer on RPG programming, RPG specification form entries, description and use of the RPG fixed logic cycle, and operating instructions with sample programs.
- *GCOS 6 Intermediate COBOL Reference*, Order No. CB10 – Complete description of the general features of Intermediate COBOL programs, language elements, language syntax, the four major divisions of an Intermediate COBOL program, specific format descriptions of all Intermediate COBOL statements (including programming examples incorporating each statement), and the types of files and data handled, ASCII collating sequence, COBOL glossary, comparison of standard COBOL with Intermediate COBOL, and Intermediate COBOL run-time considerations.
- *GCOS 6 Entry-Level COBOL Reference*, Order No. CB12 – Complete description of the general features of Entry-Level COBOL programs, language elements, language syntax, the four major divisions of an Entry-Level COBOL program, specific format descriptions *

of all Entry-Level COBOL statements (including programming examples incorporating each statement), and the types of files and data handled, compiler diagnostics, ASCII collating sequence, COBOL glossary, comparison of standard COBOL with Entry-Level COBOL, and Entry-Level COBOL run-time considerations.

- *GCOS 6 FORTRAN Reference*, Order No. CB13 – Complete description of all statements, instruction formats, types of files and data handled, FORTRAN run-time support routines (intrinsic functions, tasking, I/O), and compiler diagnostics.
- *GCOS 6 MOD 400 Program Execution and Checkout*, Order No. CB21 – Overview of program execution sequence. Detailed descriptions of Linker, Debug, Patch, Dump Memory (MDUMP), Dump Edit (DPEDIT), and interpreting and using memory dumps. Table of system service macro calls ordered by function code.
- *GCOS 6 MOD 400 Programmer's Guide*, Order No. CB22 – Description of various possible programming environments at an installation and the ways to access the system for each environment. Sample Editor session. Examples illustrating how to prepare and execute COBOL, FORTRAN, SORT and assembly language programs; how to call FORTRAN routines from an Entry-Level COBOL main program; and FORTRAN chaining. Explanation of headers on listings.
- *GCOS 6 MOD 400 System Building*, Order No. CB23 – Description of system configuration and startup procedures for the MOD 400 operating system; the software packages for distributed processing (as necessary); configuration directives; system disk layout; system overlays; minimum system hardware and configuration requirements to do program preparation; and startup halts. Description of procedures to transfer files to system disk; create a single-diskette system; place a shared version of a utility in the system library; and load and execute the Intersystem Link (ISL) loader of ISL configuration.
- *GCOS 6 MOD 400 Operator's Guide*, Order No. CB24 – Description of routine system startup, activation of the login capability, system operator interface with the system (OIM), operator commands; task interrupt break function from the operator terminal; additional operator command line arguments; listener component setup for login capability; system halt conditions; ASCII character set.
- *GCOS 6 MOD 400 Programmer's Pocket Guide*, Order No. CB27 – A pocket-size summary of commonly-used commands, directives, and operating procedures as well as a brief description of each coded error message.
- *GCOS 6 MOD 400 Master Index*, Order No. CB28 – An index of specific topics related to GCOS 6 MOD 400. Topics are listed alphabetically. Each topic is keyed to the order number of each manual in which the topic is described.
- *Display Formatting and Control*, Order No. CD46 – Overview of the forms control and manipulation facilities. Includes a description of the forms definition and cataloguing capability and the language elements required for implementation.
- *Remote Batch Facility User's Guide*, Order No. CB30 – Description of remote batch operations: communicating with the host Level 66 central processor, preparing job decks, managing job streams, input and output processing, operator commands and messages, host software control records.
- *Data Entry Facility User's Guide*, Order No. CB31 – Operation of the Data Entry Facility. Description of operation of the Operator Display Station; forms and table development; data entry and verification process; file printing; system supervisory and utility operations; interfacing with data entry and applications programs; and error and system messages.
- *Data Entry Facility Operator's Quick Reference Guide*, Order No. CB32 – A quick reference guide to procedures for operator data entry, data verification, data modification and file printing for the Data Entry Facility.
- *Level 6/Level 6 File Transmission Facility User's Guide*, Order No. CB33
- *Level 6/Level 62 File Transmission Facility User's Guide*, Order No. CB34
- *Level 6/Level 64 (Release 0300) File Transmission Facility User's Guide*, Order No. CB35
- *Level 6/Level 66 File Transmission Facility User's Guide*, Order No. CB36
- *Level 6/Series 200/2000 File Transmission Facility User's Guide*, Order No. CB37
- *Level 6/BSC2780/3780 File Transmission Facility User's Guide*, Order No. CB38
- *Level 6/Level 64 (Release 0220) File Transmission Facility User's Guide*, Order No. CB39 – Each of the above documents describes the capability of the particular file trans-

mission facility, including file organizations supported, code sets, line protocols, and equipment requirements. Individual sections of the manuals provide the operating information necessary to perform a file transfer from either end of a network (Level 6 and host).

- o *2780/3780 Workstation Facility User's Guide*, Order No. CB40
  HASP Workstation Facility User's Guide, Order No. CB41
  Each of the above documents describes the capabilities of the particular workstation emulation facility, including an overview of the facility, the commands and parameter strings to be entered at the workstation, and programming information required to interface with the IBM host system.
- o *Level 66 Host Resident Facility User's Guide*, Order No. CB42 – A description of those Level 66 software facilities that permit use of Level 66 resources to develop applications for the Level 6.
- o *Terminal Concentration Facility User's Guide*, Order No. CB43 – Description of Terminal Concentrator Operations including startup and shutdown and a description of terminal operations using the concentrator facility.
- o *Remote Batch Facility/64 User's Guide*, Order No. CF11 – Description of the facility for entering remote Level 64 jobs at a Level 6, transmitting them to the Level 64 for scheduling and execution and returning the output to a Level 6 peripheral device.
- o *GCOS 6 Interactive Function User's Guide*, Order No. CB44 – Description of Interactive Function operation including initiation procedures, the assignment of files, the free-format and forms modes for data entry to display stations, and the data entry control codes transmitted by the host to the operator station.

SECTION 2
SOFTWARE FACILITIES

GENERAL FEATURES OF SOFTWARE

The system provides a comprehensive array of software to perform multitasking; real-time and data communications applications; and batch, remote batch and data entry processing. The operating system controls execution of tasks and accessing of external devices and files. A complete set of program preparation software is available to develop and debug programs written in COBOL, FORTRAN, RPG or assembly language. An extensive set of utility programs is provided to support program development and execution, and transmission of files from the Level 6 to other computers. System software components are summarized in Figure 2-1. *

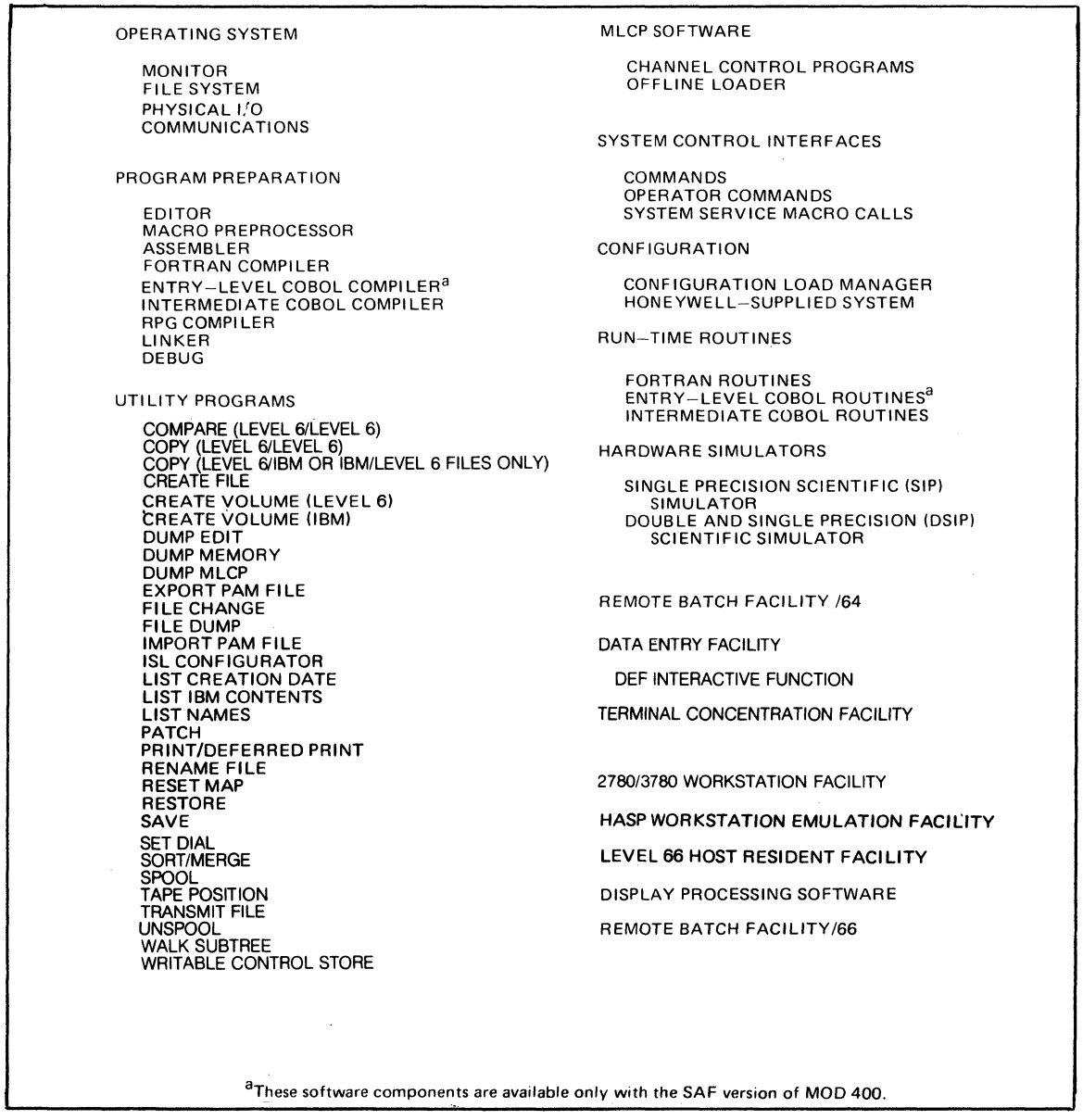

Figure 2-1. GCOS Software

The software is available in a SAF (short address form) version, which supports up to 64K words of memory, and a LAF (long address form) version which supports up to 1 million words of memory. Hardware resources associated with this system are described in Appendix C.

INTERFACES TO OPERATING SYSTEM

The software supports the following control interfaces to the operating system:

- *Commands* submitted by a user to the command processor of the user task group
- *Operator commands* submitted by the operator to the command processor of the system task group
- *System service macro calls*, specified in assembly language programs, that invoke Monitor and file system services for user task groups

Command Language

There are five functional categories of commands:

- To control execution
- To control directories and files
- To invoke program preparation software
- To invoke utility software
- Interactive commands

Some control functions at the task group level are available through commands. Commands are described in the *Commands* manual.

Commands for Execution Control

Once a task group is created, commands written by the user can be executed under the task group. More comprehensive control of execution is provided to the assembly language program through system service macro calls. Commands are used to:

- Create then initiate other task groups, or spawn task groups. This provides a multiprogramming capability.
- Abort or delete a task group, or terminate the task group issuing the request. The abort and delete functions are not available through the batch task group.
- Create then initiate the execution of a sequence of tasks under a task group, or spawn tasks within a group. Using this capability an application can be executed as a sequence of steps. When the sequencing is done so as to have several tasks active simultaneously, there is multitask execution in one task group. A batch task group cannot create another task group.
- Control of external switches for intertask communication.
- List the status of all tasks or open files in a task group.

Commands for Directory and File Control

The file system is based on a tree-structured directory hierarchy. To locate a file, the directory pathname must be known. In order to write programs that are independent of the pathname of the physical file, a program uses a logical file number (LFN). More comprehensive control of directories and files is provided to the assembly language program through system service macro calls.

Commands are used to:

- Create or release a directory or file
- List the pathname of the working directory; change the pathname of the working directory
- Reserve (get) a file for processing (through use of a logical file number).
- List, in the order searched, the directories that are searched for a given pathname; list file entries in a specified disk directory.
- Modify the share, read, or write attributes of a disk file.

- *RPG Compiler* – Translates RPG source statements of a source unit into a set of object units consisting of a root, or a root plus multiple overlays. The compiler also produces a file containing Linker directives; user-written Linker directives are thus unnecessary. When the command processor is invoked to process the statements in this file, it invokes the Linker, and supplies it with Linker directives necessary to create an executable bound unit. The compiler supports an RPG language comparable to that in current industry-wide use. Significant features include: look-ahead, control levels and matching fields on input; table and array processing; forms alignment; and editing, detail, and total time functions on output. The compiler generates Commercial Instruction code. A description of the RPG language is found in the *RPG Reference* manual.
- *Linker* – Combines object units that are the output of a compiler or the Assembler and produces a bound unit for subsequent loading. It resolves external references made between object units being linked. Linker directives can be used to create reentrant bound unit files. A description of Linker directives is found in the *Program Execution and Checkout* manual. *
- *Debug* – Used for testing programs at the machine language level. Hexadecimal patches can be made to the program. Debug is invoked as a separate task group within the system. A description of the Debug directives is found in the *Program Execution and Checkout* manual.

UTILITY SOFTWARE

A comprehensive set of utility programs is available to support file management and program development. All utility programs listed below are invoked by commands except for Memory Dump (MDUMP) and Dump MLCP (DUMCP). The usage of the utility programs is described in the *Commands* manual unless otherwise indicated.

- *Compare* – Compares two volumes, files or portions of files for equality, and lists the discrepancies.
- *Copy* – Copies a file or volume. Logically reorganizes the file. Copies can be placed on tape or disk.
- *Copy IBM* – Copies Level 6 diskette files to IBM diskette files (and vice versa).
- *Create IBM Volume* – Formats an IBM 3740 diskette.
- *Create Volume* – Creates or modifies a volume. Formats and labels a disk or tape volume, creates disk bootstrap records, or renames a disk volume.
- *Dprint* – Prints the contents of the specified file. By use of Dprint, the print request is entered in a queue, and the user can continue with other commands or terminate the session (logout).
- *Dump Edit* (DPEDIT) – Produces an edited logical or physical dump image of memory, or edits and prints out a disk file containing a dump of main memory that was obtained through the MDUMP bootstrap record. (Described in the *Program Execution and Checkout* manual.)
- *Dump Memory* (MDUMP) – Dumps the contents of memory to a disk file when a program aborts or halts, by using the bootstrap record MDUMP on a specially created disk volume. The Dump Edit utility is then used to print the dump. (Described in the *Program Execution and Checkout* manual.)
- *Dump MLCP* – Dumps contents of all or part of Multiline Communications Processor (MLCP) memory. (Described in the *Communications Processing* manual.)
- *File Change* – Changes the contents of a disk sector or control interval.
- *File Dump* – Performs both logical and physical dumps from disk or 9-track or 7-track magnetic tape; output in both alphabetic and hexadecimal notation.
- *Import/Export PAM File* – Converts members of GCOS/BES partitioned files to and from GCOS 6 variable sequential files; used to transport programs between BES and GCOS 6.

- *ISL Configurator* – Reads ISL (Intersystem Link) directives from a user input file and generates an ISL loader.
- *List Creation Date* – Lists creation dates of files in a directory.
- *List IBM Contents* – Lists the contents of an IBM diskette.
- *List Names* – Lists the file and/or directory entries contained within the specified directory.
- *Patch* – Applies hexadecimal patches to an object unit or bound unit. Provides a facility for program correction without recompilation or reassembling. (Described in the *Program Execution and Checkout* manual.)
- *Print* – Prints the contents of the specified file. The file may contain FORTRAN control characters, standard print control characters, or no print control characters.
- *Rename File* – Assigns a new name to an existing file or directory.
- *Reset Map* – Lists the number of logical sectors available for allocation on a disk volume.
- *Restore* – Restores the file structure and physically reorganizes the disk volume previously saved (by the SAVE command).
- *Save* – Saves the file structure and data of a disk volume onto a tape or disk file.
- *Set Dial* – Sets a phone number for subsequent use by a named channel/communications terminal file.
- *Spool* – Enters print requests into the queues for later unspooling.
- *Transmit File* – Supports file transmission between the Level 6 system and other Level 6 processors, or between the Level 6 and any of the following host processors: Level 62, 64, or 66; Series 200/2000; or non-Honeywell systems that use the BSC 2780/3780 protocol. Three utility programs, described in Section 7, provide the file transmission capability.
- *Unspool* – Transcribes data files to a terminal device.
- *Walk Subtree* – Executes specified command in specified directory and in all subordinate directories.

Sort/Merge

Sort and Merge are invoked by separate commands. Sort may also be called from a COBOL, FORTRAN, or assembly language program. The Sort program arranges records of a file in an order based on the values of user-specified record key fields. Merge combines the records of up to six sequentially ordered input files on the basis of record key values. Up to 16 key fields can be specified, with values to be arranged in ascending or descending order according to the ASCII collating sequence. The data type of a key field can be character string, signed binary, packed decimal, or signed/unsigned unpacked decimal. Sort/Merge options include record selection, redefinition or rearrangement of record contents, and deletion of duplicate records. See the *Sort/Merge* manual for a detailed description of these capabilities.

RUN-TIME ROUTINES

Run-Time I/O Routines

The FORTRAN run-time I/O routines provide for data transfer, peripheral or communications device manipulation, and the processing of data as specified in FORTRAN FORMAT statements. These routines use the file system to accomplish open, close, and position file functions, and to read and write formatted and unformatted records. They contain data conversion routines to edit integer, real, logical, and character data for formatted input and output. Only those routines required by a particular FORTRAN program are linked to form the bound unit.

The COBOL run-time I/O routines provide a logical I/O interface for the transfer and processing of data at program execution time. The routine is linked with the program's object unit, and uses the file system to open, close, and position files and to read and write records to peripheral or communications devices. Separate run-time routines are provided for Entry-Level and Intermediate COBOL.

The FORTRAN and COBOL routines produce diagnostic messages to inform the programmer of inappropriate or inconsistent input/output statements.

The operating system area is fixed – its contents remain the same for the life of the system – in contrast to other memory areas whose contents can vary. Almost all code loaded into this area is reentrant so that a single copy of the code is available to multiple users, thus minimizing memory requirements.

System Pool Area

The area adjacent to the resident software area is called the system pool. This area contains the system task group. In addition, the system pool accommodates the following elements:

- o Current function invoked by an operator command
- o Extended trap save areas (TSAs) needed during processing
- o Control structures for the batch task group
- o Shareable bound units
- o File system directory and file definition blocks

System Task Group

The system task group differs from other task groups in the following ways:

- o Cannot be aborted or suspended
- o Always has read and write access to all of memory
- o Handles all system dialog (including operator commands) through the designated operator terminal
- o Never terminates, so it cannot be requested

Batch Task Group Control Structures

The following control structures are found in the system pool area whenever a batch memory pool is configured:

- o Group control block (GCB)
- o Logical resource table (LRT)
- o Logical file table (LFT)
- o Task control block (TCB)
- o Batch request block

File Control Structures in the System Pool Area

The elements in the system pool area that are used for file control consist of:

- o File description block (FDB)
- o All buffers for shareable files
- o Buffers for shareable files

Pool Attributes

The user can exercise more control over memory usage by providing online memory pools with specialized attributes, as described below.

Protected Memory Pools[1]

A memory pool may be "protected" if it is so specified at configuration (by a CLM command). A protected pool is one into which a task running in another pool may not write. Through use of the Memory Management Unit (MMU), the operating system will prevent write intrusion by foreign tasks. Such a task will receive an error notice from the operating system when an intrusion is attempted.

The special size constraints that apply to protected pools are described in the System Building manual.

---

[1] Applies only to configurations having a Memory Management Unit (MMU).

Contained Memory Pools[1]

Any memory pool but the system memory pool may be "contained" if it is so specified at configuration. Tasks running in a contained pool are prevented from writing outside their own pool area. The constraints that apply to the size of contained memory pools are the same as those that apply to protected memory pools.

Unprivileged Memory Pools

At configuration, any memory pool except the system pool or the batch pool may be declared "unprivileged." A task running in an unprivileged pool cannot execute privileged instructions, and will trap if such an execution is attempted. A memory pool may be declared unprivileged regardless of whether or not the configuration has a Memory Management Unit; i.e., the privileged function is independent of the memory protection function.

The following assembly language instructions are privileged:

ASD	IO	LEV	WDTF
CNFG	IOH	RTCF	WDTN
HLT	IOLD	RTCN	

The system pool is always privileged and the attribute cannot be altered.

The batch pool is always unprivileged and the attribute cannot be altered.

Exclusive and nonexclusive pools are privileged unless specified to be unprivileged.

Serial-Usage Memory Pools

An exclusive or nonexclusive memory pool may be declared serial-usage. If so declared, such a pool may be used by only one task group at a time.

Multi-Pool Memory Protection

On a system having a Memory Management Unit (MMU), a user may specify the write protection/containment that the system is to provide. At CLM time, the user selects one of the following options. The option selected prevails until the system is reconfigured.

*

1. No protection or containment; i.e., no utilization of the MMU.
2. Protection of the system memory pool and/or containment of the batch memory pool.
3. Protection and/or containment of selected memory pools in addition to those of option 2.

Note that protection applies to memory pools and not to task groups. Thus, groups sharing a memory pool are not protected from each other. Nor is the only group in a memory pool secure from intrusion if the pool is a nonexclusive pool.

If a task group is to be protected from all other groups, it must be the only group using an exclusive memory pool, and option 3 software protection must be specified at system configuration.

Memory Layout

To obtain efficient use of memory and of the Memory Management Unit the Configuration Load Manager (CLM) sorts the memory pools in a configuration as follows:

- The system pool is in the first available memory after the system date structures.
- Nonprotected, noncontained pools are next in order of size; the smallest one comes first.
- Protected and/or contained pools come last in order of size; the smallest one comes first.
- All nonexclusive pools are considered to be a single pool, for purposes of memory allocation.

[1] Applies only to configurations having a Memory Management Unit (MMU).

Selecting Memory Pool Attributes for Task Group Execution

The different type memory pools provide system users with the means to respond to the unique demands of multiple application programs. Through the use of memory pools the user can at once exercise control over memory usage and at the same time provide individual task groups with specialized protection attributes.

The degree to which the system can efficiently and effectively handle the concurrent execution of multiple task groups depends on the number and type of memory pools available for use.

Cases 1, 2, and 3, below examine the considerations involved in the selection and use of memory pools.

Case 1:

The user's program consists of a real-time data application program. The program must co-exist with other user applications.

The data application program accepts data based on unpredictable external stimuli. The application permits a "saturation" effect to occur when data collection exceeds the effective rate of processing. The occurrence of "saturation" represents a signal to the application to initiate a data selection strategy.

The user should select an on-line pool of sufficient memory size to control the maximum desired amount of data allowed to accumulate.

Case 2:

The user's application consists of process control software with a periodic need for all of available user memory.

The application requires the total memory of the system only 30 seconds of every five minutes. For the remainder of the time, only 10% of user available memory is required.

In this case the process control application ought to be loaded into an extendable online pool. The size of the online pool should be 10% of available memory. The applications periodic need for all of available memory can be fulfilled by defining a batch pool that consists of 90% of available memory. Whenever the process control application requires more memory than that in its online pool, then the system will initiate a rollout of the batch pool. During the period when the process control application does not require all of available memory, then the use of the batch pool can be obtained by other applications.

Case 3:

Multiple applications with strict integrity requirements are to coexist with programs under development test.

The user can choose to load each of the applications with integrity requirements into a memory pool that has been designated as "protected". This will insure that the programs under test do not accidently modify data in the programs to be protected.

BOUND UNITS

Task code is derived from the source language of programs that are compiled or assembled to form object units. One or more object units are linked to form a bound unit that is placed on a file. The bound unit is an executable program that can be loaded into memory. A task represents the execution of a bound unit. Each bound unit consists of a root segment and any related overlay segments.

Overlays

To minimize the amount of memory required to execute a bound unit containing application code, the bound unit can be created as a root and one or more overlays. Object units whose code is to be loaded as overlays are defined as overlays by the Linker. The use of overlays requires careful planning so that required code is not lost or repetitively loaded.

Nonfloatable and Floatable Overlays

Overlays are part of a bound unit which comprises a root or a root and one or more overlays. There are two types of overlays: the nonfloatable overlay that is loaded into the same

memory location relative to the root each time it is requested, and the floatable overlay that is linked at relative location 0 and can be loaded into any available memory location.

Floatable overlays must have the following characteristics:

- o External location definitions in the overlay are not referred to by the root or any other overlay.
- o The overlay makes no immediate memory addressing (IMA) references to itself, and no displacement references to the root or any other overlay.
- o The overlay can contain IMA references with or without offsets to the root or any other nonfloatable overlay.
- o The overlay does not contain external references that are not resolved by the Linker.
- o The overlay must be linked *after* all nonfloatable overlays have been linked.

A user program can use one or more areas of its available memory for placement of floatable overlays. To be most effective, the program must perform its own memory management of these areas. To perform memory management, a user-written assembly language overlay manager must be linked with the root of the program bound unit. Adequate memory space must also be provided in the pool of the requesting task. If the user program does not control the placement of a floatable overlay, the system will place the overlay in available space in available memory.

Assembly language programs can use system service macro calls to load and execute nonfloatable overlays; memory management is handled by the Monitor. Similarly, COBOL programs can use CALL/CANCEL statements to control nonfloatable overlays. FORTRAN and RPG programs must link a user-written assembly language overlay manager with the application program.

Resolving References

Forward references can be made to symbols defined in object units to be linked later. Backward references can be made to symbols previously defined provided that the defined symbols were not purged from the Linker symbol table by a Linker BASE or PURGE directive. Since the specification of the BASE directive removes from the Linker symbol table all previously defined, unprotected symbols that are at locations equal to or greater than the location designated in the BASE directive, you must either define all symbols in a nonoverlaid part of the root or plan the linking of subsequent overlays so that purging of needed symbols does not occur.

Floatable overlays can refer to fixed addresses in the root or nonfloatable overlay, but cannot refer to addresses in another floatable overlay.

When a root or an overlay of a bound unit is loaded, the loader examines the attribute tables associated with the bound unit, if an alternate entry point is specified. The loader tries to resolve any references to symbols that remain unresolved at load time by searching the system symbol table (i.e., the resident bound unit attribute table); it cannot resolve any references to symbols that do not exist in that table (Linker symbol tables do not exist at load time).

*

Sample Overlay Layout

Figure 5-3 shows the relative location in memory of memory pool AA. Figure 5-4 is the layout of overlays in memory pool AA. The Linker directives to create and specify the location of these overlays are described in the *Program Execution and Checkout* manual.

When the root is loaded, the largest contiguous amount of memory necessary to accommodate the *root* and all *nonfloatable* overlays is allocated. Except for space for any floatable overlays, no other memory requests need be made. In the figure, this memory area begins at relative 0 of the root, and continues to the end of object unit OBJD. The root consists of object units OBJ1 and OBJ2. When loaded, OBJ5 of overlay ABLE will replace the previously loaded OBJ2 code of the root. Similarly, the overlay locations were specified so that OBJC of overlay ZEBRA will replace part of OBJB.

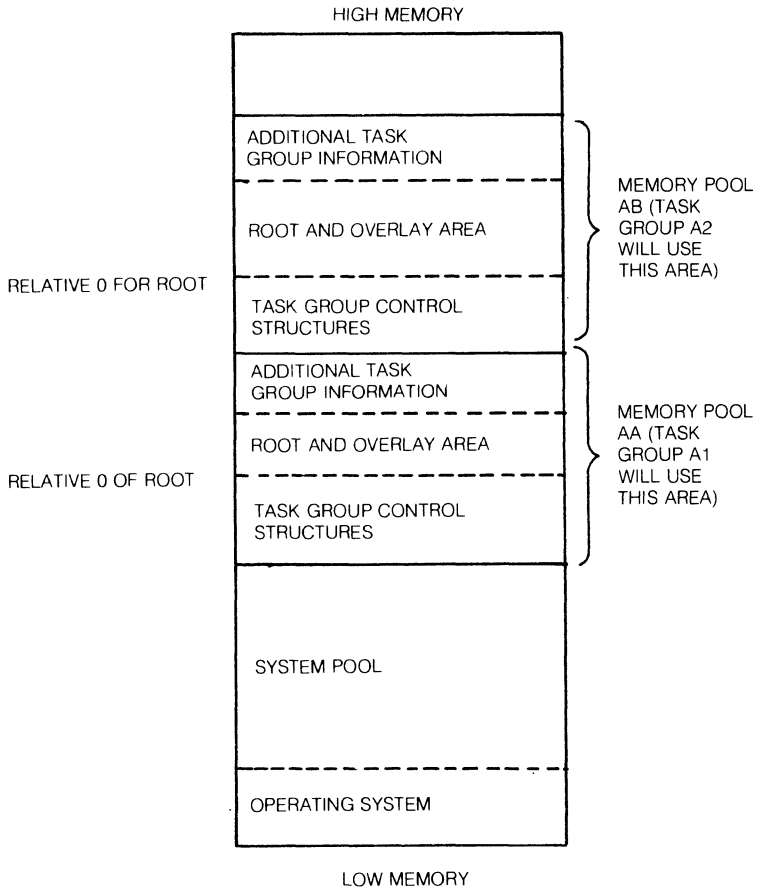

Figure 5-3. Relative Location in Memory of Memory Pool AA

Shareable Bound Units

Using shareable bound units is a way of minimizing application task group memory requirements while making reentrant code available to multiple tasks. Unlike permanently resident bound units that are loaded during system configuration, shareable bound units are transient in the system pool and are loaded during processing. A counter is incremented each time a request is made for the bound unit, and the unit remains in memory as long as a task is using the code. As soon as the counter is decremented to zero, the system pool space occupied by the bound unit is returned to available status.

Operator commands can be used to load and then unload a shareable bound unit.

To be recognized as shareable by the loader and loaded into the system pool, the bound unit must have been so marked by the Linker in response to a SHARE directive when the bound unit was linked. If the system pool does not have enough space to accommodate the bound unit, at a given instant, it will be placed in the pool associated with the task group that requested the bound unit, and cannot be shared.

Shareable bound units and the operating system extensions that are loaded when the system is configured differ in another way. Namely, operating system extensions can be referred to directly by any task, but a shareable bound unit must be accessed as a task. The reason for the difference is that an operating system extension is loaded when the system is configured – its symbols are included in the system symbol table at that time. Since there is no symbol definition once configuration is complete, and since a shareable bound unit is loaded *after* the system has been configured – no entry for it exists in the system symbol table, and it must be accessed as a task.

Table 5-2 compares permanently resident operating system extensions and transient shareable bound units.

TABLE 5-2. COMPARISON OF OPERATING SYSTEM EXTENSIONS AND SHAREABLE BOUND UNITS

Characteristics	Operating System Extension	Shareable Bound Units
Multiple Users	Yes	Yes
Permanent Resident (fixed area)	Yes	No
Temporary Resident (dynamic area)	No	Yes
Symbols in System Table	Yes	No
Accessed symbolically?	Yes	No
Accessed as a task?	Yes[a]	Yes
Can have overlays?	No	Yes
Called by Bound Unit Name	No[b]	Yes

[a]If the extension is an assembly language bound unit, it may have within it sections of code or control structures controlled by semaphores which would be accessible to other assembly language tasks.

[b]The operating system does not "remember" extensions by their names; a request for one by name results in another copy being brought into memory.

Loading Bound Units (Search Rules)

The loader follows preestablished search rules when searching through a set of directories to locate a bound unit to be loaded. The loader initiates the search in response to a command which contains an argument naming the bound unit to be loaded.

Search rules that regulate the search process define three directory pathnames and the sequence in which they are used during a search. They are:

- o The task group's working directory
- o System directory -LIB1 argument of the CSD (change system directory) command
- o System directory -LIB2 argument of the CSD command

At the completion of command processor start-up, the pathname of the system task group's working directory is ^ system volume name, and remains so until modified by one or more CWD (change working directory) commands. Both system directories are initially set to >SYSLIB1. The system STARTUP.EC file executed immediately after configuration should initially be used to change the value of the second system library searched. It would normally be changed to >SYSLIB2 on a cartridge disk or storage module system and ^ZSYS01>SYSLIB1 on a diskette system. The operator command CSD (change system directory) may be used to change pathnames associated with system directory arguments -LIB1 and -LIB2. The pathname of a user task group's working directory is established through a CWD command or through the -WD argument in the EBR (enter batch request), EGR (enter group request), or SG (spawn group) commands.

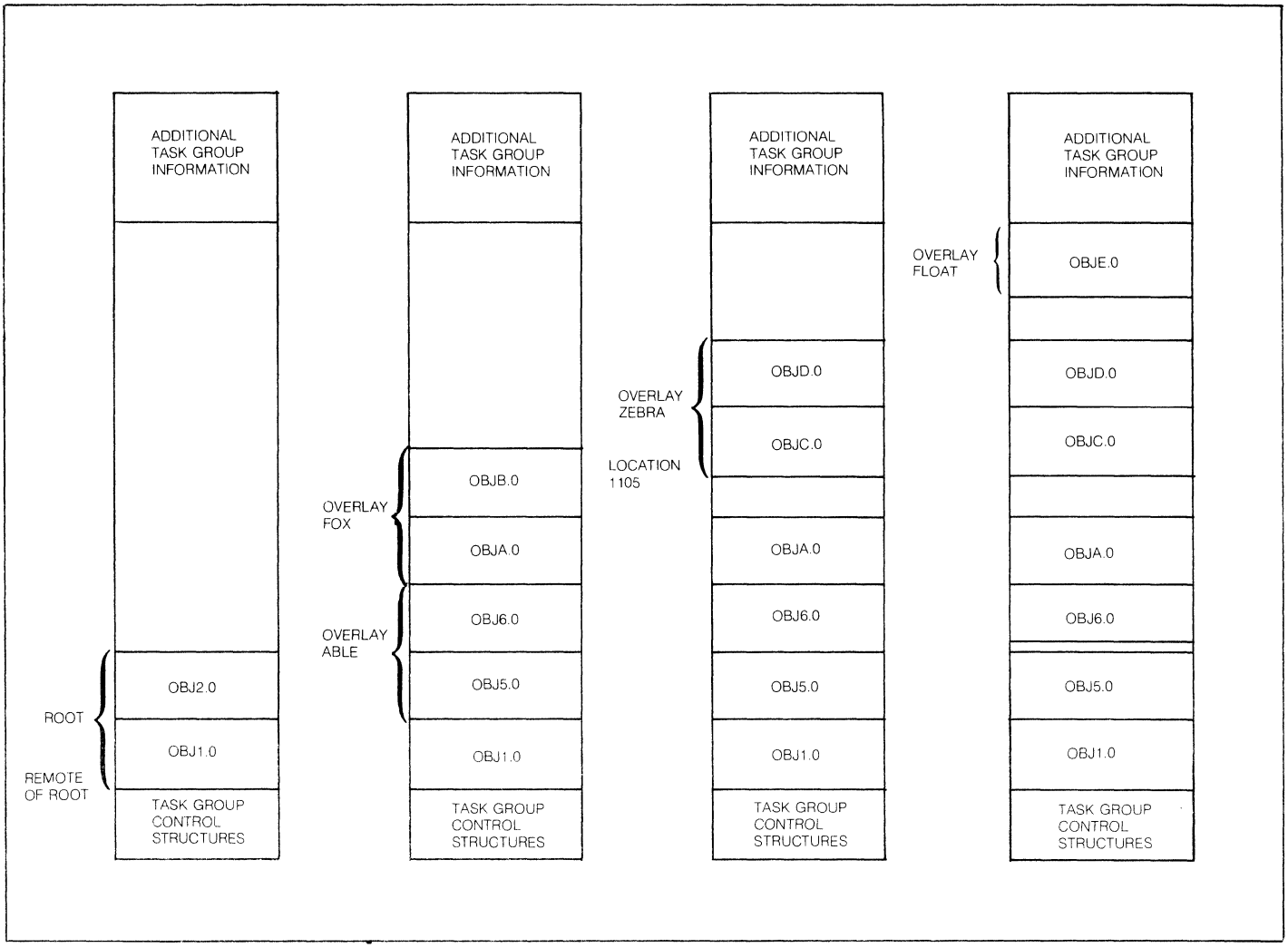

Figure 5-4. Overlays in Memory Pool AA

cription of a firmware state, in this documentation set, the term "suspend" indicates the logical state of a task as described in the paragraph "How the Operating System Handles Tasks."

- o *Activate a Level* - Mark the target priority level as active (set the level's activity indicator); save the context of the current priority level; go into resume state.
- o *Inhibit* - Mark dedicated, high-priority level as active and immediately assume this level. Continue execution on this priority level with no context save or restore.
- o *Enable* - Return to the highest active normal priority level from the inhibit level. If the highest priority level is the same one from which the inhibit level was entered, do not perform a context save or restore.

The number of the highest active priority level can be ascertained by interrogation of the contents of the S-register.

The availability of an inhibit level allows critical sections of code to be executed without danger of interruption. For example, it may be necessary to temporarily assume the inhibit level in order to protect short sequences of instructions that modify data structures shared between levels.

A standard feature of Level 6 central processors is a real-time clock, which interrupts at a preassigned priority level (level 4) each time its specified scan cycle has elapsed.

TRAP HANDLING

The operating system provides a means by which certain events which occur during the execution of a task can be "trapped", or handled by routines which are designed specifically to cover the condition causing the trap. Events such as the detection of a program error, hardware error, arithmetic overflow, or uninstalled optional instruction cause traps, control transfers to designated software routines, to occur.

Traps fall into two classes: standard system traps, for which routines are supplied with the operating system, and user-specific traps, for which the user can supply his own handler.

An application program can designate which traps are to be handled through specification of the enable/disable user trap macro calls (see the *System Service Macro Calls* manual for details). If an enabled trap occurs in the user program, the trap manager transfers control to the connected trap handler for the condition causing the trap. A trap that is enabled is local to a task, that is, it neither affects nor is affected by the handling of the same trap in another task, even within the same task group.

Any trap which occurs when its handler is not enabled, or which does not have a handler to process it, causes the executing task to be aborted.

OPERATING SYSTEM FEATURES AFFECTING TASK EXECUTION

The operating system does not monitor resource use either within a task group or among task groups using the online pools. Tasks and task groups must cooperate in their use of system resources to ensure smooth operation of the application.

Peripheral Device Assignments

DEVICE or DRIVER directives processed by the CLM during configuration identify peripheral devices to the operating system. (Communication devices require an additional device-type-specific directive; see the *System Building* manual for details.) Any online task can use any peripheral device or disk file. A batch task can use any peripheral device or disk file that is marked as shareable. The command MF (MODIFY FILE) can be used to change a peripheral device, a disk directory, or file to nonshareable and back again. MF won't be recognized when issued from the batch task group. If just one directory in the full pathname of a file is marked as nonshareable, that file cannot be used by the batch task group. The MF command is also used to control accessibility (read-only and write-only) to files created by any program, but particularly on behalf of a high-level language program that cannot set concurrency restrictions as a part of the language syntax. Read/write/share permissions can be set directly by an assembly language program.

Priority Level Assignments

Priority levels 5 through 62 are available for assignment to system, device driver, and application tasks. The priorities of system tasks and driver tasks are established during configuration. The priorities of application tasks are assigned during task group creation. The procedures for establishing priorities are described below.

Assigning Priority Levels to Devices and System Tasks

Priority levels for devices are specified by an argument of the CLM DEVICE or DRIVER directive when the system is configured. The first CLM DEVICE directive for each device type causes the appropriate Honeywell-written device driver to be loaded as part of the system. A CLM DRIVER directive is required only for a user-written driver. The two priority levels following the last one assigned to a configured device are used by system tasks and cannot be assigned to application tasks.

An example of priority level assignment is shown in Table 6-1. Levels 0 through 4 are assigned by the system and are not available to the user. The operator terminal must be specified by a CLM DEVICE directive that includes a value for the level. Normally it is assigned level 5, but any appropriate level can be assigned. At system initialization, the system bootstrap device is assigned level 6. This assignment remains in effect unless it is changed by a CLM DEVICE directive. Debug requires two priority levels which must be higher than any task it is to debug. Thus for the assignments shown in Table 6-1, it must be assigned levels 7 and 8 if it is to be able to operate on the communications supervisor.

Table 6-1 indicates I/O devices, and not device drivers, to stress that each peripheral device must have at least one level assigned to it; peripherals (other than communications devices) cannot share a level. If there are two printers, each must be assigned a unique level even though there is only one copy of a reentrant I/O driver. Communications configurations require one nonsharable level (CLM COMM directive) dedicated to processing communications interrupts, and it must be at a higher level than any communications device. Communications devices can share a level. For exampe, four TTYs and one VIP can be configured to share one level or to use up to five levels. The priorities in Table 6-1 provide maximum throughput because high-transfer-rate devices are assigned a higher priority than low-transfer-rate devices.

Theoretically, you could assign a level number as high as 59 to a device. In which case, levels 60 and 61 would be used by the system and level 62 would be assigned to a user task group. In practice, however, you would want to reserve levels for more than one user task group – especially for a system with a large number of devices. If priority levels 5 and 6 are assigned as shown in Table 6-1, the theoretical range of levels assignable on a CLM COMM directive is 7 through 58. For a device associated with a COMM directive, the range is 8 through 59.

Assigning Priorities to Application Tasks

Priorities are assigned to user task groups and tasks when they are created or spawned. The command to generate a task group contains an argument that specifies the base priority level for the task group. The base priority level is relative to the highest number of the priority level that has been assigned a configured device. When a task group is assigned a base priority level of zero, the lead task of the group executes at the physical interrupt priority level that is three level numbers above the highest level number assigned to a configured device. When other tasks in the same task group are created or spawned, they are given level numbers relative to the base priority level assigned to the task group. The physical interrupt level at which a task executes is the sum of:

- o the highest level number assigned to a configured device plus 3
- o the base priority level number of the task group, and
- o the relative priority level of the task within that group.

This sum must not exceed 62. Task groups that have the same base priority level are processed on a round robin basis.

TABLE 6-1. PRIORITY LEVEL ASSIGNMENTS FOR TASKS AND DEVICES

Physical Priority Level	Base Priority Level	Use	Comments
0 1 2 3 4	N/A N/A N/A N/A N/A	Power failure handler Watchdog timer runout Trap save area overflow Inhibit interrupts System clock	Level 0 through 4 are automatically assigned by the system
5	N/A	Operator terminal	Conventionally assigned level 5, but can be assigned any available level
6	N/A	System bootstrap device	Set to level 6 at system initialization but can be changed by the user
7 8	N/A	Debug program	Requires two levels. Can debug only those programs or drivers that have a lower priority
9	N/A	Communications supervisor	Must be a higher level than any communications device
10 10 10	N/A N/A N/A	TTY device TTY device TTY device	Communications devices can share priority levels
11 11	N/A N/A	Removable cartridge disk Fixed cartridge disk	The priority level for a pair of fixed/removable disks must be the same
12 13 14	N/A N/A N/A	Diskette Diskette Diskette	
15 16	N/A N/A	Line printer Card reader	
17 18	N/A N/A	Reserved by system Reserved by system	The next two levels following the last used for a configured device are used by the system
      • •	0 1 • • • 10	Task group A Task group B Task group n	
63	N/A	System idle loop	Always active

User tasks that are to execute in the online dimension are usually given higher priorities (lower level numbers) than those in the batch dimension. Tasks that are I/O bound should be run at a higher priority than tasks that are central processor bound. This permits I/O-bound tasks, which run in short bursts, to issue I/O data transfer orders as needed, wait for I/O completion, and, while in the wait state, relinquish control of the central processor to the central processor-bound tasks. Otherwise, if the central processor-bound tasks have a higher priority, the I/O devices would be idle while I/O bound tasks waited to receive central processor time.

Logical Resource Number (LRN)

An LRN is an internal identifier used to refer to task code and devices independently of their physical priority levels. Use of LRNs makes assembly language application task code independent of priority levels so that if circumstances require a change in priority levels, the task code does not have to be reassembled.

Device LRNs

LRNs are assigned to devices in the CLM DEVICE directives when the system is configured. The operator's terminal is assigned to LRN 0 at configuration. The bootstrap device is given an LRN value of 1, and, if an MLCP-connected operator's terminal is configured, the system assigns it an LRN of 2. Figure 6-3 is an example of priority level assignments for devices and system tasks and the related device LRNs.

LRN	LEVEL	
	0	
	3	INT
	4	CLOCK
0	5	OPERATOR'S TERMINAL
1	6	DISK
3	7	LINE PRINTER
4	8	SERIAL PRINTER
5	9	CARD READER
	10	OPERATOR INTERFACE MANAGER INTERRUPT
	11	SYSTEM TASK

Figure 6-3. Example of LRN and Priority Level Assignments to System Tasks and Devices

Application Task LRNs

LRN assignments to application program tasks are not dependent on the system configuration on which the application task group is running. LRNs are assigned to task code within an assembly language application program through specification of the create group/task macro calls, as well as the macro calls that build data structures ($IORB, $TRB, etc.). LRNs can be assigned at the control language level through the use of the commands (including operator commands) for creation of tasks groups and tasks. An LRN for an application task can have any value from 0 through 252. Within a task group, the LRN for each task must be unique. More than one LRN can be associated with the same level. For example, two tasks at level 23 can be assigned LRNs of 28 and 29, respectively.

Logical File Number (LFN)

Logical file numbers are internal file identifiers that are associated with file pathnames either at the assembly language level, or for high-level languages at the command level, through GET or ASSOCIATE commands.

Inter/Intra Task Group Communication

Whether or not information can be passed between task groups and tasks depends upon the following considerations:

- o Language in which task code is written
- o Use of the same file by more than one task group
- o Use of the message facility

Language Considerations

Task code written in assembly language can pass information to other assembly language tasks in the same task group by using variable-length request blocks. (See the *System Service Macro Calls* manual for details about building these data structures.) High-level languages cannot use this mechanism directly, but would require called subroutines written in assembly language.

Use of Common Files

Tasks within the same (or different) task group can communicate via disk files. The concurrency status must be the same for all tasks using the files. The requesting tasks must have access rights to the files.

Use of the Message Facility

The message facility allows online communications between two task groups using assembly language code. Two task groups communicate by sending or receiving messages to/from a uniquely named message queue container called a mailbox.

In using the message facility, the task groups issue commands to prepare the mailboxes and system service macro calls to send and receive messages.

Mailbox Preparation

The user or operator must create the mailbox directory, create the needed mailboxes, and set access controls on the mailboxes.

The mailbox directory is the directory that is to contain the simple names of the mailboxes. It is created through a standard CD (create directory) command.

The user creates each needed mailbox through a CMEX (create mailbox) command. This command creates a directory corresponding to the mailbox name and a file ($MBX) defining the mailbox attributes.

Since memory queuing of the messages is required, the CMBX command must have the -MEM argument with a pool-id that has been defined at system building. The queue capacity in bytes is specified by the -SIZE argument.

To prevent unauthorized use of the message queues, the user should set access controls on each created mailbox. Access rights must be set as follows:

- The sender must have list access on the directory defining the mailbox.
- The receiver must have read access on the $MBX file for a given mailbox.

The following examples show how to create mailboxes and how to set access to them. (See the *Commands* manual for details on the various commands.)

1. Create the special mailbox directory.[1]
   CD >MDD

[1] If the user creates a mailbox directory named MDD, the CMBX command need not specify >MDD and >$MBX (the system assumes the absolute pathname to be >MDD>mailbox_directory>$MBX). If the user creates a mailbox directory not named MDD, the CMBX command must specify an absolute pathname.

CUT ALONG LINE

HONEYWELL INFORMATION SYSTEMS

Technical Publications Remarks Form

TITLE	SERIES 60 (LEVEL 6) GCOS 6 MOD 400 SYSTEM CONCEPTS ADDENDUM C	ORDER NO.	CB20-00C
		DATED	JULY 1979

ERRORS IN PUBLICATION

SUGGESTIONS FOR IMPROVEMENT TO PUBLICATION

Your comments will be promptly investigated by appropriate technical personnel and action will be taken as required. If you require a written reply, check here and furnish complete mailing address below. ☐

FROM: NAME _____ DATE _____

TITLE _____

COMPANY _____

ADDRESS _____

_____

PLEASE FOLD AND TAPE –
NOTE: U. S. Postal Service will not deliver stapled forms

FIRST CLASS
PERMIT NO. 39531
WALTHAM, MA
02154

Business Reply Mail
Postage Stamp Not Necessary if Mailed in the United States

Postage Will Be Paid By:

HONEYWELL INFORMATION SYSTEMS
200 SMITH STREET
WALTHAM, MA 02154

ATTENTION: PUBLICATIONS, MS 486

FOLD ALONG LINE

FOLD ALONG LINE

CUT ALONG LINE

Honeywell