Honeywell

SYSTEM CONTROL

SERIES 60 (LEVEL 6)

GCOS 6/MDT

SUBJECT:

Detailed Description of System Control for Series 60 (Level 6) GCOS Multi-Dimensional Tasking (GCOS 6/MDT)

SOFTWARE SUPPORTED:

This publication supports Release 0101 of Series 60 (Level 6) GCOS Multi-Dimensional Tasking (GCOS 6/MDT) software. When a later release of the system occurs, see the Subject Directory of the latest Series 60 (Level 6) *GCOS 6/MDT Overview and User's Guide* (Order No. AX11), to ascertain whether this revision of this manual supports that release.

DATE:

March 1977

ORDER NUMBER:

AX07, Rev. 0

PREFACE

This manual describes the system control language for the Series 60 (Level 6) GCOS Multi-Dimensional Tasking (GCOS 6/MDT) operating system. Unless stated otherwise, the term MDT is used throughout this manual to refer to the GCOS 6/MDT software; the term Level 6 indicates the specific models of Series 60 (Level 6) on which the described software executes.

The System Control manual comprises five sections, whose subject matter is briefly described below.

- Section 1 – An introduction to MDT concepts which are useful from the point of view of system configuration and control.
- Section 2–Formats and functional descriptions of the directives used to define the hardware and software components which constitute the operating system.
- Section 3–Formats and functional descriptions of the set of commands which can be used for operator control of the system.
- Section 4–Formats and functional descriptions of the set of commands which can be used for user control of the system.
- Section 5–A list of the error and status messages which can be issued by the various system components.

Three appendices supply summary or supplementary information to that which is contained in the main body of the manual.

- Appendix A–An alphabetical list of all operator control language and execution control language commands with their parameters.
- Appendix B–A description of the Level 6 to Level 66 file transmission capability and the commands which enable its use.
- Appendix C–The definition of the standard GCOS 6/MDT character set and their hexadecimal equivalents.

File No.: 1S13

GCOS 6/MDT Subject Directory

This subject directory lists topics in alphabetical order. Each topic is accompanied by the order number of each manual in which the topic is described. Following the Subject Directory is a list, by order number, of all GCOS 6/MDT manuals.

The following publications constitute the GCOS 6/MDT manual set. The Subject Directory in the latest Series 60 (Level 6) GCOS 6/MDT Software Overview and User's Guide lists the current revision number and addenda (if any) for each manual in the set.

Order No.	*Manual Title*
AX07	*Series 60 (Level 6) GCOS 6/MDT System Control*
AX08	*Series 60 (Level 6) GCOS 6/MDT Program Preparation and Checkout*
AX09	*Series 60 (Level 6) GCOS 6/MDT Data File Organization and Format*
AX10	*Series 60 (Level 6) GCOS 6/MDT Monitor and I/O Service Calls*
AX11	*Series 60 (Level 6) GCOS 6/MDT Overview and User's Guide*
AX12	*Series 60 (Level 6) GCOS 6/MDT Assembly Language Reference Manual*
AX13	*Series 60 (Level 6) GCOS 6/MDT COBOL Reference Manual*
AX14	*Series 60 (Level 6) GCOS 6/MDT FORTRAN Reference Manual*
AX15	*Series 60 (Level 6) GCOS 6/MDT Sort Manual*
AX16	*Series 60 (Level 6) GCOS 6/MDT RPG Reference Manual*

In addition to the GCOS 6/MDT manual set, the following documents provide GCOS 6/MDT users with a general hardware reference:

Order No.	*Document Title*
AS22	*Honeywell Level 6 Minicomputer Handbook*
AT04	*Level 6 System and Peripherals Operation Manual*
AU22	*GCOS/BES Programmer's Reference Card*

The following manual provides detailed information regarding programming for the Multiline Communications Processor:

AT97	*Series 60 (Level 6) MLCP Programmer's Reference Manual*

CONTENTS

ILLUSTRATIONS

TABLES

SECTION 1

INTRODUCTION

The MDT operating system is a system that supports concurrent multiple interrupt-driven activities. Control over activities and their accesses to external devices and files is vested in the MDT Monitor and Input-Output Control system, which utilizes external interrupts and a set of physical priority levels to determine the sequencing and degree of concurrency of functions performed on behalf of these activities.

The topics presented in this section relate to the definition of the MDT operating system configuration and its subsequent control by the system operator and its various users. The material contained herein by no means exhausts the concepts necessary to a complete understanding of the operating system; rather it centers on system configuration and control, and thus reflects the most visible interfaces between the operating system and a user at an operator terminal or a remote terminal device. For a more complete presentation of these topics, refer to the Overview and User's Guide manual.

APPLICATION EXECUTION

The basic element which is executed on behalf of an application is the *task.* One or more tasks constitute a *task group.* Associated with a task group is a *group identification* by which it is known to the operating system, and a set of *resources* consisting of the member task(s), the central processor, and peripheral devices.

Tasks

A *task* is defined as a sequence of instructions which has an explicit starting point and an explicit termination point, and performs some identifiable function. A task has associated with it a unique priority level and is identified by a logical resource number (LRN). These attributes are defined when the task and the group of which it is a member are defined.

Task Groups

A *task group* is a named set of one or more tasks which share the same set of resources. The task group is the framework within which all user applications, all program development and checkout activities, and all operating system service functions operate.

Some task group concepts which are useful from the point of view of system configuration and control are discussed in the following paragraphs.

Task Group Identification

Every task group is known to the operating system by a task group identifier, or, as it is commonly known, a *group id.* The system task group and the batch task group each have a fixed group id; the former's is $S, while that of the latter is $B. The identifier of an online task group can be any two characters from the sets of alphabetic and numeric characters (e.g., A1, XY, 2Z). A special online task group, identified as $H, is used during system configuration (refer to Section 2).

The task group identification is used in a number of commands to identify the group upon which action invoked by the command is to be taken. It is also used for operator and user communication with specific task groups in response to messages issued by the groups.

User Identification

A user identification, or *user id,* is meaningful only in the context of a batch or online task group, and identifies the user on whose behalf the task group is currently active. The user id is supplied when a request for the task group's services is entered. It is a field of up to 18 characters, comprising two subfields of the form "user.project". These subfields are used to establish the initial pathname of the working directory.

Task Group Resources

The resources "owned" by a task group, in addition to the *tasks* described above, fall into three main categories, central processor *priority levels,* central processor *memory*, and *peripheral devices.*

Priority Levels

A task group, when it is created, is assigned to a "physical" or base priority level. Each task that is a member of this group is assigned to a priority level relative to the group's base level. In this way the tasks within a group can be made to retain the same priority levels relative to each other, while the base level can be changed to reflect the relative urgency of the application at any given invocation.

Task Group Memory

When a task group is created, it is assigned a fixed block of memory for execution. This block is known as a *memory pool.* From it the task group obtains segments of memory as required for such things as task code, task control data structures, and input/output buffers.

A detailed description of memory pools and their definition is contained in Section 2.

Peripheral Resources

A task group's peripheral resources consist of those devices assigned to it when they are requested. By implication, this also includes any data which may be read from or written to these devices, be they magnetic storage devices, unit record devices, or terminal-type devices.

The mechanism by which peripheral devices are assigned to a task group is the logical resource number (LRN). An LRN is defined for each peripheral device when the system is configured, beginning with LRN0 for the operator terminal. LRNs 1 through n are assigned to other devices (including MDC-connected local devices and MCP-connected remote devices). LRNs from n+1 through 255 are available for assignment to tasks.

When a task group is created a maximum LRN can be specified. If it is, it must be large enough to include the LRNs of any tasks that are expected to operate within the task troup. If it is not specified, the default maximum LRN is the highest one defined during system configuration (i.e., the value attained by n, as described above).

FILE SYSTEM PATHNAMES

The file system is represented by a tree-structured hierarchy. The basic elements of this structure are known as files. Some of the files are of a special type known as directories; the remainder of the files comprise aggregates of data.

Files

A file is defined as any unit of storage, external to the central processor, which is capable of supplying data to or receiving data from a task. A file can be simply a peripheral device such as a printer, card reader, or terminal device; or it can be an aggregate of data stored within a directory structure on a magnetic storage device. A source unit, object unit, listing, or bound unit is stored as a source unit file, object unit file, list file, or bound unit file, respectively.

Directories

A directory is a file that contains information about other files, such as their physical and logical attributes, and the attributes of the peripheral devices upon which they reside. The files whose attributes are described in a directory are said to be immediately contained in, or subordinate to, that directory. They may themselves be directories, or they may be data files.

At the base of each tree structure is a directory known as the root directory, or simply the root. The root directory name is the same as the identifier or the label of the volume on which it resides.

Naming Conventions

Each directory or file name in the file system can consist of ASCII characters from the following sets:

- Uppercase alphabetic (A through Z)
- Numeric (0 through 9)
- The underscore (_)
- The period (.)
- The dollar sign ($)

The first character of any name must be either an alphabetic or the dollar sign ($). The underscore character can be used to join two or more words which are to be interpreted as a single name (e.g., DATE_TIME). The use of the period character followed by one or more alphabetic or numeric characters is normally interpreted as a suffix appended to a file name.

The name of a root directory or a volume identifier can consist of from one through six characters. The names of other directories, and those of files, can comprise from one to twelve characters. The length of a file name must be such that any potential system-supplied suffix does not result in a name of more than 12 characters. The total length of a pathname cannot exceed 57 characters.

Pathname Construction

The access path to any file system entry (directory or file) begins with a root directory name and proceeds through zero or more subdirectory levels to the desired entry. The series of directory names (and a single file name if a file is the target entry) is known as the entry's *pathname.*

In constructing pathnames, certain symbols are used to indicate the hierarchical relationship among the pathname's elements. These symbols and their meanings are shown below.

- The circumflex (^)–Used exclusively to denote the name of a root directory. It precedes the root directory name, thus: ^VOL01.
- The greater than (>)–Used to connect two directory names or a directory name and a file name. Each occurrence of the symbol denotes a change of one directory level; the name to the right of the symbol is immediately subordinate to the name on the left. Reading a pathname from left to right thus traverses the tree structure in a direction *away from* the root. If the root directory VOL01 contains a directory name DIR1, then the path name of DIR1 is

  ^VOL01>DIR1

  If the directory named DIR1 in turn contains a file named FILEA, then the pathname of FILEA is

  ^VOL01>DIR1>FILEA

- The less than (<)–Used in certain cases to indicate movement through the tree structure in a direction *toward* the root. This symbol represents a change of one level toward the root. Consecutive symbols can be used to represent a change of more than one level.

The last element in a pathname is the name of the entry upon which action is to be taken. This element can be either a directory name or a file name, depending on the function to be performed.

Absolute Pathnames

An absolute pathname is one that begins with a directory name preceded by circumflex (^) or a greater-than symbol (>). When it begins with a circumflex it is called a full pathname. When it begins with a greater-than symbol, the first element is immediately subordinate to the root directory of the system volume.

Relative Pathnames and the Working Directory

A relative pathname is one that does *not* begin with the circumflex or greater-than symbol. The first (or only) name in a relative pathname identifies a directory or file that is immediately subordinate to a directory known as the *working directory.*

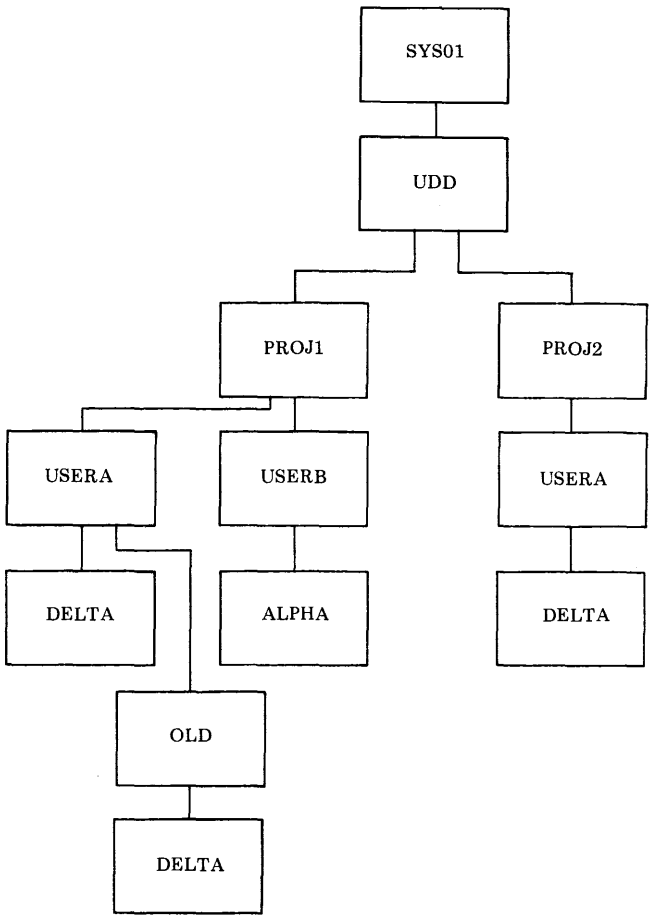
A *simple* name is a special case of the relative pathname. It consists of one and only one element, the name of the desired entry in the working directory.

The following examples and diagram show some relative pathnames and the full pathnames they represent when the working directory pathname is

>UDD>PROJ1>USERA

and the system was initialized from a volume whose volume identification is SYS01.

Relative Pathname	*Full Pathname*
DELTA	^SYS01>UDD>PROJ1>USERA>DELTA
OLD>DELTA	^SYS01>UDD>PROJ1>USERA>OLD>DELTA
<USERB>ALPHA	^SYS01>UDD>PROJ1>USERB>ALPHA
<<PROJ2>USERA>DELTA	^SYS01>UDD>PROJ2>USERA>DELTA
<	^SYS01>UDD>PROJ1



OPERATOR SYSTEM DIALOG

Communication between system operator and the various active task groups is conducted through a terminal-type device known as the *operator terminal.* The system operator, for the purpose of this discussion, is any person who carries on a dialog with the operating system through this terminal.

The operator terminal is normally (although it need not be) a terminal which is in close proximity to the central processor, and usually is housed in the same cabinet. A terminal is defined as an operator terminal at system configuration by virtue of being attached to the lowest channel number. The system assigns an LRN of zero to this device.

The operator terminal is controlled by a system software component known as the Operator Interface Manager (OIM). This component provides a standard means by which all tasks can communicate with the operator. The OIM guarantees consistency of input and output spacing to prevent overprinting of messages; it also manages multiple requests for input from the operator terminal and insures that critical messages can be printed promptly. The operator can choose which requests he will respond to and when he will respond to them.

The OIM recognizes four classes of messages. One of these classes comprises output messages (messages sent from a task group to the operator terminal), while the remaining three comprise input messages (messages sent from the operator terminal to a task group or to the OIM itself). These classes are further described in the paragraphs that follow.

Output Messages

The OIM identifies the messages written to the operator's console by providing the task group identification in a prefix to each message. If the operator must respond, a message number is also provided. The formats for the two kinds of output messages to the operator's console are shown below.

Output Messages	*Meaning*
#n(id)message	A message written to the operator's console that requires operator response. OIM provides the message number n.
(id)message	An informational message from the task group "id" to the operator's console that does not need a response.

Up to ten messages requiring responses can be outstanding at any given time. After the tenth message (message #9) is issued, if none of the messages have been responded to, no further messages requiring responses are issued. The task(s) attempting to issue the eleventh (and successive) message(s) are stalled until one or more of the outstanding messages are answered, making a message number available.

The Operator Interface Manager provides standard formatting for all output messages. Each output message is preceded by a line feed and followed by a carriage return and line feed. The first byte of each output message is a control byte initialized by the OIM. In addition, the OIM provides a pacing feature to control the rate at which orders for output are displayed on the operator's console device. On a CRT, pacing controls the rate of rollup on the screen. The operator can change the pacing rate by issuing an input directive to the OIM (see below).

Output message length is restricted to 140 bytes. If the output IORB specifies a range of more than 140 bytes, it will be reset to 140 by the OIM, without notice to the caller. If the specified range is greater than the physical line length of the LRN 0 device, the issuing task is responsible for imbedding the appropriate carriage returns and line feeds as required to display the message on at least two lines.

Input to the OIM

Input message formats are shown in the paragraphs that follow.

Input Messages

Two message formats are provided for messages sent to a task group, not in response to an input request. One format requires the task group id; the other assumes the message is intended for the OIM default task group.

Immediately after system configuration the default task group is the system task group "$S". At this time OCL commands (i.e., messages through the OIM) need not be preceded by "$S". If a directive to OIM were to change the default to another task group, all OCL commands would have to be preceded by "$S".

Two other input message formats are provided to respond to requests for input. One includes a prefix for message number, i.e., the number contained in the output message requesting the response; the other requires no message number and is assumed to respond to message #0.

A message consisting of ΔC/R or C/R is interpreted as a response to message #0; this number then becomes available for another message. If ΔC/R is entered and message #0 does not exist, an error message is generated.

The four input message formats are shown below.

Input Messages	*Meaning*
messageC/R	Serial input to the default task group. After configuration the system task group, $S, is the default.
id messageC/R	Serial input to a task group specified by task group id.
ΔmessageC/R	Response to output message 0.
Δn messageC/R	Response to output message #n where n=0 to 9, inclusive.

Message Control

Output to the console can be halted, to enable the operator to leave the console, by typing a space and omitting the carriage return. To resume output the space is erased by typing the @ character on the same line, followed by a carriage return.

To delete a message line before it is accepted as input, i.e., before C/R is typed, enter Δ(CONTROL-X)C/R.

Message control formats are shown below.

Input Message	*Control Meaning*
Δ(no C/R)	Halt output.
Δ@C/R	Resume output.
Δ(CONTROL-X)C/R	Delete partial input (prior to carriage return)

Input Directive Messages

The OIM recognizes three directives to: list outstanding messages, change the default task group, and change the pacing rate.

All outstanding messages are listed by entering

ΔCΔ?C/R

To minimize the amount of information that must be entered from the operator's console during the operator/task group dialog, the OIM recognizes one task group to be the default, i.e., a group identifier is not required in the prefix of an input message. The default can be changed using the OIM C (change) directive

ΔCΔ:group-id:C/R

To change the pacing rate at which output messages are displayed at the console (e.g., one message per second or one message per tenth of a second) enter

ΔCΔPnnnC/R

where nnn is the pacing interval in tenths of a second.

Input directive formats are shown below.

Input Directive Messages	*Meaning*
ΔCΔ?C/R	OIM lists all outstanding output messages.
ΔCΔ:id:C/R	The default task group is changed to the task group whose identifier is "id".
ΔCΔPnnnC/R	Change the pacing rate to nnn; nnn values are 001 to 999 tenths of seconds.

NOTE: In all message formats:
C/R is Carriage Return
Δ indicates exactly one space

Input Message Length

The end of an input message is indicated by typing a C/R (carriage return) Table 5-1 lists the maximum physical line lengths by device type. When the line length limit is reached, the OIM considers the message finished as if a C/R were entered. To extend the length of a message beyond the physical line length, the carriage return can be "escaped" with a preceding backslash (\), e.g., \ C/R. Then, up to 140 bytes can be entered.

ASSEMBLY LANGUAGE PROGRAMMING CONSIDERATIONS

Assembly language programming considerations are discussed in the section on the Operator's Console of the *Monitor and I/O Service Calls* manual.

TABLE 1-1. SUMMARY OF CONSOLE MESSAGE LENGTHS

Device Type	Physical Line Length (bytes)	Maximum Message Length	
		Keyboard Input	Screen/Printer Output[a]
KSR	72	140	140 136(+4)[b]
CRT	80	140	140 136(+4)
Keyboard Typewriter Console	132	140	140 136(+4)

[a]Carriage return and line feed characters must be imbedded in the user program's output message to continue beyond the physical line length limit.

[b]A 4-character identifier for the message.

SAMPLE CONSOLE DIALOG

Figure 1-1 provides a sample operator-task group dialog. The boxed entries represent console input; the other entries are system and user task group output to the console.

Entry	*Meaning*
($S)CONSOLE READY	Output from the system task group
CG AB 16	Input to default task group ($S)
#0(AB)KEY IN THE TIME PLEASE	Task group AB requesting a response
Δ1021	Response to message #0
(AB)END OF SECTION 1	Informational message from task group AB
CG YZ 21	Input to default task group ($S)
#0(AB)KEY IN THE TIME AGAIN	Task group AB requesting a response
#1(YZ)MOUNT UNLINED PAPER PRINTER 1	Task group YZ requesting a response
Δ1ΔDONE	Response to message #1
(YZ)ENTER LIST W FOLLOWED BY "FINI"	Informational message indicates input need
ΔCΔ:YZ:	Operator defines new default task group
JONES	Serial input to default task group YZ
SMITH	
LAPIERRE	
FINI	
ΔCΔ:$S:	Changes default task group to system task group
SSPG AB	OCL input to default task group $S(OCL processor) to suspend task group AB
($S)GROUP AB SUSPENDED	Informational message
(YZ)ENTER LIST Z FOLLOWED BY "FINI"	Informational message
ΔYZΔLOWELL	Serial input to task group YZ
ΔYZΔHART	
ΔYZΔFINI	
ΔCΔ?	Operator requests list of outstanding messages
#0Δ(AB)KEY IN THE TIME AGAIN	Redisplay of message #0
Δ0Δ1100	Response to message #0

Figure 1-1. Sample Console Dialog

SECTION 2

SYSTEM STARTUP AND CONFIGURATION

Prior to system startup and configuration, it is desirable to have an understanding of the following MDT concepts described in the following manuals:

Overview and User's Guide
- o Allocation of memory: Provides information pertaining to the concept of memory pools used by task groups.
- o Communications capability: Provides information about the full range of communications capabilities for communications applications.
- o Online/batch operation: Provides information about online/batch operation as a result of assignment at system configuration and program preparation time.
- o Priority assignment: Provides information about priority levels assigned to each execution of tasks within task groups.

System Control
- o Operator Control Language (OCL): Provides information about the control language allowing the operator to control the system.
- o Execution Control Language (ECL): Provides information about the control language used to control application execution within the system.

SYSTEM STARTUP AND CONFIGURATION

System startup consists of either: (1) the Honeywell-supplied system startup or (2) the user specialized system startup.

The Honeywell-supplied system startup involves bootstrapping a preconfigured system residing on the system volume in the form of directive files which, when processed, provide the user with a minimum operating environment in which to build a configuration directive file to configure a system corresponding to the actual installation hardware.

Specialized system startup includes existence of the minimum operating environment and invoking the Editor to build the configuration directive file named CLM_USER. A second file, named START_UP.EC can also be built through use of the Editor; this file can contain OCL (operator control language) commands which, when executed, define the installation-specific operating environment consisting of task groups and directives. After both files are created, the system is again bootstrapped but configuration directives in the CLM_USER file are used to control the configuration and the OCL commands in the START_UP.EC File are used to define the operating environment.

Configuration directives, used in the directive file, are available to:

- o Describe available central processing unit options such as the real-time clock and scientific processor.
- o Describe peripheral and communications devices and their characteristics.
- o Specify memory pools, into which memory is partitioned, and memory pool sizes. The system and each user task group under which applications execute must be associated with a single memory pool. System and user memory pools are mutually exclusive.
- o Indicate system software, which otherwise would execute as an overlay, to be permanently resident.
- o Indicate that an application-specific bound unit should be resident and be part of the operating system.

The Honeywell-supplied startup, specialized startup and configuration directives are described in detail in this section.

Honeywell-Supplied System Startup

The Honeywell-supplied system startup provides the capability of configuring a minimal MDT system consisting of the following:

- o A line printer (LPT)
- o A card reader (CDR)
- o Either a removable cartridge disk (RCD) or two pairs of diskettes (DSK) depending on the bootstrap device
- o Either a local (MDC-connected) or remote (MCP-connected) terminal
- o One online memory pool and an online task group.

Once the initial Honeywell-supplied startup has been completed, the user can either: (1) invoke the Editor to build his own CLM_USER configuration directive file and START_UP.EC operator control language (OCL) file and proceed to specialized system startup or (2) execute tasks under the control of the existing system as configured.

The Honeywell-supplied system startup consists of the following steps:

1. Power-up the hardware by turning on the "Power" switch on the central processing unit (CPU).
2. Mount either the removable cartridge disk or one of the diskette pairs (i.e., for diskette, on either channels 0400 and 0480 or 1200 and 1280) containing the Honeywell-supplied MDT system.
3. Execute the following standard bootstrap routine by pressing the following keys on the CPU:

   Stop
   CLeaR
   Load
   Execute

   a. Wait for the "Traffic" light to extinguish; channel 0400_{16} is the bootstrap channel. At this point, certain bootstrap options are available to the user by setting hexadecimal values in register D1 (see "Bootstrap Routine Options," later in this section).
   b. Press the Execute key on the CPU and the appropriately mounted Honeywell-supplied MDT system file is selected. At this point, line speeds for a TTY operator's terminal can be specified (see "Line Speeds For A TTY Terminal," later in this section).

Upon completion of this phase, a basic system is available.

For specialized system startup, when the user-built CLM_USER configuration directive file is available, only the bootstrap routine (i.e., step 3.) is utilized. In any case, for the Honeywell-supplied system startup, Figure 2-1 shows an example of the type of information that will be supplied on the operator's terminal upon completion of the bootstrap routine.

```
($S)SYSTEM  0101   STARTUP
($S)(131301) 3E
($S) TYPE "CLMIN ---" TO REDIRECT INPUT, ELSE "QUIT"
QUIT
($S) CLMST1  770119
($S) CLMST2  770121 A
($S) CLMCOM  770124
($S) CLM 770119 A  COMPLETED
($S)SYSTEM GROUP READY
($H)GROUP READY
```

Figure 2-1. System Startup Terminal Responses

Figure 2-1 provides the following information:

- o ($S)SYSTEM 0101 STARTUP
  Indicates the revision number of system startup being used; in this case, revision number 0101.

o ($S) TYPE"CLMIN—" TO REDIRECT INPUT, ELSE "QUIT"

Indicates that additional configuration directives can optionally be entered, in the form of an input file, through either the operator's terminal, card reader or diskette in order to alter the Honeywell-supplied MDT system configuration. However, if any communications configuration directives are entered, they cannot be referenced until after a QUIT configuration directive has been entered. Any configuration directives, not previously specified, can be contained within the input file except for the following conditions:

If more than one SYS configuration directive has been previously specified, the last SYS configuration directive entered has precedence over any others.

If more than one MEMPOOL configuration directive, having an S (system) pool type, the last MEMPOOL configuration directive entered has precedence over any others.

The following are examples of the pathnames that might be entered through the operator's terminal to identify the input file:

CLMIN >SPD>CONSOLE (operator's terminal)
CLMIN >SPD>CDR00 (card reader)
CLMIN ^Z00B00>SID>CONFIGA (system disk)

If a configuration directive in the input file contains an error generated by either an incorrect configuration directive syntax or an illegal duplication of a configuration directive, the following error message is generated on the operator's terminal:

($S) (1313zz) hh {line 1
secondary message indicating line in error {line 2

zz—the error code
hh—level of task group containing CLM (Configuration Load Manager)

The user has one chance to enter the directive correctly. Upon correction of the error, the remaining directives in the original input file will be processed.
The QUIT configuration directive should only be used in the following instances:
If no configuration directives are to be entered from the input file.
If the input file doesn't already contain a QUIT configuration directive.
If a QUIT configuration directive is not present in the input file and an EOF (end-of-file) is reached, the following "error" message appears on the operator's terminal:

($S) (13130A) hh {line 1
021F {line 2

hh—level of the task group containing CLM (Configuration Load Manager)

The operator can either enter: (1) a QUIT configuration directive, (2) a CLMIN configuration directive to reference another input file, or (3) any other configuration directive which will again be followed by the EOF "error" message. However, as end-of-file for the original input file has been reached, no remaining directives in the original input file will be processed. At this point, the user can continue to enter configuration directives from the operator's terminal by entering CLMIN >SPD> CONSOLE until either a QUIT or another CLMIN configuration directive is entered to reference another input file.

o ($S) CLMST1 770119
($S) CLMST2 770121 A
($S) CLMCOM 770124
($S) CLM 770119 A COMPLETED

Indicates the names and the dates of the configuration software which change with each release of the MDT system.

- ($S) SYSTEM GROUP READY

  Indicates that the system task group, having the operator control language (OCL) processor as its lead task, is ready for processing. The Honeywell-supplied START_UP.EC file has configured a single task group and has caused this typeout to occur.

- ($H) GROUP READY

  Indicates that the online task group, requested in the Honeywell-supplied START_UP.EC file, having the execution control language (ECL) processor as its lead task, has been created and can be used for either: (1) generation of a specialized system to be used in the specialized system startup procedure, (2) program development, or (3) execution.

Configuration Concepts

The following configuration concepts *must* be thoroughly understood when configuring systems.

Configuration Directives

The following configuration concepts are implemented through use of the configuration directives mentioned within each conceptual description.

Overlays (RESOLA Configuration Directive)

The operating system software includes system functions implemented as overlays. Each overlay executes in a system overlay area. There is one overlay configured by default. Increasing the number of overlay areas can speed-up the operation. A total of 10 overlay areas (having a size of 256 words per overlay area) can be configured through the use of the olan parameter on the SYS configuration directive. The operating system determines which overlay area should be used.

An alternate capability to control overlays within the system is available. Instead of defining multiple overlay areas for critical functions, system overlays may be made resident in memory at configuration time through use of the RESOLA configuration directive. All of the system overlays are shown in Table 2-9, later in this section. Each of the system overlays has been defined as floatable through use of the FLOVLY linker directive.

Operating System Extensions (LDBU Configuration Directive)

Operating system extensions consist of reentrant code in the form of a bound unit. Extensions are most effective when the bound unit containing reentrant code, is more or less continuously used: (1) by multiple task groups in situations that do not permit the possible delay of the initial loading of a shared bound unit (i.e., a bound unit not resident) or (2) as a subroutine accessed by symbolic references in an applications task group.

Operating system extension bound units may be specified as resident through use of the LDBU configuration directive when the operating system is initialized (see "Honeywell-supplied System Startup," earlier in this section). Such bound units may define system-wide global address symbols that are included in the operating system's resident symbol table. Any bound unit loaded subsequently may contain references to any global address symbols that were unresolved at link time; the operating system loader resolves references to this code as it loads the bound unit. These global address symbols must be defined in the operating system extension bound unit and not in a bound unit loaded after initialization.

System extension code not used by multiple applications could be linked into the bound units of the task requiring such code (rather than being specified as resident when the operating system is initialized) thereby resolving all address symbols at link time and remaining memory resident only while the task is not aborted or deleted.

Memory requested by an operating system extension bound unit is supplied from the application task group memory pool in which the reentrant code executes and not form the operating system memory pool.

Memory Allocation (MEMPOOL Configuration Directive)

Memory is divided between system, online and batch memory pools.

A task group is a named set of one or more application tasks sharing the same resources. Each task group is assigned to one and only one memory pool defining an area of memory in which it resides and

from which it may acquire additional memory during execution. More than one task group can be assigned to the same online memory pool. There are three memory pool classifications, one of which is divided into two subclassifications:

- o System pool
- o Online pool
  - Exclusive pools
  - Nonexclusive pools
- o Batch pool

See the Overview and User's Guide manual for a detailed description of memory pools.

Peripheral Device (DEVICE Configuration Directive)

A device can be accessed through either file management or physical I/O. (However, in a communications environment, a DEVICE configuration directive is not required for a device accessed through physical I/O.) Each device accessed through file management, must be described by certain physical and logical parameters specified in the DEVICE configuration directive described later in this section.

The physical parameters specified by the DEVICE configuration directive include:

- o Unique (i.e., nonduplicate) device-types
- o Central processing unit (CPU) priority level number
- o Device channel number on the I/O bus

The logical parameters specified by the DEVICE configuration directive include:

- o Logical resource number by which the device is requested
- o Unique (i.e., nonduplicate) device names
- o Maximum expected record size
- o Specification of buffered files

At system startup, all files on devices are defined as being shareable. They can be modified later through use of the MF (Modify File) OCL command described in Section 3 of this manual. A task executing in an online pool area can reference an inactive file on any device associated with either the online or batch pool area. However, a task executing in the batch pool area can only reference an inactive file on a device associated with the online pool area if that inactive file is defined as shareable.

Terminals

All terminals used by the MDT system are controlled either by the multiple device controller (MDC) or by the multiline communications processor (MCP).

The MDC controls the following:

- o Noncommunications terminals (KSRs)
- o Diskettes (DSKs)
- o Fixed cartridge disks (FCDs)
- o Removable cartridge disks (RCDs)
- o Card readers (CDRs)
- o Magnetic tapes (MT9s)

The MCP controls the following:

- o Asynchronous TTY terminals
- o Synchronous VIP terminals with or without ROPs
- o Binary synchronous communications (BSCs)

Determination Of The Operator's Console

The operator's terminal is either connected through an MDC or an MCP.

In a noncommunications environment with only one MDC-connected KSR, the default channel for the operator's terminal is 0500_{16} (X'0500') . However, if there is more than one MDC-connected KSR, the one with the smallest channel number will be the operator's terminal.

In a communications environment (with no MDC-connected KSRs) with more than one MCP-connected KSR, the one with the largest channel number will be the operator's terminal.

Line Speeds For A TTY Terminal

If the operator's terminal is a MCP-connected TTY terminal and neither a user-built configuration directive file (i.e., CLM_USER) exists nor is there a MDC-connected KSR, the line speed for the TTY terminal, having the highest channel number, is determined by resetting register D1 after completing step 3b of the "Honeywell-supplied System Startup," (described earlier in this section). After step 3b, a value of 0110_{16} is displayed in register D1 on the CPU. The user then resets register D1 to one of the line speeds shown in Table 2-1. When register D1 has been set to the desired speed, the Execute key on the CPU is pressed.

TABLE 2-1. TTY TERMINAL LINE SPEEDS

Line Adapter 2108	Line Adapters 2110 or 2118
Speed (Entered in register D1)	**Speed (Entered in register D1)**
0050	0050
0075	0075
0110	0110
0134	0134
0150	0150
0300	0200
0600	0300
0900	0600
1200	1050
1800	1200
2400	1800
3600	2000
4800	2400
7200	4800
9600	9600

NOTE: A speed of 0134 represents 134.5 bits/second.

If a user-built configuration directive file (i.e., CLM_USER) exists and there is no MDC-connected KSR, the line speed is determined by the speed parameter of the TTY configuration directive used in defining the operator's terminal (see "Communications Configuration Directives," later in this section).

Bootstrap Routine Options

During the bootstrap routine, certain system generation options are available by setting register D1 to one of the hexadecimal values shown in Table 2-2. One of the main features of these options is to permit a Honeywell-supplied system configuration file to be used when an error has occurred on a user-built system configuration file (i.e., CLM_USER). By using the Honeywell-supplied system configuration file, the user has the capability of again configuring a minimum MDT system in order to invoke the Editor to correct or rebuild his own CLM_USER file.

After completion of step 3a of the bootstrap routine (see "Honeywell-supplied System Startup," earlier in this section), the default bootstrap channel 0400_{16} is displayed in register D1 on the CPU. To set register D1 to one of the hexadecimal values (in effect, options) shown in Table 2-2, press the following keys on the CPU:

- o Change
- o Write
- o Enter the appropriate hexadecimal value for register D1 by utilizing the numeric register keys on the CPU.

  Example:
  Change: D1=0400_{16} to 0404_{16} (halt after bootstrap)
  Press Register Keys: 0 4 0 4
  The value 0404_{16} will be left-justified when register D1 is displayed on the CPU.
- o Load
- o Execute

TABLE 2-2. BOOTSTRAP OPTIONS

Values (Hexadecimal)	Options
0400	No change from current operation.
0401	Bootstrap the system from the FCD (fixed cartridge disk); use either the CLM_USER or a Honeywell-supplied file.
0402	Ignore the CLM_USER file and bootstrap the system from diskette; use a Honeywell-supplied file.
0403	Ignore the CLM_USER file and bootstrap the system from the FCD (fixed cartridge disk); use a Honeywell-supplied file.
0404	Halt after bootstrap.
0405	Bootstrap the system from the FCD and then halt.
0406	Halt after the bootstrap. Ignore the CLM_USER file and bootstrap the system from diskette; use a Honeywell-supplied file.
0407	Ignore the CLM_USER file. Bootstrap the system from the FCD and halt; use a Honeywell-supplied file.

NOTE: When a halt is indicated, press the Execute key on the CPU to continue.

Specialized System Startup

Specialized system startup consists of the following procedures:

- o Through use of the Editor creation of a user file, named CLM_USER, containing configuration directives which, when executed, configure the system to correspond to the actual installation hardware.
- o Through use of the Editor, creation of an optional file, named START_UP.EC, containing one or more operator control language (OCL) commands which are interpreted by the OCL processor. This file provides a mechanism where by a sequence of routinely performed OCL functions can be executed without the need for manually entering the functions through an interactive terminal.

After having completed the Honeywell-supplied system startup, a foreground task group (i.e., $H) is now available to invoke the Editor through the ED execution control language (ECL) command. Once the Editor is available, the necessary Editor commands in conjunction with the desired configuration directives and, optionally, OCL commands can be used to build the CLM_USER and START_UP.EC files, respectively. Figures 2-2 and 2-3 are generalized and specific examples, respectively, of the input structure involved in creating and building the CLM_USER configuration directive file and the START_UP. EC OCL file.

Operator's Terminal		Interpretation
($H)GROUP READY		Indicates the foreground task group has been created.
ΔC :$H:	c/r	Change directive. Enables entry of Editor directives without each directive being prefaced by ($H).
ED	c/r	Invokes the Editor.
A	c/r	Indicates the append Editor directive is being used to insert one or more configuration directives into the current buffer.
LDBU ...	c/r	Indicate one or more of the configuration directives which can be entered to configure a user-specialized system, depending on the available hardware. See "System Configuration Directives," and "Communications Configuration Directives," later in this section.
CLMIN ...	c/r	
DEVICE ...	c/r	
MEMPOOL ...	c/r	
RESOLA ...	c/r	
SYS ...	c/r	
COMM ...	c/r	
MODEM ...	c/r	
LPHDEF ...	c/r	
BSC ...	c/r	
LPHn ...	c/r	
STATION ...	c/r	
TTY ...	c/r	
VIP ...	c/r	
!F	c/r	Indicates an Editor escape sequence character which terminates the appended text (in the form of directives to be written to the CLM_USER configuration directive file).
1,$P (optional)	c/r	Indicates use of the Editor print directive to list each line of the contents of the current buffer. This may prove useful for verification of the configuration directives.
W SID>CLM_USER	c/r	Indicates a write Editor directive is being used to write the contents of the current buffer (containing configuration directives) as the file CLM_USER residing on the system volume. NOTE: Issuance of the write Editor directive, in effect, creates the file CLM_USER.
1,$D	c/r	Deletes all of the lines in the current buffer (containing configuration directives).
A	c/r	Indicates the append Editor directive is being used to insert one or more OCL commands into the current buffer.
ABORT_BATCH ...	c/r	Indicate one or more of the OCL commands which can be entered to perform routine operator control language (OCL) functions. See Section III of this manual for a complete description of all OCL commands.
ABR ...	c/r	
ABORT_GROUP ...	c/r	
AGR ...	c/r	
ACTB ...	c/r	
ACTG ...	c/r	
CSD ...	c/r	
CWD ...	c/r	
CB ...	c/r	
CG ...	c/r	
DB ...	c/r	
DG ...	c/r	
EBR ...	c/r	
EGR ...	c/r	
EC ...	c/r	
FO ...	c/r	
LSR ...	c/r	
LWD ...	c/r	

Figure 2-2. Generalized Example of Creating and Building CLM_USER and START_UP. EC Files

Operator's Terminal		Interpretation
MSW ...	c/r	
MF ...	c/r	
RDF ...	c/r	
RDN ...	c/r	
RAS ...	c/r	
SET_DATE ...	c/r	
SG ...	c/r	
STS ...	c/r	
STG ...	c/r	
SSPB ...	c/r	
SSPG ...	c/r	
!F	c/r	Indicates an Editor escape sequence character which terminates the appended text (in the form of commands to be written to the START_UP.EC OCL file).
1,$P (optional)	c/r	Indicates use of the Editor print directive to list each line of the contents of the current buffer. This may prove useful for verification of the OCL commands.
W >START_UP.EC	c/r	Indicates a write Editor directive is being used to create the contents of the current buffer (containing OCL commands) as the file START_UP.EC residing on the system volume. NOTE: Issuance of the write editor directive, in effect, creates the file START_UP.EC.
QUIT	c/r	Designates the last directive has been entered, and control returns to the ECL (Execution Control Language) processor.
LS -PN ^Z10100>SID	c/r	Indicates use of the List Names ECL (Execution Control Language) command to verify the existence of CLM_USER file on the System Initialization Directory (SID) on system volume Z10100.
LS -PN ^Z10100 START_UP.EC	c/r	Indicates use of the List Names ECL (Execution Control Language) command to verify the existence of START_UP. EC file on the major directory on system volume Z10100.

NOTES:
1. c/r = carriage return key on the operator's terminal.
2. If the START_UP.EC file is created, it must appear in the volume major directory of system volume Zrrr00 (where: rrr is the revision number). However, if the user doesn't create his own START_UP.EC file, the Honeywell-supplied HIS>START_UP.EC file (located on the SID (System Initialization Directory) of system volume Zrrr00) is the default.
3. For a description of the Editor and Editor directives see the Program Preparation and Checkout manual.

Figure 2-2 (cont). Generalized Example of Creating and Building CLM_USER and START_UP. EC Files

```
($S)SYSTEM  0101  STARTUP
($S)(131301) 3E
($S) TYPE "CLMIN ---" TO REDIRECT INPUT,ELSE "QUIT"
QUIT
($S) CLMST1  770223 A
($S) CLMST2  770223
($S) CLMCOM  760105
($S) CLMCM2  760105
($S) CLM 770223 A  COMPLETED
($S)GROUP READY
($H)GROUP READY
 C :$H:
ED
($H)EDIT 0101
A
DEVICE DSK01,2,7,X'0480'
DEVICE DSK02,3,8,X'1200'
DEVICE DSK03,4,9,X'1280'
DEVICE LPT01,5,12,X'1380',LPT01
COMM 5
MODEM 3,X'20',X'20',X'20',X'00',X'88'
BSC 29,6,X'FD00',3,S,EB
DEVICE BTNH01,29,6,X'FD00',HOST
DEVICE BTNL01,29,6,X'FD00',HOSTA
TTY 22,8,X'FF00',2,1200
DEVICE TBCH01,22,8,X'FF00',TTY01,119
MEMPOOL S,,9500
MEMPOOL E,AB,*
!F

1,$P
($H)DEVICE DSK01,2,7,X'0480'
($H)DEVICE DSK02,3,8,X'1200'
($H)DEVICE DSK03,4,9,X'1280'
($H)DEVICE LPT01,5,12,X'1380',LPT01
($H)COMM 5
($H)MODEM 3,X'20',X'20',X'20',X'00',X'88'
($H)BSC 29,6,X'FD00',3,S,EB
($H)DEVICE BTNH01,29,6,X'FD00',HOST
($H)DEVICE BTNL01,29,6,X'FD00',HOSTA
($H)TTY 22,8,X'FF00',2,1200
($H)DEVICE TBCH01,22,8,X'FF00',TTY01,119
($H)MEMPOOL S,,9500
($H)MEMPOOL E,AB,*
W >SID>CLM_USER
1,$D

A
CB 30 -LRN 10 -LFN 8
CG AA 25 -POOL AB
EBR PROG.DEV >SPD>TTY01
EGR AA COMM.FILE1 >SPD>HOST -OUT >SPD>DSK03
EGR AA COMM.FILE2 >SPD>HOSTA -OUT >SPD>LPT01
&P ONE BATCH AND TWO GROUPS CREATED
!F
1,$P
($H)CB 30 -LRN 10 -LFN 8
($H)CG AA 25 -POOL AB
($H)EBR PROG.DEV >SPD>TTY01
($H)EGR AA COMM.FILE1 >SPD>HOST -OUT >SPD>DSK03
($H)EGR AA COMM.FILE2 >SPD>HOSTA -OUT >SPD>LPT01
($H)&P ONE BATCH AND TWO GROUPS CREATED
W >START_UP.EC
QUIT
LS -PN ^Z1D100>SID
($H)
```

Figure 2-3. Specific Example of Creating and Building CLM_USER and START_UP.EC Files

```
   DIRECTORY:  ^Z10100>SID
($H)                                STARTING  NUMBER OF  RECORD
($H)    ENTRY NAME   TYPE           SECTOR    SECTORS    LENGTH
($H)  ***************************************************
($H)    CLM          R2             382         38        100
($H)    CLMST1       R2             3BA         38        100
($H)    ZGQCDS       R2             3F2         38        100
($H)    DSK_MDC      S              42A         10        100
($H)    SIPSIM       R2             43A         38        100
($H)    SIPSIM_SP    R2             472         38        100
($H)    CLMCOM       R2             4AA         38        100
($H)    CLMST2       R2             4E2         38        100
($H)    ZQPTTY       R2             51A         38        100
($H)    ZQPVIP       R2             552         48        100
($H)    ZQPBSC       R2             59A         48        100
($H)    EMULATOR     R2             5E2         38        100
($H)    BUFM         R2             61A         38        100
($H)    ZQEXEC       R2             652         38        100
($H)    CLMCM2       R2             68A         38        100
($H)    START_UP.EC  *              6C2         10        100
($H)    CLM_USER     S              6EA          A        100
($H)  ***************************************************
LS -PN ^Z10100 START_UP.EC
($H)


   DIRECTORY:  ^Z10100
($H)                                STARTING  NUMBER OF  RECORD
($H)    ENTRY NAME   TYPE           SECTOR    SECTORS    LENGTH
($H)  ***************************************************
($H)    START_UP.EC  S              34A         10        100
($H)  ***************************************************
```

Figure 2-3 (cont). Specific Example of Creating and Building CLM_USER and START_UP. EC Files

Once the CLM_USER configuration directive file has been generated, all that remains to be done is to rebootstrap the system (see entry number 3, "Honeywell-supplied System Startup," earlier in this section). The user-defined configuration directives residing on the CLM_USER file will now be used to control the configuration and the user-defined OCL commands residing on the START_UP.EC file will be used to define the operating environment.

SYSTEM CONFIGURATION DIRECTIVES

The Configuration Load Manager (CLM) accepts configuration directives from either a Honeywell-supplied input file or a user-generated input file (CLM_USER). The configuration directives described on the following pages are meant to be used when creating the file CLM_USER to configure a specialized system.

In configuration directive parameters, unsigned positive integers can be expressed in decimal or hexadecimal notation, unless stated otherwise. A decimal integer consists of one or more decimal digits; a hexadecimal integer consists of one or more hexadecimal digits expressed in the following format: X'hexadecimal integer' (see the channel parameter in the DEVICE configuration directive for an example of the usage of the hexadecimal integer).

CLM INPUT STREAM DIRECTIVE

Directive Name: CLMIN

The CLMIN configuration directive causes the next series of configuration directives to be obtained from the specified file.

FORMAT:

CLMIN path

PARAMETER DESCRIPTION:

path

Consists of an alphanumeric (ASCII code) character string (i.e., pathname) that identifies the file from which subsequent configuration directives will be obtained. For a description of pathnames, see Section 1 in this manual.

FUNCTION DESCRIPTION:

Before using the CLMIN directive, a DEVICE directive must have been used to define the peripheral device. As soon as the CLMIN directive is processed, the input file (from which the configuration directives have been read) is transferred from the current input file to the beginning of the new specified file represented by the pathname; the next configuration directive is read from this new file.

Example:

```
CLMIN ^ABCVOL>BETA1
```

In this example, the configuration directive input stream will be transferred to file BETA1 residing on a volume called ABCVOL.

NOTE: An absolute pathname is used in this example.

DEVICE DIRECTIVE

Directive Name: DEVICE

The DEVICE configuration directive defines peripheral devices available for use by online/batch task groups.

NOTE: In a communications environment, this directive is not required if the device is to be accessed by physical I/O.

FORMAT:

DEVICE device-unit,lrn,level,X'channel',[dev-name],[record size] [,B]

PARAMETER DESCRIPTION:

device-unit

Consists of an ASCII string of up to six characters which must be unique. The last two characters of the string identify a particular unit of the device type specified by the preceding three (or four) characters shown in Table 2-3. Table 2-4 (Implicit I/O Options For A KSR) should be used in conjunction with Table 2-3.

In a communications environment, Tables 2-6 through 2-8 (Implicit I/O Options) should be used in conjunction with Table 2-5.

TABLE 2-3. AVAILABLE DEVICE TYPES

Device Type[a]	Operand Value	Default Physical Record Size (decimal)
Line Printer	LPT	72
Serial Printer	SPT	133
Card Reader	CDR	80
Diskette	DSK	128[b]
Removable Cartridge Disk	RCD	256[b]
Fixed Cartridge Disk	FCD	256[b]
Magnetic Tape (9 track)	MT9	32767
KSR[c] (Input/Output)	{KSR[d] / KBCL}	73
KSR[c](Input/Output)	KBNL	72
KSR[c]	KPL	73
KSR[c]	KDL	72

[a]Communications device types are shown in table 2-4.

[b]Cannot be altered by a specified record length.

[c]Each KSR device-type is distinguished by a different operand value indicating implicit I/O options. See Table 2-4 to determine the operand value based on the desired implicit I/O options.

[d]Either operand value can be used.

TABLE 2-4. IMPLICIT I/O OPTIONS FOR A KSR

Device Types	Input/Output Options				
KSR	CR	LF	CB	E	
{KBCL[a] / KSR}	X	X	X	X	
KBNL	X	X		X	
KPL	X		X	X	
KDL					

Legend:
CR–Carriage return option specified
LF–Line feed option specified
CB–Control byte specified
E–Echo mode specified

[a]Either device type is acceptable.

lrn

Specifies the logical resource number by which the device is requested. lrn is an integer having a decimal value of 1 through 255.

level

Specifies the logical priority level on which the device is to execute. Level is an integer having a decimal value of 5 through 60.

X'channel'

Specifies a hexadecimal number indicating the channel number of the device.

NOTE: To define the operator's terminal (KSR only) at initialization, set the channel number to a value of 0000_{16} (i.e., X'0000'). However, a corresponding decimal value of zero (0) must also be specified for the logical resource number (i.e., lrn parameter).

[dev-name]

For disk devices and magnetic tape, dev-name must either be omitted or contain an asterisk (*).

NOTE: For a description of the use of the asterisk (*), see the description of dev-name for nondisk devices which follows.

Nondisk Devices Only:

[dev-name]

Consists of an ASCII string of up to 12 alphanumeric characters, the first of which must be alphabetic. It is the unique symbolic name by which the device is referenced and, if not specified, has a default value equal to the value specified in the device-unit parameter.

If an asterisk (*) is used for dev-name, it indicates a nonsharable device to be accessed through physical I/O. As a result of the use of the asterisk, both disk and nondisk devices cannot be referenced through file management.

[record size]

Is an integer representing the physical record size of the device. See Table 2-3 (earlier in this section) and Table 2-5 (later in this section.)

[B]

Indicates that read and write commands to a file are double buffered. If this parameter isn't specified, read and write commands to a file are single buffered.

The KSR, TTY, LPT and SPT device-types (specified in the device-unit parameter) are always double buffered to accommodate possible tabulation characters.

NOTE: In a noncommunications environment, all of the prior DEVICE parameters must be unique (i.e., cannot be duplicated) on multiple DEVICE cards.

Communications Devices Only:

In a communications environment, the following restrictions apply to DEVICE parameters when used on multiple DEVICE cards:

- The level parameter can be duplicated for all communications devices.
- The level and channel parameters must be duplicated if the same device is to be accessed using two different device names (i.e., via the dev-name parameter) so that different device characteristics can be used.
- In a polled VIP environment, the channel parameter can be duplicated provided different lrn and level parameters are specified.

When using the DEVICE configuration directive in conjunction with communications configuration directives, the device-types shown in Table 2-5 should be used in the device-unit parameter of the DEVICE directive (described earlier in this section). To choose the desired device-type, the following general format must be understood:

General Format:

Columns: 1 2 3 4 5 6
Device Type: X X X X Y Y

XXXX–Indicates the columnar position of alphabetic characters constituting the desired device type. Interpretation of the following characters, shown in column positions 1 through 4 of the device types in Table 2-5, enables choice of the proper device type:
RTY–72 column ROP (columns 1 to 3)
RTN–120 column ROP (columns 1 to 3)
B–BSC line protocol handler if in column 1; indicates bidirectional device if in column 2.
T–TTY line protocol handler if in column 1; indicates transparent mode (BSC) if in column 2 (except for RTNH, RTNL, RTYH or RTYL).
V–VIP line protocol handler
C–Control byte option specified
N–No control byte specified
P–Printer emulation
D–Data entry
H–Hang-up phone option specified on disconnect
L–Leave phone connected option specified on disconnect

TABLE 2-5. COMMUNICATIONS DEVICE TYPES FOR DEVICE DIRECTIVES

Device Types					
TTY	**Default Record Size**	**VIP**	**Default Record Size**	**BSC**	**Default Record Size**
Columns (1-4) XXXX		Columns (1-4) XXXX		Columns (1-4) XXXX	
TBCH	73[a]	VBCH	81[a]	BBCH	137[a]
TBCL	73[a]	VBCL	81[a]	BBCL	137[a]
TBNH	72	VBNH	80	BBNH	136
TBNL	72	VBNL	80	BBNL	136
TPH[b]	73[a]	VPH[b]	81[a]	BTNH	136
TPL[b]	73[a]	VPL[b]	81[a]	BTNL	136
TDH	72	RTNH[b]	119[a]		
TDL	72	RTNL[b]	73[a]		
		RTYH[b]	73[a]		
		RTYL[b]	73[a]		

[a]Includes control byte
[b]Output only device type

YY–Indicates any pair of alphabetic, numeric or alphanumeric characters used to distinguish a particular device from another device having the same device type (e.g., TBCH01 and TBCH02 or TBCHAA and TBCHAB).

Tables 2-6, 2-7 and 2-8 show the implicit input/output options available for the following three line protocols BSC, TTY and VIP.

TABLE 2-6. IMPLICIT I/O OPTIONS FOR A BSC

Device Types	Input/Output Options				
BSC	ETB	TB	CB[a]	PH	QA
BBCH			X	X	X
BBCL			X		X
BBNH	X			X	X
BBNL	X				X
BTNH	X	X		X	X
BTNL	X	X			X

Legend:
ETB–Send ETB
TR–Send transparent mode
CB–Leading control byte option specified
PH–Physical disconnect option specified
QA–Queue abort option specified
EOT–Send EOT bit set

[a]Leading control bytes allow settings of EOT, ETB/ETX and TR.

NOTES: 1. The two-buffer option cannot be specified as it is not possible through the file system.
2. The sending of RV1 is not possible through the file system.

TABLE 2-7. IMPLICIT I/O OPTIONS FOR A TTY

Device Types	Input/Output Options						
TTY	CR	LF	E	TR	CB	PH	QA
TBCH	X	X	X		X	X	X
TBCL	X	X	X		X		X
TBNH	X	X	X			X	X
TBNL	X	X	X				X
TPH	X		X		X	X	X
TPL	X		X		X		X
TDH				X		X	X
TDL				X			X

Legend:
CR–Trailing carriage return option specified
LF–Trailing line feed option specified
E–Echo mode specified
TR–Transparent mode specified
CB–Leading control byte specified
PH–Physical disconnect specified
QA–Queue abort specified

TABLE 2-8. IMPLICIT I/O OPTIONS FOR A VIP

Device Types	Input/Output Options								
VIP	CR	LF	CB	PH	QA	PO	TIM	OSR[a]	POLL
VBCH	X	X	X	X	X	X	X		X
VBCL	X	X	X		X	X	X		X
VBNH	X	X		X	X	X	X		X
VBNL	X	X			X	X	X		X
VPH	X		X	X	X				
VPL	X		X		X				
RTYH	X		X	X	X				
RTYL	X		X		X				
RTNH	X		X	X	X				
RTNL	X		X		X				

[a]Available only through physical I/O.

Legend:

CR—Carriage return option specified
LF—Line feed option specified
CB—Leading control byte specified
PH—Physical disconnect specified
QA—Queue abort specified
PO—Page overflow recovery specified
TIM—Timeout on read specified
OSR—One shot read specified
POLL—Poll interval of one second (ignored if nonpolled line)

FUNCTION DESCRIPTION:

Each device to be used must be explicitly defined in a DEVICE configuration directive. The device-type operand is a generic name, and there may be more than one device of the same type (e.g., two diskettes); however, the level and channel operands must be unique for each device. The DEVICE configuration directive also creates a system task for the driver of the specified noncommunications device type. The lrn operand specifies the logical resource number and must be unique.

In a communications environment, a DEVICE configuration directive must be used in conjunction with either a TTY, VIP or BSC communications configuration directive to define either a TTY, VIP or BSC to file management. The lrn, level and channel parameters must be duplicated on the DEVICE configuration directive exactly as indicated in either a TTY, VIP, or BSC communications configuration directive.

Example 1 (Noncommunications Environment):

```
DEVICE LPT01,12,20,X'1380'
```

In this example, the line printer is the device-type having an assigned unit number of Ø1, a logical resource number of 12 and a priority level of 20. The hexadecimal channel number of 1380 for the line printer was assigned when the hardware was installed. The following default assignments will be defined for the remaining parameters:

- o The dev-name (device name) will be the same as the device unit (i.e., LPT01).
- o The record size will be 137_{10}.
- o Single buffering will be used.

Example 2 (Communications Environment):

```
DEVICE TBCH01, 21,8,X'FF80',TTY1
```

In this example, the device-unit parameter TBCH indicates the following:

- o T–The device-type is a TTY indicating that this DEVICE configuration directive would be used in conjunction with a TTY communications configuration directive.
- o B–Indicates a bidirectional device.
- o C–The control byte option is in effect.
- o H–The hang-up phone option is to be allowed on a disconnect.

The remainder of the parameter string shows a logical resource number of 21 and a priority level of 8. The hexadecimal channel number of FF80 is assigned because it is known as the hardware channel number (i.e., installation-specific) for a bidirectional device. TTY1 is the dev-name with a default physical record size of 73_{10} (see Table 2-5 earlier in this section).

LOAD BOUND UNIT DIRECTIVE

Directive Name: LDBU
The LDBU configuration directive defines a bound unit name to be added to the end of the bound unit list, and specifies that this bound unit is to be made resident.

NOTE: Any bound unit loaded by the LDBU configuration directive must have been linked with a SYS linker directive.

FORMAT

```
LDBU path
```

PARAMETER DESCRIPTION:

path
Is the pathname of the bound unit to be permanently loaded. It consists of an alphanumeric character (ASCII) string that identifies a file when it is opened. For a description of pathnames, see Section 1 in this manual.

FUNCTION DESCRIPTION:
Application-specific code (usually in the form of subroutines shared among multiple task groups) that is referenced symbolically during application execution is brought into memory permanently by an explicit LDBU configuration directive. The LDBU configuration directive pathname is added to a temporary bound unit list. If more than one LDBU configuration directive is specified, the order of the directives determines the order in which the pathnames are added to the end of the bound unit list. If a pathname is already listed, the duplicate LDBU directive is ignored.

Subsequently, after a QUIT configuration directive, the roots of the bound units identified by pathname are loaded into memory permanently and then symbol tables are added to the resident system symbol table list. During the loading process, unresolved symbols in subsequent bound units are resolved if they have been previously defined through use of the EDEF linker directive. If unresolved symbols have not been defined, an error is generated and the loading process is prevented.

Example:

```
LDBU  ^ABCVOL>ALPHA1
```

In this example, a bound unit named ALPHA1, residing on a volume called ABCVOL, will be added to the end of the bound unit list.

Bound units loaded using the LDBU configuration directive may contain an Initialization Subroutine Table (IST) if the code associated with the bound units does not:

- o Attempt to define system symbols
- o Extend the memory area required by the root segment beyond the memory area specified when the bound unit was loaded
- o Memory used by the IST code may be returned to the operating system when the IST code completes

For a description of the IST, see the Program Preparation and Checkout manual.

MEMORY POOL DIRECTIVE

Directive Name: MEMPOOL

The MEMPOOL configuration directive permits the user to define memory pools (i.e., memory areas) required by the system. Each MEMPOOL configuration directive defines one pool set.

NOTE: To properly use the MEMPOOL configuration directive, it is necessary to first reference the description of memory pools in the Overview and User's Guide manual.

FORMAT 1:

MEMPOOL $\left\{\begin{matrix}S\\B\end{matrix}\right\}$,,size

FORMAT 2:

MEMPOOL$\left\{\begin{matrix}E\\null\end{matrix}\right\}$,pool name,size$\left\{\begin{matrix},X\\null\end{matrix}\right\}$ $\left[\right.$,pool name,size$\left\{\begin{matrix},X\\null\end{matrix}\right\}\left.\right]$ ...

PARAMETER DESCRIPTION:

$\left\{\begin{matrix}S\\B\end{matrix}\right\}$

Represent the pool types for format 1 and are defined as follows:

S - System pool; if not specified, a default system pool size of 3K is created.

A separate MEMPOOL configuration directive must be used for the S pool type being specified. If previous S pool types have been specified, each on a separate MEMPOOL configuration directive, the last S pool type specified replaces any of the previous ones.

With the S pool type, it is not necessary to assign a pool name; if one is assigned, it is ignored.

NOTE: An execution control language (ECL) reference to the system pool requires $S to be used as the pool name.

B – Batch pool

The B pool type can only be specified once on a separate MEMPOOL configuration directive; if a second occurs, an error is generated (see "Configuration Load Management Error Messages," Section 5).

With the B pool type, it is not necessary to assign a pool name; if one is assigned, it is ignored.

{E / null(no-entry)}

Represent the pool types for format 2 and are defined as follows:

E – One or more exclusive pools

The E pool type is only specified once on a separate MEMPOOL configuration directive. Multiple definition of *only* the pool name, size and, optionally, X or null (no entry) parameters indicates each additional exclusive pool being defined which, in total, constitute a pool set.

If more than one exclusive pool is specified in the same directive, the additional exclusive pools occupy contiguous physical memory areas.

null – Nonexclusive (i.e., shareable) pools

The null (or no entry) parameter is only specified once per MEMPOOL configuration directive. However, more than one MEMPOOL configuration directive can be used. Multiple definition of *only* the pool name, size and, optionally, X or null parameters indicates each additional nonexclusive pool which, in total, constitute a pool set.

Multiple nonexclusive pool sets, on multiple MEMPOOL configuration directives, establish alternative (i.e., overlapping) definitions of the same physical memory area.

size -

Represents the pool size used in both formats 1 and 2 and is defined as follows:

Is either a positive integer representing the size (in number of 16-bit words) of the pool or an asterisk (*) character. Pool sizes are rounded up to the next multiple of 32.

If an asterisk (*) character is used, the length of the pool is extended either to the beginning of high memory or the batch pool area (if it has been defined). There can be no additional pools in the pool set, otherwise an error is generated (see "Configuration Load Management Error Messages," Section 5). Furthermore, the asterisk (*) character cannot be used with either the B (batch) or S (system) pool types.

The following parameters are *only* used in format 2 and are defined as follows:

pool name-

Is a two-character ASCII pool name indicating a particular memory pool. This name is the one used in either a subsequent CG (Create Group) OCL or ECL command. If the pool name begins with a decimal digit, the pool name parameter must be surrounded by an apostrophe (e.g., '1A', not 1A).

[{,X / null}]

X indicates the batch pool is to be rolled out of memory, if a task running in the memory pool has insufficient memory, resulting in a logical extension of the pool to which this parameter is being applied.

null (no entry) indicates a pool cannot logically extend into the batch pool area.

FUNCTION DESCRIPTION:
When the MEMPOOL configuration directive is interpreted, a pool descriptor list is created. Then, after memory requirements for the operating system and its extensions are known, each pool set is checked to determine if it is too large to be contained in the remaining available memory pool area. If the pool sets can be contained in the available pool area, a pool descriptor is created for each member of the pool set; if the pool sets are too large, an error is generated (see "Configuration Load Management Error Messages," Section 5).

The following four examples of the MEMPOOL configuration directive show the following:

Example 1:

```
MEMPOOL S,,4096
```

In this example, a system memory pool is defined having a pool size containing 4096_{10} 16-bit words. Because a system pool is being defined, it is not necessary to designate a pool name.

Example 2:

```
MEMPOOL B,,12736
```

In this example, a batch pool is defined having a pool size containing 12736_{10} 16-bit words. Because a batch pool is being defined, it is not necessary to designate a pool name.

Example 3:

```
MEMPOOL E,AB,2048,,CD,1024,X
```

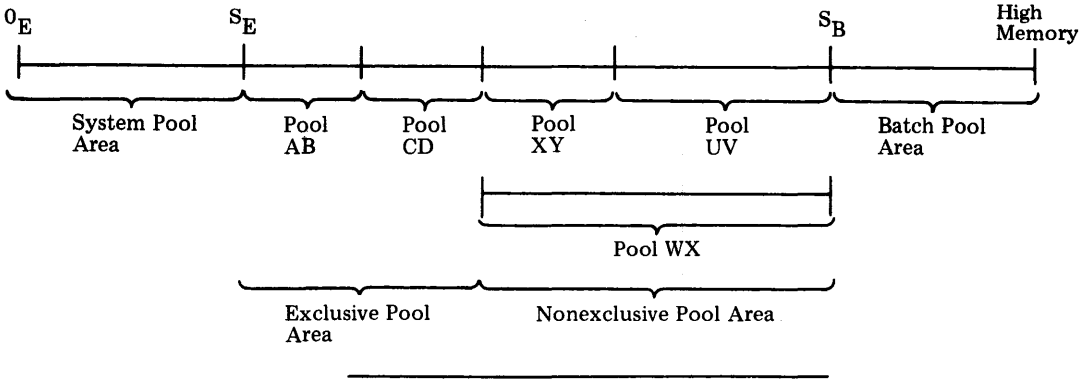
In this example, two exclusive pools are being defined in the online pool area. The first exclusive pool has a pool name AB and a pool size containing 2048_{10} 16-bit words; it cannot extend into the batch pool area (i.e., parameter 4 is null). The second exclusive pool has a pool name CD' a pool size containing 1024_{10} 16-bit words, and may extend into batch pool area (i.e., parameter 7 is X).

Example 4:

```
MEMPOOL ,XY,512,,UV,*
MEMPOOL ,WX,*,X
```

In this example, two sets of nonexclusive pools (i.e., parameter 1 is null in each directive) are defined. Because these two sets are nonexclusive, pools XY and UV may be overlapped by pool WX or vice versa; any contention for space is resolved by the system. Pool XY has a size of 512_{10} 16-bit words and cannot extend into the batch pool area (i.e., parameter 4 in the first MEMPOOL directive is null). Pool UV has a size that extends from the end of pool XY to either the beginning of the batch pool area (i.e., the asterisk (*) parameter), if defined, or high memory; it cannot extend into the batch pool area (i.e., parameter 7 is missing). Pool WX utilizes all the area beginning at the end of pool CD (i.e., the last exclusive pool defined in example 3) to either the beginning of the batch pool area, if defined, or high memory; it may extend into the batch pool area (i.e., parameter 4 is X indicating rollout).
Combining the five pool sets in these examples, the following memory pool layout would occur.

MEMORY POOL DIRECTIVE



RESIDENT OVERLAY DIRECTIVE

Directive Name: RESOLA

The RESOLA configuration directive can be used to define only monitor and input/output overlays to be permanently resident in memory for the duration of the configured system.

FORMAT:

RESOLA overlay name [,overlay name]...

PARAMETER DESCRIPTION:

overlay name-

Consists of up to six alphanumeric characters indicating the name of the system overlay; the first character must be alpha. Additional overlay names can be specified in the same RESOLA configuration directive.

NOTE: See Table 2-9 for the names of the system overlays which can be defined as permanently resident through use of the RESOLA configuration directive.

FUNCTION DESCRIPTION:

When using the RESOLA configuration directive, the system symbol table is searched for the symbolic overlay name. If the overlay name is not found, an error is generated (see "Configuration Load Management Error Messages," Section 5).

During the loading phase of the Configuration Load Manager (CLM), the specified system overlay is loaded; this procedure is repeated for each overlay name in the RESOLA configuration directive line. Any error causes an overlay name to be ignored; however, other names in the directive line are processed.

Example:

RESOLA OUERS,OXWATL,ZYOV12,ZYOV23

In this example, system overlays OUERS,OXWATL,ZYOV12 and ZYOV23 will be added to the resident overlay list and are loaded during the loading phase of the CLM.

TABLE 2-9. SYSTEM OVERLAYS

Overlay Name	Description
Executive Services	
OXPCL	Process command line
OXCTSK	Create and spawn task
OXDTSK	Delete task
OXCGRP	Create and spawn group
OXCGCT	Create and spawn group
OXRQGP	Request and spawn group
OXGRQS	Start group request
OXGRQT	Terminate group request
OXDLGP	Delete and abort group (and abort group request)
OXCDRV	Create driver
OUERS	Error reporter
OXC_XD	Convert internal to external date/time
OXC_IT	Convert external to internal date/time
OXSI01	Standard I/O (new user, COMMAND_IN)
OXSI02	Standard I/O (new user, USER_OUT)
OIOIM0	OIM input processor
OIOIM1	OIM command processor (output completion)
OIOIM2	OIM output control
OIOIM3	OIM diagnostics
OIOIM4	Abort group request OIM purge
OXAVR1	Automatic volume recognition
OXGSSP	Group suspend (rollout phase 1)
OXGACT	Group activate (rollout phase 2)
OXTRAP	Connect user trap handler, enable, disable, user traps
OXWATL	Wait list processing
File Management Services (Disk and Unit Record)	
ZYOV0	Associate-/dissociate-file
ZYOV1	Close-file (sequential, relative and indexed)
ZYOV2	Close-file (nondisk and nontape)
ZYOV3	Create-/release-directory
ZYOV4	Create-/release-file
ZYOV5	Allocate extent (part 1)
ZYOV6	Allocate extent (part 2)
ZYOV7	Create-file (build directory entries)
ZYOV8	Create-file (check for file already in existence)
ZYOV9	Create-file (delete directory entries for indexed data file)
ZYOV10	Create-file (temporary file)
ZYOV11	Create-file (calculate extent information for indexed file)
ZYOV12	Change-working-directory
ZYOV13	Get-file-information

TABLE 2-9 (CONT). SYSTEM OVERLAYS

Overlay Name	Description
ZYOV14	Get-file
ZYOV15	Get/create FCB (file control block)
ZYOV16	Get/create FDB (file descriptor block)
ZYOV17	Get RXB (remote extent block)
ZYOV18	Initialize FDB
ZYOV19	Open-file (all files except tape)
ZYOV20	Open-file (access method open for sequential, relative and indexed)
ZYOV21	Open-file (Access method open for device files)
ZYOV22	Deallocate file extents
ZYOV23	Remove-file (remove FCB)
ZYOV24	Remove-file (remove FCB)
ZYOV25	Rename-file
ZYOV26	Volume level access
ZYOV27	Volume mount request (disk/diskette)
OYVMNT	Volume mount routine (disk/diskette)
ZYOV28	Volume mount message handler
ZYOV29	Expand pathname and get working directory
ZYOV30	I/O error processing
Relative Files Data Management	
ZYOV40	Fixed-relative (delete, read, rewrite, write)
ZYOV41	Unified relative (delete, read, rewrite, write)
Sequential Files Data Management	
ZYOV50	Delete-record Rewrite-record Position functions
Indexed Files Data Management	
ZYOV60	Delete-record
ZYOV61	Read-record
ZYOV62	Rewrite-record
ZYOV63	Write-record (insert)
ZYOV64	Write-record (load mode)
ZYOV65	Close-file (build index if load mode)
ZYOV66	Describe general overflow (create inventory entries)
ZYOV67	Update inventory
ZYOV68	Search overflow (insert)
ZYOV69	Search index
ZYOV70	Indexed file common subroutines
ZYOV71	Update index (load mode)

TABLE 2-9 (CONT). SYSTEM OVERLAYS

Overlay Name	Description
BES Accommodation File and Data Management	
ZYOV80	Establish attributes (get-file)
ZYOV81	Create file
ZYOV82	Temporary file creation
ZYOV83	File maintenance
ZYOV84	Open
ZYOV85	Positioning
ZYOV86	Read record
ZYOV87	Rename
ZYOV88	Set file status (R7)
ZYOV89	Write record
ZYOV8A	MCL interface
Tape Files Storage Management	
ZYOV90	Tape storage management (positioning functions)
ZYOV91	Tape file (open)
ZYOV92	Tape file (close)
ZYOV93	Read tape
ZYOV94	Write tape
ZYOV95	Input processing mode
ZYOV96	Output processing mode
ZYOV97	Label processing (output)
ZYOV98	Conversion utility routine
ZYOV99	Interface routine and storage manager
ZYOV9A	Label processing (input)
Tape Volume Mount	
ZYOV9B	Volume mount/dismount routine
ZYOV9C	Volume mount/dismount request

SYSTEM DEFINITION DIRECTIVE

Directive Name: SYS
The SYS configuration directive defines system variables.

FORMAT:

SYS [Hz],[scan-cycle],[{SSIP / DSIP / null}],[olan],[tsa],[irb]

PARAMETER DESCRIPTION:

[Hz]

Specifies the line frequency used to drive the system clock. Possible values are 60 (for 60 Hz) or 50 (for 50 Hz). The default value is 60 Hz (i.e., the U.S. standard).

[scan-cycle]

Specifies the time, in milliseconds, between periodic real-time clock-generated interrupts. The default value is milliseconds.

The following lists show the possible values of the scan-cycle for both line frequencies:

50-Hz Line (milliseconds)	60-Hz Line (milliseconds)
10	8
20	16
50	25
100	33
	50
	100

Since the system clock is activated to service executive timer requests after each scan-cycle, the scan-cycle should be carefully chosen to minimize the Clock Manager interrupt service overhead while providing the timer request resolution the application requires.

Note: System clock resolution is accurate to ± one scan cycle.

[{ SSIP / DSIP / null (no entry) }]

Specify usage of single- and double-precision scientific instruction in an application. These instructions are interpreted by either a software or hardware scientific instruction processor which processes either single-precision (SSIP) or double-precision (DSIP) instructions.

When either (or both) the hardware or software scientific instruction processors are available, the following occurs:

- If the hardware scientific instruction processor is not available, a software scientific instruction processor (i.e., simulator) is loaded to process the scientific instructions.
- If the hardware scientific instruction processor is available the scientific instructions trap to the hardware processor and a software processor (i.e., simulator), is not made available.

SSIP

Indicates single-precision scientific instructions to be processed.

NOTE: Only single-precision is BES2 compatible.

DSIP

Indicates double-precision scientific instructions are to be processed.

NOTE: Usage with MDT only.

null

Indicates either no scientific instructions are present or, if present, they will be processed by the hardware scientific instruction processor if available at the user installation.

The length of the task control block (TCB) is modified by the length of the interrupt save area (ISA) being modified by usage of either the SSIP, DSIP or null (no entry) parameters as shown in Table 2-10.

[olan]

Specifies the number of 256-word overlay areas to be created. olan is an integer from 1 to 10. The default value is 1 indicating one overlay area for the system.

[tsa]

Specifies the number of additional Trap Save Areas (TSAs) to be configured. The size of each additional TSA is the same as those already defined for the operating system. The default number of TSAs for the operating system is 12.

[irb]

Specifies the desired number of additional intermediate request blocks. The default number of IRBs for the operating system is 20.

TABLE 2-10. ISA MODIFICATION BASED ON USAGE OF SCIENTIFIC INSTRUCTIONS

Hardware / Software	Model 6/30 (without SIP)	Model 6/40 (without SIP)	Model 6/40 (with SIP)
null (no entry)	ISA ends inclusive of register M1. TCB (including ISA) is 64 words.	ISA ends inclusive of registers M1, M2 through M7, and the stack register. TCB (including ISA) is 64 words.	ISA ends inclusive of registers M1, M2 through M7, and the stack register. TCB (including ISA) is 64 words.
SSIP	ISA ends inclusive of register M1. TCB (including ISA) is 64 words.	ISA ends inclusive of registers M1, M2 thorugh M7, and the stack register. TCB (including ISA) is 64 words.	ISA ends inclusive of registers M1, M2 through M7, stack register, and the SIP context. TCB (including ISA) is 96 words.
DSIP	ISA ends inclusive of registers M1, M2 through M7, stack register and the SIP context. TCB (including ISA) is 96 words.	ISA ends inclusive of registers M1, M2 through M7, stack register, and the SIP context. TCB (including ISA) is 96 words.	ISA ends inclusive of registers M1, M2 through M7, stack register, and SIP context. TCB (including ISA) is 96 words.

NOTES: 1. The length of the registers shown in this table is:

a. M1 (one word)
b. M2 through M7 (six words)
c. Stack register (one word)
d. SIP context (13 words)

2. The TCB (task control block) has a base length of 50 words. The length of the registers (terminating the ISA) shown in note 1 is added to the base length of the TCB to produce a total size for the TCB of either 64 words or 96 words; in each case, the total size for the TCB is produced from rounding up the sum (produced by the previous addition) by increments of 32 (i.e., 32 + 32 = 64 + 32 = 96).

Example:
Assumed (from Table 2-10): No SIP, Model 6/30 (without SIP)
Summation: Base = 50 (words)
Register M1 = 1 (word)
Rounded: The number 51 (i.e., sum) is rounded up to 64

FUNCTION DESCRIPTION:
The SYS configuration directive is used to define hardware- and/or software-related options to be included as part of the operating system. If all of the default values are to be chosen, the SYS configuration directive need not be used . If multiple SYS configuration directives are issued, the last one is effective. Null parameters in the parameter string indicate default values.

Example:

SYS 50,10,SSIP

In this example, the following occurs:

- A line frequency of 50 is used.
- A time of 10 milliseconds will elapse between real-time clock-generated interrupts.
- The scientific instruction processor (SSIP) has been specified for single-precision instructions.
- One (i.e., 1) overlay area for the system has been specified by default.
- No additional trap save areas (TSAs) will be obtained.
- No additional intermediate request blocks (IRBs) will be obtained.

QUIT DIRECTIVE

Directive Name: QUIT
The QUIT configuration directive is the last configuration directive in the user input file.

FORMAT

QUIT

PARAMETER DESCRIPTION:
Not applicable

FUNCTION DESCRIPTION:
When this directive is encountered, the CLM stops reading directives from the input file and initiates the loading phase. As required: any final data structures are created, the communications system is initialized, bound units are loaded (as defined by LDBU configuration directives), system overlays become resident (as defined by RESOLA configuration directives), memory pool descriptors are created (as defined by MEMPOOL configuration directives), and the CLM task terminates.

COMMUNICATIONS CONFIGURATION DIRECTIVES

The communications configuration directives perform the following functions:

- Establish data structures (i.e., tables) corresponding to the communications hardware available at the user's installation
- Load the following bound units into memory:
  - Communications supervisor and Multiline Communications Processor (MCP) driver
  - One or more line protocol handlers (i.e., TTY, VIP, BSC, or user-written)
- Load the following into the memories of one or more multiline communications processors:
  - Data set channel control program
  - Channel control programs (up to three per MCP) of one or more line protocol handlers
  - Line control tables

COMMUNICATIONS SYSTEM DIRECTIVE

Directive Name: COMM
The COMM communications configuration directive is mandatory for systems with communications and specifies the interrupt priority level for all communications devices. It must precede the following communications configuration directives: TTY, VIP, BSC, LPHn and STATION.

FORMAT:

COMM interrupt level

PARAMETER DESCRITION:

interrupt_level
Specifies the priority level at which a multiline communications processor (MCP) interrupts the central processing unit. interrupt_level is an integer from 5 to 60; it must be less than the levels specified for the communications devices in the communications directives which follow the COMM communications directive.

Example:

COMM 5

In this example, the communications interrupt level is 5.

NOTE: The levels that the communications supervisor uses to process input/output requests to communications devices are specified in the TTY, VIP, LPHn and BSC communications directives later in this section.

MODEM DEFINITION DIRECTIVE

Directive Name: MODEM
The MODEM communications configuration directive is used to define a nonstandard modem type. The information provided in this command is used to test entries in the LCT (line control table) for the device to verify a connection or disconnection.

FORMAT:

MODEM type-number,connection-AND-mask, connection-XOR-mask,disconnection-AND-mask,disconnection-XOR-mask,data-set-control

PARAMETER DESCRIPTION:

type-number
An integer from 3 to 15 that is assigned to this modem definition and may then be used in a communications device directive (i.e., such as, TTY, VIP, BSC, and LPHn directives).

connection-AND-mask
A two-digit hexadecimal number whose value determines which bits (i.e., 0 through 4) of LCT byte 14 (receive channel data set status) and byte 46 (transmit channel data set status) will be examined when a connect request is processed.

connection-XOR-mask
A two-digit hexadecimal number whose value specifies which bits (i.e., 0 through 4) of LCT bytes 14 and 46 must be on (i.e., set to 1) for a connection.

disconnection-AND-mask

A two-digit hexadecimal number whose value determines which bits (i.e., 0 through 4) of LCT bytes 14 and 46 will be examined when a disconnect request is processed, or when a test for the occurrence of a disconnect is made.

disconnection-XOR-mask

A two-digit hexadecimal number whose value determines which bits (i.e., 0 through 4) of LCT bytes 14 and 46 must be on (i.e., 1_2) for a disconnection. (Entries 14 and 46 of the LCT are the data set status for the receive and transmit channels respectively.)

data-set-control

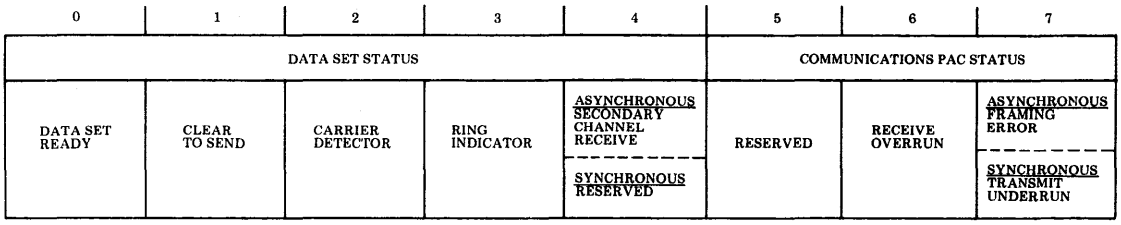
A two-digit hexadecimal number loaded into byte 20 of the LCT and line register 2 (LR2) of the communication line adapter (CLA) when a line is to be connected.

NOTES:
1. To test for a successful *connection,* bytes 14 and 46 of the LCT are first subjected to a logical AND operation against the (user-supplied) connection-AND-mask; then a logical exclusive OR operation is performed on the result of the first operation, against the (user-supplied) connection-XOR-mask. If the result is zero, a connection has been established.
2. To test for a *disconnection,* the same operations are carried out using the analogous disconnection masks. A zero result indicates a disconnection.
3. The following shows the mask and data set control values for the standard CLM-recognized modem types:

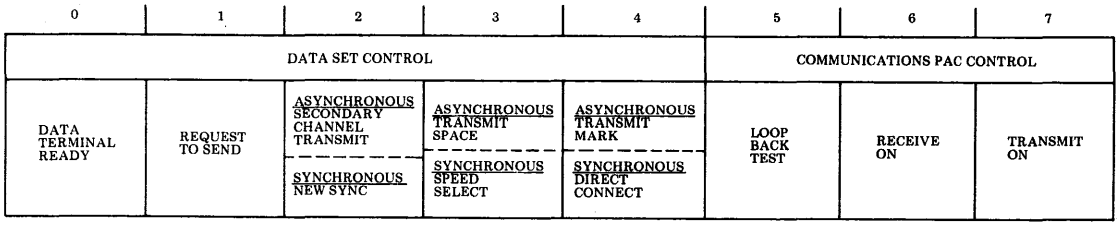
Modem Type	Type Number	Connection Masks AND	Connection Masks XOR	Disconnection Masks AND	Disconnection Masks XOR	Data Set Control
Direct Connect	0	X'80'	X'80'	X'80'	X'00'	X'88'
Bell 1xx	1	X'A0'	X'A0'	X'A0'	X'00'	X'80'
Bell 2xx	2	X'80'	X'80'	X'80'	X'00'	X'80'

LTC bytes 14/46 and 20/52 are shown below; see Section 5 of the Series 60 (Level 6) MLCP Programmer's Reference manual, Order No. AT97 for a detailed description of the bytes.

LCT Byte 14/46:



LCT Byte 20/52:



Example:

MODEM 3,X'20',X'20',X'20',X'00',X'88'

In this example, a modem type requiring only the carrier detector signal for a connection and lack of this signal for a disconnection is defined. The actual bit settings (i.e., bits 0 through 4) of LCT bytes 14/46, resulting from the hexadecimal values of the connection/disconnection AND-mask and XOR-mask parameters shown in this example of the MODEM communications directive, are as follows:

Connect-AND-Mask (X'20')

Bits:	0	1	2	3	4	5	6	7
	0	0	1	0	0	0	0	0

Connect-XOR-Mask (X'20')

Bits:	0	1	2	3	4	5	6	7
	0	0	1	0	0	0	0	0

Disconnect-AND-Mask (X'20')

Bits:	0	1	2	3	4	5	6	7
	0	0	1	0	0	0	0	0

Disconnect-XOR-Mask (X'00')

Bits:	0	1	2	3	4	5	6	7
	0	0	0	0	0	0	0	0

LINE PROTOCOL HANDLER DEFINITION DIRECTIVE

Directive Name: LPHDEF

For each (line protocol handler) you write, you can include an LPHDEF communications configuration directive to define the sizes of tables used for the channels and stations controlled by the line protocol handler. If the LPHDEF directive is not included when you write a line protocol handler, then channel table and station table default sizes will be used for channels and stations controlled by the line protocol handler (see channel-table-size and station-table-size parameters below).

FORMAT:

LPHDEF lph,[channel-table-size] [,station-table-size]

PARAMETER DESCRIPTION:

lph

Consists of a number from 0 to 3 identifying your LPH. An LPH directive (described later in this section) may not precede the corresponding LPHDEF directive.

[channel-table-size]

Specifies the number of words needed for the channel table and the CQBs (communication queue blocks). It must have a value of at least 10 words; the default value is 33 words.

[station-table-size]

Specifies the number of words needed for this LPH's station table (resource control table). It must have a value of at least 10 words; the default value is also 10 words.

Example:

```
LPHDEF  0,X'30'
LPH0  27,8,X'FD80',,,FDX,0
STATION  28,1
```

In this example, line FD80 has two synchrnous full duplex terminals controlled by a user-written line protocol handler. Each of the two-channel tables for the line has a size of 30 words as defined by the channel-table-size (i.e., X'30') parameter in the LPHDEF communications configuration directive.

BINARY SYNCHRONOUS COMMUNICATIONS DIRECTIVE

Directive Name: BSC
The BSC communications configuration directive identifies each binary synchronous communications line included in the system.

FORMAT:

```
BSC  lrn,level,X'channel' [,modem] [,primary/secondary] [,character-set]
```

PARAMETER DESCRIPTION:

lrn

The logical resource number associated with the device. lrn is an integer having a decimal value of 1 through 255. A program may use this number to identify the device when it requests an input/output operation to the device.

level

The priority level at which the communications supervisor processes requests for an input/output operation on the device. level is an integer from 6 to 60; it may be the same as the level specified for other communications devices, but it must be greater than the communication interrupt level specified in the COMM directive (described earlier in this section). The level specified for one or more communications devices may not also be used for noncommunications devices or tasks.

X'channel'

Is a four-digit hexadecimal number (from X'0400' to X'FF80') specifying the channel number of the device and has the following format:

Bits 0 through 5—The six-bit number of the MCP on the bus
Bits 6 through 9—The four-bit even number of the receive channel of one of the eight lines of a MCP (multiline communications processor)
Bits 10 through 15—Are set to zero

modem

A number specifying the type of data set. Possible values are:
0—Direct connect.

1–Bell 1xx-type modem (103A, 113F, etc). Both data-set-ready and carrier-detect signals are needed for a connection; lack of both signals is a disconnection.

2–Bell 2xx-type modem (201A,201C,208A,etc). The data-set-ready signal is needed for a connection; lack of this signal is disconnection.

3 to 15–User-defined modem type (see MODEM communication configuration directive earlier in this section).

NOTE: The default value is modem type 2.

primary/secondary

Values may be specified as P or S; indicates whether this is the primary or secondary endpoint of the transmission. A primary endpoint has priority in contention mode.

character set

One of the following may be specified:
AS–ASCII (default)
EB–EBCDIC

Example:

```
MODEM 3,X'20',X'20',X'20',X'00',X'88'
BSC 29,6,X'FD00',3,S,EB
DEVICE BTNH,29,6,X'FD00',HOST
DEVICE BTNL,29,6,X'FD00',HOSTA
```

In this example, line FD00 is used for communication with another computer; it uses the modem type defined in the MODEM directive. The level 6 computer is the secondary endpoint on the line, and two files are defined for the line.

When the file HOST is closed, the phone can be hung-up. When the file HOSTA is closed, the phone is left connected.

DEVICE configuration directives are used in conjunction with the BSC communications directive if the terminal is to be referenced as a file (e.g., in a higher level language).

LINE PROTOCOL HANDLER DIRECTIVE

Directive Name: LPHn

The LPHn directive identifies a communications device driven by a user-written line protocol handler.

FORMAT:

```
LPHn lrn,level,X'channel' [,modem] [,speed] [,FDX/HDX] [,lph-specific-word]
```

n–Consists of the integer 0, 1, 2, or 3. n matches the first parameter of the LPHDEF communications configuration directive used for the line protocol handler which drives the device.

PARAMETER DESCRIPTION:

lrn

The logical resource number associated with the device. lrn is an integer having a value of 1 through 255. A program may use this number to identify the device when it requests an input/output operation to the device.

level

The priority level at which the communications supervisor processes requests for an input/output operation on the device. level is an integer from 6 to 60; it may be the same as the level specified for other communications devices, but it must be greater than the communication interrupt level specified in the COMM communications configuration directive (described earlier in this section).

The level specified for one or more communications devices may not also be used for noncommunications devices or tasks.

X'channel'

Is a four-digit hexadecimal number (from X'0400' to X'FF80') specifying the channel number of the device and has the following format:
Bits 0 through 5—The six bit number of the MCP (multiline communications processor) on the bus.
Bits 6 through 9—The four-bit even number of the receive channel of one of the eight lines of a MCP.
Bits 10 through 15—Are set to zero

[modem]

A number specifying the type of data set. Possible values are:
0—Direct connect.
1—Bell lxx-type modem (103A,113F,etc). Both data-set-ready and carrier-detect signals are needed for a connection; lack of both signals is a disconnection.
2-Bell 2xx-type modem (201A, 201X, 208A, etc). The data-set-ready signal is needed for a connection, lack of this signal is disconnection.
3 or greater—User-defined modem type (see MODEM communication configuration directive earlier in this section). The default value is modem type 2.

NOTE: If the line is direct connect and asynchronous, modem type 2 must be specified; if the line is direct connect and synchronous, specify modem type 0.

[speed]

The data rate in bits per second. The default value is zero, and signifies a synchronous line.
For an asynchronous line with a line adaptor whose id has a value of 2108_{16} (i.e., X'2108'), use the following values for speed:

	50	300	2400
	75	600	3600
(default)	110	900	4800
	134	1200	7200
	150	1800	9600

For an asynchronous line with a line adaptor whose id has a value of 2118_{16} (i.e. X'2118'), use the following values for speed:

	50	200	1800
	75	300	2000
(default)	110	600	2400
	134	1050	4800
	150	1200	9600

NOTE: If the data rate of the line is 134.5, specify 134.

[FDX/HDX]

Specifies whether the line is full- or half-duplex. If it is full-duplex (FDX), two channel tables will be assigned. The default value is HDX.

[lph-specific-word]

A word containing user-defined information to be passed to the LPH via the station table at offset ZQSSTS. The default is zero.

FUNCTION DESCRIPTION:
The directive must be included once for each line (i.e., a pair of channels on a multiline communications processor (MCP)) on which there are devices driven by the user-written LPH. An LDBU directive must be included so that the CLM will load the user-written LPH bound unit and execute its initialization code (see "Load Bound Unit Directive", earlier in this section). If the sizes of the channel and station tables are different than the default sizes for these tables, then an LPHDEF directive must be included before the LPHn directives (see "Line Protocol Handler Definition Directive," earlier in this section).

If there is more than one device on a line which is driven by the user-written LPH, then the additional devices on the line may be identified with STATION directives which immediately follow the LPHn directive instead of with additional LPHn commands (see "Station Directive," later in this section).

Example:

```
LPH0  27,8,X'FD80',,,FDX,0
STATION  28,1
```

In this example, line FD80 has two synchronous full duplex terminals controlled by a user-written (i.e., through the LPH directive) line protocol handler.

STATION DIRECTIVE

Directive Name: STATION
The STATION communications configuration directive is used to specify additional devices on lines controlled by your LPH's (Line Protocol Handlers) that drive multiple devices per line. One device on the line must be identified by an LPHn command; additional devices are identified in STATION commands, one per device, immediately following their corresponding LPHn commands. For additional (polled) VIPs on a line, use additional VIP directives with the same channel number rather than STATION directives.

FORMAT:

```
STATION lrn [,lph-specific-word]
```

PARAMETER DESCRIPTION:

lrn
The logical resource number associated with the device. lrn is an integer having a decimal value of 0 through 255. A program may use this number to identify the device when it requests an input/output operation to the device.

[lph-specific-word]
Specifies a word containing information you've defined which is to be passed to the LPH via the station table at offset ZQSSTS. The default is 0.

NOTE: The priority level, channel number, modem type, line speed, and line procedure (FDX/HDX) of devices described in STATION directives, are obtained from the LPHn directive which precedes the STATION directive (see "Line Protocol Handler Directive," earlier in this section).

TELEPRINTER DEVICE DIRECTIVE

Directive Name: TTY
The TTY communications configuration directive indentifies each teleprinter device.

FORMAT:

TTY lrn,level,X'channel' [,modem] [,speed]

PARAMETER DESCRIPTION:

lrn

The logical resource number associated with the device. lrn is an integer having a decimal value of 1 through 255. A program may use this number to identify the device when it requests an input/output operation to the device.

level

The priority level at which the communications supervisor processes requests for an input/output operation on the device. level is an integer from 6 to 60; it may be the same as the level specified for other communications devices, but it must be greater than the communication interrupt level specified in the COMM communications configuration directive (described earlier in this section). The level specified for one or more communications devices may not also be used for noncommunications devices or tasks.

X'channel'

Is a four-digit hexadecimal number (from X'0400' to X'FF80') specifying the channel number of the device and has the following format:
Bits 0 through 5—The six-bit number of the MCP (multiline communications processor) on the bus.
Bits 6 through 9—The four-bit even number of the receive channel of one of the eight lines of a MCP
Bits 10 through 15—Are set to zero.

[modem]

A number specifying the type of data set. Possible values are:
0—Direct connect.
1—Bell lxx-type modem (103A,113F,etc). Both data-set-ready and carrier-detect signals are needed for a connection; lack of both signals is a disconnection.
2—Bell 2xx-type modem (201A,201X,208A,etc). The data-set-ready signal is needed for a connection; lack of this signal is disconnection.
3 or greater—User-defined modem type (see MODEM communication configuration directive earlier in this section).

NOTE: The default value is modem type 1.

[speed]

The data rate in bits per second. For an asynchronous line with a line adapter whose id has a value of 2108_{16} (i.e., X'2108'), use the following values for speed:

	50	300	2400
	75	600	3600
(default)	110	900	4800
	134	1200	7200
	150	1800	9600

For an asynchronous line with a line adapter whose id has a value of either 2110_{16} (i.e., X'2110') or 2118_{16} (i.e., X'2118'), use the following values for speed:

	50	200	1800
	75	300	2000
(default)	110	600	2400
	134	1050	4800
	150	1200	9600

NOTE: If the date rate of the line is 134.5, specify 134.

Examples:
A DEVICE configuration directive must be used in conjunction with each TTY communications directive if the terminal is to be referenced as a file (e.g., in a higher level language).

Example 1:

```
TTY  21,8,X'FF80'
DEVICE  TBCH,21,8,X'FF80',TTY1
```

In this example, the TTY terminal is connected by a Bell lxx-type modem and operates at 110 bits per second. Default values specifying the modem and speed have been chosen.

Example 2:

```
TTY  22,8,X'FF00',2,1200
DEVICE TBCH,22,8,X'FF00',TTY2,119
```

In this example, the TTY terminal is connected by a Bell 2xx-type modem and operates at 1200 bits per second. Both values specifying the modem and speed have been specifically defined.

NOTE: Read/write data bytes (including a control byte) to the TTY terminal can have a length of up to 119 bytes instead of 72 bytes.

VIP DEVICE DIRECTIVE

Directive Name: VIP
The VIP communications configuration directive identifies each visual information projection (VIP) device.

FORMAT:

VIP lrn,level,X'channel'[,modem][,poll_address] $\left[,\left\{\begin{matrix}C\\T\end{matrix}\right\}\right]$

[,ROP_lrn][,ROP_type][,ROP_form_feed]

PARAMETER DESCRIPTION:
lrn

The logical resource number associated with the device. lrn is an integer having a value of 1 through 255. A program may use this number to identify the device when it requests an input/output operation to the device.

level

The priority level at which the communications supervisor processes requests for an input/output operation on the device. level is an integer from 6 to 60; it may be the same as the level specified for other communications devices, but it must be greater than the communication interrupt level specified in the COMM communications configuration directive (described earlier in this section). The level specified for one or more communications devices may not also be used for noncommunications devices or tasks.

X'channel'

Is a four-digit hexadecimal number (from X'0400' to X'FF80') specifying the channel number of the device and has the following format:
Bits 0 through 5—The six-bit number of the MCP (multiline communications processor) on the bus
Bits 6 through 9—The four-bit even number of the receive channel of one of the eight lines of a MCP
Bits 10 through 15—Are set to zero

[modem]

A number specifying the type of data set. Possible values are:
0—Direct connect.
1—Bell lxx-type when (103A,113F,etc). Both data-set-ready and carrier-detect signals are needed for a connection; lack of both signals is a disconnection.
2—Bell 2xx-type modem (201A,201X,208A,etc). The data-set-ready signal is needed for a connection; lack of this signal is disconnection.
3 or greater—User-defined modem type (see MODEM communication configuration directive earlier in this section). The default is coded type 2.

[poll_address]

Specifies the address of the VIP controller on the line specified by the X'channel' parameter. poll_address is an integer having a value of 0 through 31. If no polling address is specified, no polling occurs on the channel; in this case, only one VIP controller can be on the line.

[{C}]
[{T}]

C specifies normal VIP processing. T specifies File Transmission is to be used on this channel. The default for either C or T is a value of C.

[rop_lrn]

Specifies the logical resource number of a receive-only printer connected to the VIP controller. lrn is an integer having a decimal value of 0 through 255. The default is no receive-only printer is connected to the VIP controller; in this case, the rop_type and rop_form feed parameters (which follow) must be omitted.

[rop_type]

Specifies the following TTY or Terminet model numbers having a receive-only printer.
Terminets:
TN100
TN150
TN300
TN1200

TTYS:
TTY33
TTY35

The default assignment is a TN300.

[rop_form_feed]

Specifies whether or not the receive-only printer has a form feed option.

{FORM}
{FO }

Indicates receive-only printer will have a form-feed option.

{NOFORM}
{NO }

Indicates receive-only printer will not not a form-feed option.

The default is Terminets *have* a form-feed option and TTY's *do not* have a form-feed option.

NOTE: For VIP lines that have more than one terminal (i.e., polling is required): (1) a VIP directive is required for each terminal on the line, (2) the VIP directives must be grouped together and (3) the logical resource numbers (LRNs) for each terminal should be in sequential order.

In the following three examples, three VIP terminals are connected by Bell 2xx-type modems. Default values specifying the modems have been chosen. The DEVICE configuration directive *must be* used in conjunction with the VIP directive if the terminal is to be referenced as a file (e.g., in a higher level language). The first two examples show VIP terminal on the same line (i.e., channel FE80).

Example 1:

```
VIP  23,8,X'FE80',,0
DEVICE  VBCH00,23,8,X'FE80',VIP0
```

In this example, the VIP terminal has a poll address of 0.

Example 2:

```
VIP  24,8,X'FE80',,1,,25,TN1200,FORM
DEVICE  VBCH01,24,8,X'FE80',VIP1
DEVICE  RTNH,25,8,X'FE80',ROP1
```

In this example, the VIP terminal has a poll address of 1. In addition to a screen, the VIP terminal has a TermiNet 1200 (with form feed) and is attached as a receive-only printer. The VIP screen has a logical resource number 24 and the receive-only printer has a logical resource number 25.

Example 3:

```
VIP 26,6,X'FE00',,,T
```

In this example, line FE00 is used for file transmission with a Level 66 computer. As the Level 6 computer is a tributary station on the line, the Level 66 computer sends the first quiescent (Q) frame. The VIP line protocol handler does not poll the line. Since the line is used only by the file transmission program, and the file transmission program does not access the line as a file, no file is defined (i.e., through use of a DEVICE configuration directive) for the line.

NOTE: For VIP lines that have more than one terminal (i.e., polling is required): (1) a VIP directive is required for each terminal on the line, and (2) the VIP directives must be grouped together. It is also recommended that the logical resource numbers of VIP terminals on the same line be in order.

SECTION 3

OPERATOR CONTROL LANGUAGE

This section describes the elements of the operator control language (OCL), by which the system operator exercises control over the MDT operating system. For the purpose of this section the system operator is defined as any person who communicates with the operating system through the terminal device designated as logical resource number zero (LRN 0) at system configuration. The LRN 0 device is known as the operator terminal.

COMMANDS

OCL commands are read and interpreted by a system software component known as the OCL processor, which executes as the lead task in the system task group. Each command causes a task to be spawned within the system task group to perform the requested function (e.g., create an online task group, enter a group request, abort a group). When the execution of a command terminates, control is returned to the OCL processor, which is then capable of accepting another command.

Command Line Format

A command line to the OCL processor is a string of up to 127 ASCII characters whose general form is

$$arg_1\ [arg_2\ \ldots\ arg_n]$$

where arg_1 is the simple pathname of the bound unit that performs the command's function. It is the character string defined in the "Command Name" entry in each of the command descriptions in this section. Each subsequent arg entry is a parameter or control argument. The functions of parameters and control arguments are described in the following paragraphs.

Parameters

A parameter is an operator-supplied character string which supplies a value for some command variable, such as the pathname of a file on which action is to be taken by a command, the identification of a task or task group, or the pathname of an input file required by a command. Parameters are positional in a command line, that is, their positions in the line indicate which variables the values are being supplied to. Some commands require no parameters; others accept one or more as indicated in the syntax in the individual command descriptions. In a few commands, a parameter may be optional, as indicated by its being enclosed in brackets, thus: [path].

Control Arguments

A control argument is used to specify some option applicable to a command, such as the pathname of an alternate input or output file. Control arguments are differentiated from parameters by the presence of a keyword, which is a fixed-form character string preceded by a hyphen, thus: -ECL. A keyword may be followed by a user-supplied value, as -IN path.

Control arguments, when included in a command syntax, are collectively represented by the term ctl_arg, and the parameter descriptions define the individual keywords applicable to the command. Unless otherwise noted, keywords are optional and can appear in any order in the command line. If all keywords are optional in a given command, the ctl_arg term is enclosed in brackets in the syntax definition; if an argument is required, the requirement is noted in the individual keyword description and the ctl_arg term is not enclosed in brackets.

Spaces in Command Lines

Arguments in command lines are separated from each other by spaces. Unless otherwise indicated, wherever a space appears in a command line syntax, it represents one or more space characters, or one or more horizontal tab characters, or any combination of these. Spaces can be embedded within an

argument by enclosing the argument in apostrophes (') or double quote characters ("). If the enclosing character is also required *within* the argument, it is represented by two successive characters, thus: "NAME = " "SMITH" " AREA 203".

STANDARD OCL PROCESSOR FILES

Four files are always associated with the OCL processor. These are:

- o Command Input file (COMMAND_IN)
- o User Input file (USER_IN)
- o Operator Output file (OPERATOR_OUT)
- o Error Output file (ERROR_OUT)

The functions and characteristics of these files are described in the following paragraphs.

Command Input File

The command input file for the OCL processor is the file from which OCL command lines are read. More specifically, it is the device designated as the operator terminal (the LRN 0 device established at system configuration). It can at times, however, be assigned temporarily to another device or file as during the execution of the EC command. At the termination of execution of this command, the command input file reverts to the operator terminal.

User Input File

The user input file is the file from which a command function, during its execution, reads its own input. At the conclusion of OCL processor initialization, and as long as no alternate user input file is specified as a command parameter, the user input file remains the same as the command input file. At the termination of a command which names an alternate user input file, the user input file reverts to its initial assignment.

Operator Output File

The operator output file is the file to which a command function writes its output. At the conclusion of OCL processor initialization, and as long as no alternate operator output file has been specified, the operator output file is the device designated as the operator terminal at system configuration.

The operator output file can be directed to another device through the use of the FILE OUT command. It remains assigned to this device until another FILE OUT command is processed, at which time it can be directed to yet another device, or back to the operator terminal, at the system operator's discretion.

Error Output File

The error output file is the file to which the OCL processor and any commands invoked by it writes information related to error conditions detected by it. The error output file is always the operator terminal; it cannot be reassigned by any OCL command or command parameter.

OCL COMMAND FORMATS AND DESCRIPTIONS

The remainder of this section comprises complete descriptions of the formats, parameters, control arguments, and functions of the operator control language commands. In cases in which the command formats are non-trivial, one or more illustrative examples are also given.

The command descriptions given on the following pages are arranged in alphabetic order to facilitate references to specific commands. A summary list of the commands, grouped by functional categories, is given below.

Task Group Creation and Deletion Commands

Function Name	*Command Name*
Abort Batch Task Group	ABORT_BATCH
Abort Online Task Group	ABORT_GROUP

Function Name	*Command Name*
Create Batch Task Group	CB
Create Online Task Group	CG
Delete Batch Task Group	DB
Delete Online Task Group	DG
Spawn Online Task Group	SG

Task Group Execution Commands

Function Name	*Command Name*
Abort Batch Request	ABR
Abort Group Request	AGR
Activate Batch Task Group	ACTB
Activate Online Task Group	ACTG
Enter Batch Request	EBR
Enter Group Request	EGR
Execution Command	EC
Suspend Batch Task Group	SSPB
Suspend Online Task Group	SSPG

File and Directory Control Commands

Function Name	*Command Name*
Change System Directory	CSD
Change Working Directory	CWD
File Out	FO
List Search Rules	LSR
List Working Directory	LWD
Modify File	MF
Reassign	RAS

System and Status Commands

Function Name	*Command Name*
Modify External Switches	MSW
Ready Off	RDF
Ready On	RDN
Set Date	SD
Status Group	STG
Status System	STS

ABORT_BATCH

Command Name: ABORT_BATCH
Suspend the batch task group and terminate it.

Format:

ABORT_BATCH

Parameter Description:
No parameters are required or permitted with this command.

Function Description:
The ABORT BATCH command causes the suspension and termination of the batch task group, whether it is active or dormant. It removes all of the data structures which define and control the execution of the task group, and returns all memory used by the group to the batch memory pool. Any files that may have been open during the execution of the task group are closed. Any requests pending against the batch task group are cancelled.
The action of the ABORT BATCH command is similar to the DELETE BATCH command, the difference being that the latter must wait until the task group becomes dormant, while the former takes effect as soon as all outstanding input or output orders are complete.

ABORT BATCH REQUEST

Command Name: ABR
Terminate the execution of the current batch request.

Format:

ABR

Parameter Description:
No parameters are required or permitted with this command.

Function Description:
The ABORT BATCH REQUEST command causes the cessation of execution of the current request in the batch task group. It removes all defining and controlling data structures except those associated with the execution control language (ECL) processor, and returns all associated memory to the batch memory pool. Any files that are open and in use by the batch task group are closed. At the conclusion of execution of the ABR command, the ECL processor honors the next request in the batch request queue, if any.

ABORT GROUP

Command Name: ABORT GROUP
Suspend the indicated online task group and terminate it.

Format:

ABORT_GROUP id

Parameter Description:
id

The group identification of a task group previously created by a CG or SG command specifying the same id.

Function Description:
The ABORT GROUP command causes the suspension and termination of an existing online task group whether it is active or dormant. It removes all of the data structures which define and control the execution of the task group, and returns all memory used by the group to the appropriate memory pool. Any files that may have been open during the execution of the task group are closed. Any requests that may be pending against the group are cancelled. The action of the ABORT GROUP command is similar to the DELETE GROUP command, the difference being that the latter must wait until the task group becomes dormant, while the former takes effect as soon as all outstanding input or output orders are complete.

Example:

ABORT_GROUP AX

A task group identified as AX is terminated.

ABORT GROUP REQUEST

Command Name: AGR
Terminate the execution of the current request in the indicated task group.

Format:

AGR id

Parameter Description:
id

The group identification of a task group previously created by a CG or SG command specifying the same id.

Function Description:
The ABORT GROUP REQUEST command causes the cessation of execution of the current request in the indicated task group. It removes all defining and controlling data structures except those associated with the lead task (as defined by the CG or SG command specifying this id) and returns associated memory to the appropriate memory pool. Any files that are open and in use by this task group are closed. At the conclusion of execution of the AGR command, the lead task processes the next request against this group, if any.

Example:

AGR AX

Execution of a request against a task group identified as AX is terminated. Upon termination of this request, the next request in this task group's request queue is executed.

ACTIVATE BATCH

Command Name: ACTB
Resume execution of the batch task group, which was previously suspended.

Format:

ACTB

Parameter Description:
No parameters are required or permitted with this command.

Function Description:
The ACTIVATE BATCH command causes the resumption of execution of any tasks that were active at the time a SUSPEND BATCH command was issued. All tasks which had been active at the time of suspension are requeued on their respective level queues.
Resumption of execution of the task group presupposes that the group has not been rolled out during the time it was suspended; if it was, execution resumes when the group is rolled in again; i.e., when one or more online task groups return memory to the batch memory pool.

ACTIVATE GROUP

Command Name: ACTG
Resume execution of a previously suspended online task group.

Format:

ACTG id

Parameter Description:
id

The name of a task group previously suspended which is to be reactivated.

Function Description:
The ACTIVATE GROUP command causes the resumption of execution of any tasks that were active at the time a SUSPEND GROUP command with the same group id was issued. All tasks that had been active at the time of suspension are requeued on their respective level queues.

Example:

ACTG AX

The task group identified as AX, previously suspended, is to be returned to the active state.

CHANGE SYSTEM DIRECTORY

Command Name: CSD
Define a new pathname for one of the system directories.

Format:

CSD [path] [ctl_arg]

Parameter Description:
[path]

The pathname of the new system directory. If this parameter is omitted, the pathname >SYSLIB1 is assumed.

[ctl_arg]

Only one control argument is recognized, and is described below.

-LIBx

The name of the system directory that is to be changed to the new pathname. Possible values are LIB1 and LIB2. If not speciifed, the default is LIB1.

Function Description:
The CHANGE SYSTEM DIRECTORY command gives the system operator the ability to change the pathname of one of the two directories that the system uses in its search for bound units. The pathname given can be either a simple name, a relative pathname, or a full pathname. If it is a simple name or relative pathname, elements of the system task group's working directory are used to construct a full pathname. The working directory is system_volume_name, unless it has been modified by a CHANGE WORKING DIRECTORY command. Both system directory pathnames can be changed by using two CSD commands.
The system uses a set of rules, known as search rules, to govern its search of directories for a given path. These rules are described in detail in the discussion of the LIST SEARCH RULES command.

Example:

CSD NEW_DIR -LIB2

Assuming that the system task group's working directory has not been modified, the system constructs the pathname >NEW_DIR, and uses this pathname whenever LIB2 is referred to.

CHANGE WORKING DIRECTORY

Command Name: CWD
Change the system task group's default working directory to the specified path.

Format:

CWD [path]

Parameter Description:
[path]

The pathname of the new working directory. It may be a relative name or a full pathname. If this parameter is omitted, the pathname established as the system task group's initial working directory is assumed.

Function Description:
The CHANGE WORKING DIRECTORY allows the system operator to modify the pathname of the system task group's default working directory. At the conclusion of OCL processor startup, the working directory is system_volume_name. However, there can exist directories subordinate to this directory, and these subdirectories can contain files used by the system task group. If these files are to be referred to by simple pathnames then it is necessary to change the directory point of reference to the directory which immediately contains these files.
Assume for example that there is a series of functions which the system operator routinely performs at the beginning of a day's operations, after system initialization is complete. These functions could be cataloged in a file contained in a directory subordinate to the working directory, and used as input to the execution command processor (see the EC command).
After issuing a CWD command naming the subdirectory as its path parameter, the EC command could be given specifying the simple name of the file containing the functions to be performed.
If the CWD command is issued without a path parameter, the directory reference point reverts to the path name which was in effect at the conclusion of OCL processor startup.

Example:
A file containing a series of OCL CREATE GROUP commands exists in a directory EC_ROUTINES subordinate to the working directory. The name of the file is CR_GRPS, and it is used each day to create a predetermined set of task groups for the day's operations. The system operator issues a command,

CWD EC_ROUTINES

to move the directory point of reference to the EC_ROUTINES directory level. He then issues a command,

EC CR_GRPS

to initiate the execution of the set of CG commands.

CREATE BATCH

Command Name: CB
Perform the initialization functions necessary to the initiation of the batch task group.

Format:

CB phys_lvl [ctl_arg]

Parameter Description:
phys_lvl
The base priority level relative to which all tasks in the batch task group will execute.

[ctl_arg]
One or more control arguments chosen from the following list.

-LRN n
Specifies the highest logical resource number (LRN) which will be referred to by any task in the batch task group. The minimum value that can be specified for n is the highest LRN used by the system task group; this is also the default if this argument is not specified.

-LFN n
Specifies the highest logical file number used by any task in the batch task group. If -LFN is not specified, n assumes the value 15. Refer to the ASSOCIATE PATH command, Section 4.

Function Description:
The CREATE BATCH command causes the allocation and initialization of all data structures used by the system to define and control the execution of the batch task group. It causes the loading of the execution control language (ECL) processor, and defines it as the lead task of the task group. It does not cause the activation of the ECL processor; this is done by use of the ENTER BATCH REQUEST command.

Example:

CB 12 -LFN 6

The batch task group control data structures are created and initialized. No task in the group is expected to execute at a priority level lower than 12, nor refer to a logical file number greater than 6.

CREATE GROUP

Command Name: CG
Perform the initialization functions necessary to the initiation of an online task group.

FORMAT:

CG id phys_lvl ctl_arg

PARAMETER DESCRIPTION:
id
The group identification of the new task group. It is a two-character name that cannot have the $ as its first character.

phys_lvl
The base priority level relative to which all tasks in this task group will execute.

ctl_arg

One or more control arguments chosen from the following list. The -POOL argument is required.

{-EFN root
-EFN root?entry}

The root segment of a bound unit root is to be loaded as the lead task if it is not already loaded.

The root segment name can be suffixed with ?entry, where entry is a symbolic start address within the root segment. If not given, the start address established when the bound unit was linked is assumed.

-ECL

The root segment of the execution control language (ECL) processor is to be loaded as the lead task.

-LRN n

Specifies the highest logical resource number (LRN) that will be referred to by any task in the task group. The minimum value that can be specified for n is the highest LRN used by the system task group; this is also the default if this argument is not specified.

-LFN n

Specifies the highest LFN used by any task in the task group. If -LFN is not specified, n assumes the value 15. Refer to the ASSOCIATE PATH command, Section 4.

-POOL id

id is a two-character ASCII identifier and is the name of the memory pool from which all memory required by this task group is to be taken. This argument is required, and must name a pool defined at system initialization by a CLM MEMPOOL directive.

NOTE: In any invocation of the CG command, -EFN *or* -ECL, but not both, can be specified. If neither is specified, -ECL is assumed.

FUNCTION DESCRIPTION:
The CREATE GROUP command causes the initialization and allocation of all data structures used by the system to define and control the execution of a task group. It causes the loading of the root segment of the lead task of the task group. It does not cause the system to activate any task within the task group.

Example:

CG AX 15 -EFN MAIN_PG?ENTRY1 -LRN 18 -POOL A2

A task group identified as AX is created. The lead task of the group is the program MAIN_PG, whose execution is to be started at the symbolic address ENTRY1. No task in the group will execute at a priority level lower than 15, nor refer to a logical resource number higher than 18. Memory will be obtained from the pool identified as A2 at system configuration.

DELETE BATCH

Command Name: DB
Delete the batch task group definition previously created by the CREATE BATCH command.

FORMAT:

DB

PARAMETER DESCRIPTION:
No parameters are required or permitted with this command.

FUNCTION DESCRPTION:
The DELETE BATCH command removes all of the data structures that were constructed by the CB command issued previously. No more ENTER BATCH REQUEST commands can be issued for the batch task group after the DB command has been executed. The DB command takes effect immediately if the task group is dormant when the command is issued. If it is active (i.e., if its code is being executed and/or there are still requests in the task group's request queue), the DB command takes effect when execution terminates and there are no more requests in the queue.
When the batch task group is deleted, the memory occupied by the data structures defining the group and any memory associated with the execution of the group is returned to the batch memory pool.

DELETE GROUP

Command Name: DG
Delete an online task group definition previously created by a CREATE GROUP command.

FORMAT:

DG id

PARAMETER DESCRIPTION:
id

The group identification of a task group previously created by a CG command specifying the same id.

FUNCTION DESCRIPTION:
The DELETE GROUP command removes all of the data structures that were constructed by the CG command issued previously with this id. No more ENTER GROUP REQUEST commands can be issued for this task group after the DG command has been executed. The DG command takes effect immediately if the task group is dormant when the command is issued. If it is active (i.e., if its code is being executed and/or there are still requests in this task group's request queue), the DG command takes effect when execution terminates and there are no more requests in the queue.
When a task group is deleted, the memory occupied by the data structures defining the group and any memory associated with the execution of the group is returned to the appropriate memory pool.

ENTER BATCH REQUEST

Command Name: EBR
Enter a request for execution of the execution command language (ECL) processor in the batch request queue.

FORMAT:

EBR user_id in_path [ctl_arg]

PARAMETER DESCRIPTION:
user_id

A field comprising two subfields in the form person.project, by which this request is identified. The user_id subfields are also used to establish the working directory for this request.

in_patch
The name of the file from which the ECL processor is to read its commands.

[ctl_arg]
One or more control arguments chosen from the following list.

-OUT out_path
Defines the pathname of the file which is to receive output from the batch task group. If not specified, one of the following assumptions is made:
If in_path specifies a mass storage file, out_path = in_path.AO
If in_path specifies an interactive terminal, out_path = in_path
If in_path specifies an input-only device, out_path is null.

-WD path
Specifies that path is to be used as the working directory pathname instead of the pathname established by the user id parameter.

FUNCTION DESCRIPTION:
The ENTER BATCH REQUEST command initiates the execution of the ECL processor as the lead task in the batch task group previously created by the CREATE BATCH command. If the task group is dormant at the time the EBR command is issued, execution begins immediately; otherwise, the request is queued for execution when the group becomes dormant (i.e., a previous EBR command has activated the task group and the ECL processor or a command invoked by it is still executing). The ECL processor obtains its commands from the file named in the in_path parameter. This means that the file must begin with an ECL command, although it may contain other items which the called ECL function may require for its execution (e.g., Editor directives).

Example:

```
EBR BROWN.LIBRARY CMMD_IN
```

The batch task group is to be activated by a request identified as BROWN.LIBRARY. It will receive its input from and direct its output to files identified as CMMD_IN and CMMD_IN.AO, respectively. The working directory pathname for this request is >UDD>LIBRARY>BROWN.

ENTER GROUP REQUEST

Command Name: EGR
Activate the lead task of an online task group previously created by a CREATE GROUP command.

FORMAT:

```
EGR id user_id [in_path] [ctl_arg]
```

PARAMETER DESCRIPTION:
id
The group identification of a task group previously created by a CG command specifying the same id.

user_id
A field comprising two subfields in the form person.project, by which this request is identified. The user_id subfields are also used to establish the working directory for this request.

[in_path]
The name of the file from which the ECL commands or user input are to be read by the task group during execution. This argument is set to null if it is not specified. It is required if the CG command specified the control argument -ECL.

[ctl-arg]

One or more control arguments chosen from the following list.

-OUT out_path

Defines the pathname of the file that is to receive user output from the task group. If not specified, one of the following assumptions is made:

If in_path specifies a mass storage file, out_path = in_path.AO
If in_path specifies an interactive terminal, out_path = in_path
If in_path is not specified, out_path is null
If in_path specifies an input-only device, out_path is null.

-ARG

Indicates that additional arguments required by the task group during execution follow. These additional arguments are passed to the lead task to be used as necessary. If used, the -ARG control argument must appear last.

-WD path

Specifies that path is to be used as the working directory pathname instead of the pathname established by the user id parameter.

FUNCTION DESCRIPTION:

The ENTER GROUP REQUEST command initiates the execution of the lead task of a task group previously created by a CREATE GROUP command. If the task group is dormant at the time the EGR command is issued, execution begins immediately; otherwise, the request is queued for execution when the group becomes dormant (i.e., a previous EGR command has activated this task group and it has not yet terminated). Execution begins at the point specified by the -EFN control argument of the CG command, if specified. If the -EFN control argument was not used (i.e., the lead task is the ECL processor), execution begins by reading the file named by the in path parameter. This means that this file must begin with an ECL command, although it may contain other items which the called ECL function may require for its execution.

Example:

```
EGR AX SMITH.SERVICES MPG_DATA -ARG '07/12/76 1100AM'
```

The task group identified as AX in a previous CG command is to be activated. This request is identified as SMITH.SERVICES. The task group expects its input data to come from a file named MPG_DATA, in the working directory >UDD>SERVICES>SMITH, and will write its output to a file named MPG_DATA.AO, also in the working directory. The lead task expects one argument, a date and time item. The item is enclosed in apostrophes because there is an embedded space, but it is to be interpreted as a single argument.

EXECUTION COMMAND

Command Name: EC

Invoke the execution command (EC) processor to read OCL commands from a designated file.

FORMAT:

```
EC path
```

PARAMETER DESCRIPTION:

path

The name of a file, path.EC, containing OCL commands and EC directives.

FUNCTION DESCRIPTION:

The function of the EC processor is to read from a previously created file a series of OCL commands and EC directives. It provides a mechanism whereby a sequence of routinely performed functions can be executed without the need for manually entering the commands through the operator terminal.

The file path.EC is a file which has been previously created by use of the Editor. It contains one or more OCL commnds and EC directives which are interpreted in sequence by the EC processor and acted upon as described in the following paragraphs.
When a command is encountered by the EC processor, it is simply passed to the OCL processor for interpretation and execution. This means that the syntax of the command as read from the file path.EC must be identical to that which would have been entered from a terminal device if the function were requested manually. All parameters and control arguments must be supplied as specified in the individual OCL command descriptions.
When a command execution terminates, control is returned to the EC processor, which then reads the next line from the file.
The EC processor also recognizes several directives which are not passed to the command processor, but are interpreted and acted upon by the EC processor itself. These directive lines are identified by a character string beginning with & and followed by a Δ (space or tab character). They provide control over certain operational aspects of the EC processor as well as a degree of control over the logic of execution of the series of commands. Any & directive other than those described below is treated as an &QΔ directive, except that an error status code is returned to the task that invoked the EC command. This code comprises the last four hexadecimal digits of the error code as defined in Section 5.
The EC control directives are described in detail in the following paragraphs.

&Δ

This signifies a comment line and is not processed further. It is visible only by obtaining a listing of the EC file, and can be used, for example, to describe the function performed by the commands contained in the file.

&FΔ

Command line printing is to be turned off; i.e., command lines are not to be written to the user out file. This is the default; command lines are not normally written to user_out.

&NΔ

Command line printing is to be turned on. Each command line read from the EC file is written to the user_out file before being passed to the command processor. The & directive lines, except for &PΔ lines, are not written.

&PΔ

The entire line, except for the &PΔ, is written to the user_out file. Printing of &PΔ lines occurs regardless of whether command line printing is on or off.

&IFΔ

This directive permits the interrogation by the EC processor of an error status code returned by the command executed immediately prior to the &IFΔ directive. For compatibility across all GCOS systems, it has the format:

&IFΔ[EQUALSΔ&STATUSΔ0]Δ&THENΔ&ELSEΔ&QUIT

and is interpreted as follows:
If the error status code returned from the execution of the immediately preceding command line is zero, continue with the next command or directive line; otherwise quit.

The &IFΔ directive enables the EC processor to exit from any further processing of the EC file when an error condition resulting from the interpretation or execution of a command would make meaningless the execution of any subsequent commands.

&QΔ

The execution of the current EC file is terminated, and control is returned to the invoking task. Implicit &QΔ directives may be executed as described above, by invalid & directives or because of error status codes returned by the interpretation or execution of a command line. To ensure proper termination of the EC command, every EC file should have the &QΔ directive as its last line.

Example:
At the beginning of each day's operations, the system operator is required to create and initiate three task groups. One is the batch task group used for program development; the other two are online task groups, one using the ECL processor and the other a daily-run production program. The operator has created an EC file in the system task group's initial working directory, system_volume_name. The name of the file is CR_GRPS.EC and the following OCL commands are contained in it:

```
CB 30 -LRN 10 -LFN 8
CG EC 25 -LRN 12 -POOL A1
CG PR 20 -LRN 15 -LFN 6 -EFN PROD_A -POOL P1
EGR EC GRP.PRIME >SPD>VIP01
EBR DEV.PROG >SPD>KSR02 -OUT >SPD>LPT01
EGR PR ADMIN.PROJ >SPD>CDR01 -OUT >SPD>LPT02
&PΔGROUPS CREATED AND ACTIVATED
&QΔ
```

The batch task group, created by the CB command, runs at base priority level 30 with the ECL processor as its lead task. Its user_id, established by the EBR command, is DEV.PROG, from which is derived its default working directory >UDD>PROG>DEV. Its input (commands and user input) comes from an interactive terminal KSR02, and its error and user output is directed to line printer LPT01.
The first CG command creates an online task group EC, which, because no EFN is specified, uses the ECL processor as its lead task. This task group can be used to create other groups in response to the day-to-day needs of the installation, using whatever ECL commands are required. Task group EC operates at base priority level 25 and utilizes memory from the memory pool defined as A1 at system configuration. Its input and output are through the interactive terminal VIP01.
The second CG command creates an online task group PR, which uses a bound unit named PROD_A as its lead task. It operates at the highest priority level of the three tasks (20) and obtains its memory from the memory pool P1. It receives its user input from a card reader CDR01 and writes its user and error output to a second line printer LPT02.
At the conclusion of processing of the EC file a message is directed to the system operator stating that the task groups have been created and activated. If any errors were encountered in the interpretation or execution of the OCL commands, an appropriate error message would be displayed to the system operator (refer to Section 5, Error Messages), and the processing of the EC file continues.
The sequence of commands in the above example are intended to show that, once a set of task groups has been created, the order of activation (by the ENTER commands) is immaterial. While group EC may begin its execution first by virtue of its request having been processed first, as soon as the request for group PR is processed, group PR takes priority over group EC (because its priority level is higher).

FILE OUT

Command Name: FO
Change the destination to which system messages to the operator are sent.

FORMAT:

FO [path]

PARAMETER DESCRIPTION:
[path]
The pathname of the new destination for operator output. It can represent any file or device capable of being used for output. If the parameter is omitted, the operator output file reverts to that established at the conclusion of OCL processor startup.

FUNCTION DESCRIPTION:
The FILE OUT command defines a new file or device pathname to which output generated by the system task group will be written. When the OCL processor is initially activated, the system output file pathname is >SPD>CONSOLE. Error output is also written to the same file.
The FO command makes it possible for the operator to redirect the message output (but not the error output) to a different file or device for reasons, say, of high output message activity, in which case a faster output device such as a line printer might be desirable.
The use of the FO command without the path parameter resets the destination of the operator output to the file/device established when the OCL processor was initially activated.

Example:

FO >SPD>LPT01

The output generated by the system task group is directed to the line printer LPT01.

LIST SEARCH RULES

Command Name: LSR
Display the search rules currently defined for the system task group.

FORMAT:

LSR

PARAMETER DESCRIPTION:
No parameters are required or permitted with this command.

FUNCTION DESCRIPTION:
The LIST SEARCH RULES command writes to the operator output file the full pathnames of the directories used by the system task group in its search for bound units.
The search rules define three directory pathnames and the sequence in which they are used during a search. The first of these is the system task group's working directory; its pathname is ^system_volume_name at the completion of OCL processor startup, and remains so until modified by one or more CHANGE WORKING DIRECTORY commands. The second is the system directory LIB1, whose pathname is >SYSLIB1. The third is the system directory LIB2, whose pathname is also >SYSLIB1. The pathnames associated with LIB1 and LIB2 can be changed through the use of the CHANGE SYSTEM DIRECTORY command. The pathnames returned by the LSR command always reflect the current directory pathnames.

Example:
Assume that the system task group's initial working directory is ^SYSVOL, and that no CWD or CSD commands have been issued. The LSR command returns

^SYSVOL

^SYSVOL>SYSLIB1

^SYSVOL>SYSLIB1

Assume now that a CSD NEW DIR -LIB2 command has been executed at some point prior to the issuing of the LSR command. The LSR command now returns

^ SYSVOL

^ SYSVOL>SYSLIB1

^SYSVOL>NEW_DIR

LIST WORKING DIRECTORY

Command Name: LWD
List the full pathname of the working directory of the system task group.

FORMAT:

LWD

PARAMETER DESCRIPTION:
No parameters are required or permitted with this command.

FUNCTION DESCRIPTION:
The LIST WORKING DIRECTORY command can be used to write to the operator output file the full pathname of the working directory currently being used by the system task group. It is at times useful to be able to establish the identity of the working directory after having made several changes of working directories through the use of CHANGE WORKING DIRECTORY commands. The LWD command causes the full pathname of the working directory to be written to the operator output file in the form

^ system_volume_name[>dir1]...

The ellipsis indicates that one or more subordinate levels may be included in the path name of the current working directory, depending on the nature of previously issued CWD commands.

Example:
Assume that the system task group's initial working directory pathname was ^VOL_01 as established at OCL processor startup, and that a CWD EC_DIR command has been issued since that time. The LWD command returns

^ VOL_01>EC_DIR

If, starting with this working directory, a CWD < command is issued, a subsequent LWD command would return

^VOL_01

MODIFY EXTERNAL SWITCHES

Command Name: MSW
Modify selected external switches associated with the indicated task group.

FORMAT:

MSW id ctl_arg

PARAMETER DESCRIPTION:
id

The group identifier of the task group whose switches are to be modified.

ctl_arg

One or more control arguments chosen from the following list:

-ON $S_i[S_i]$...

Set the external switch indicated by S_i ON. Each S_i is a hexadecimal digit from 0 through F.

-OFF $S_i[S_i]$...

Set the external switch indicated by S_i OFF. Each S_i is a hexadecimal digit from 0 through F.

-ALL v

Set all switches to the value v. The value v can be either ON or OFF.

FUNCTION DESCRIPTION:
The MODIFY EXTERNAL SWITCHES command enables the system operator to modify the external switches by which a user task group can control its execution. An external switch can be thought of as a hardware switch on a control panel, which can be set on or off manually by an operator. There is a separate switch word associated with each task group created, giving each group the capability of addressing 16 switches. A user program can contain instructions or statements which interrogate the settings of one or more of these switches, and can use these settings to control the execution logic of the program.

Example:

```
MSW AX -ON 25 -OFF 7B
```

In the task group identified as AX, external switch numbers 2 and 5 are to be set ON, and external switch numbers 7 and B are to be set OFF.

MODIFY FILE

Command Name: MF
Modify the attributes of the specified file.

FORMAT:

```
MF path ctl_arg
```

PARAMETER DESCRIPTION:
path

The pathname file whose attributes are to be changed.

ctl_arg

One or more control arguments chosen from the following list:

{-SHARE / -SHR}

Specifies that the named file is to be made accessible to the batch task group.

{-NONSHARE / -NS}

Specifies that the named file is to be made inaccessible to the batch task group.

{-READ / -RD}
Specifies that no users are given permission to write to the named file; only reading is permitted.

{-WRITE / -WR}
Specifies that users are permitted access to the named file in the output, update, or extend mode.

NOTE: The arguments within the argument pairs -SHARE and -NONSHARE, and -READ and -WRITE, are mutually exclusive.

FUNCTION DESCRIPTION:
The MODIFY FILE command allows the accessibility and permission attributes of a file to be modified. When a file is first created (refer to the CREATE FILE command, Section 4), it is accessible to both online and batch task groups. It can also be read from and written to by any task. Its initial attributes are thus SHARE/WRITE.
If a file is made inaccessible to the batch task group (through the use of the -NS control argument), no access of any kind by the batch task group is permitted. Furthermore, directories can be given the -NS attributes; in this case the directory and all subdirectories and files contained within it are inaccessible to the batch task group.
Another kind of protection can be given a file by the use of the -RD control argument. This argument makes the file a read-only file, preventing any task groups, online or batch, from writing to the file. It can still be read by online tasks and, unless the -NS argument has also been specified, by batch tasks as well. Attributes assigned to nondisk files by this command remain in effect only for the current initialization of the system. If the system is reinitialized, attributes for these files revert to SHARE/WRITE.

Example:

```
MF >UDD>PROJ1>USERA>FILE01 -NS
```

A file is to be made inaccessible to the batch task group. It remains accessible for writing by online tasks.

READY_OFF

Command Name: RDF
Suppress the ready message printed at the completion of each OCL command.

FORMAT:

```
RDF
```

PARAMETER DESCRIPTION:
No parameters are required or permitted with this command.

FUNCTION DESCRIPTION:
The READY_OFF command suppresses the printing of a message issued by the system at the completion of execution of each OCL command. The message informs the operator that the system is prepared to accept another command.
If the RDF command is issued from within an EC file, when execution of the EC file is completed the system reverts to the ON/OFF state which was in effect when the EC command was invoked.
The initial state of the ready function at the conclusion of OCL processor initialization is OFF.

READY_ON

Command Name: RDN
Activate the printing of the ready message at the completion of each OCL command.

FORMAT:

RDN

PARAMETER DESCRIPTION:
No parameters are required or permitted with this command.

FUNCTION DESCRIPTION:
The READY_ON command activates the printing of a message issued by the system at the completion of execution of each OCL command. The message informs the operator that the OCL processor is prepared to accept another command.
If the RDN command is issued from within an EC file, when execution of the EC file is completed the system reverts to the ON/OFF state which was in effect when the EC command was invoked.
The initial state of the ready function at the conclusion of OCL processor initialization is OFF.

REASSIGN

Command Name: RAS
Exchange one device for another of the same type, or cancel a mount request for a device or volume.

FORMAT:

RAS ctl_arg

PARAMETER DESCRIPTION:
ctl_arg
One control argument chosen from the following list.

-SWAP dev_name$_1$ dev_name$_2$
The device controlled by the driver associated with dev_name$_2$ is to be placed under the control of the driver associated with dev_name$_1$. Both devices must be of the same type. dev_name$_1$ is set to the disabled state.

-CANCEL name
This control argument is used when a device or volume named in a File Manager mount message is unavailable. It instructs the File Manager to continue processing along the "not found" path. The form of the name argument is either ^ vol_id or dev_name, depending on the mount message.

FUNCTION DESCRIPTION:
The REASSIGN command enables the system operator to substitute one device for another of the same type. In case of a disk device malfunction, for example, the device can be replaced by another through the use of the -SWAP control argument. This argument gives the symbolic device names of the replaced and replacing devices as dev_name$_1$ and dev_name$_2$, respectively. The device names are those assigned to the devices at system configuration.
If a task issues a request for a file or volume to the File Manager, and that file or volume is not mounted, the File Manager issues a mount message to the system operator. If the operator determines that the requested volume, or the device upon which it is to be mounted, is unavailable, he can respond to the message with an RAS command specifying the -CANCEL control argument, causing the File Manager to respond to the task with a "not found" status code.

Example 1:

RAS -SWAP DSK03 DSK05

The disk device DSK03 is to be placed in the disabled state and replaced by the disk device DSK05.

Example 2:

RAS -CANCEL ^ USER04

A task has requested the mounting of a volume USER04, and the File Manager has issued a message to the operator directing him to mount the volume. The operator has determined that the volume is not available. He issues the RAS command to inform the File Manager that the volume is unavailable, and that it is to return a "volume not found" status code to the task.

SET DATE

Command Name: SD
Set the system internal clock to the indicated date and time.

FORMAT:

SD 'yyyy/mm/ddΔhh[mm[:ss]]'

PARAMETER DESCRIPTION:
'yyyy/mm/ddΔhh[mm[:ss]]'

The date and time to which the clock is to be set. yyyy is the year, mm is the month and dd is the day, in decimal. hh is the hour of day, and the optional mm and :ss specify minutes and seconds, respectively. The Δ represents exactly one space.

FUNCTION DESCRIPTION:
The SD command permits the system operator to initialize the system's internal clock to a specified date and time of day. The date and time, expressed as an ASCII character string, are converted to an internal form representing the number of milliseconds elapsed since January 1, 1901. The use of this command enables the system to respond appropriately to any of the several executive system service calls related to task control based on the passage of time.
The date/time value specified must be enclosed in apostrophes because of the embedded space between the dd and hh portions.

SPAWN GROUP

Command Name: SG
Create, request the execution of, and then delete a task group.

FORMAT:

SG id user_id phys_lvl [in_path] ctl_arg

PARAMETER DESCRIPTION:
id

The group identification of the task group to be spawned. It is a two-character name that cannot have the $ as its first character.

user_id

A field comprising two subfields in the form person.project, by which the requested execution of this task group is identified. The user_id subfields are also used to establish the working directory for this request.

phys_lvl

The priority level relative to which all tasks within this task group will execute.

[in_path]

The name of the file from which commands and user input are to be read by the task group during its execution. The file name is set to null if the in_path parameter is not specified. in_path must be specified if the control argument -ECL (see below) is used or implied.

ctl_arg

One or more control arguments chosen from the following list. The -POOL argument is required.

-OUT out_path

Defines the pathname of the file which is to receive user output from the task group. If not specified, one of the following assumptions is made:

If in_path specifies a mass storage file, out_path = in_path.AO
If in_path specifies an interactive terminal, out_path = in_path.
If in_path is not specified, out_path is null
If in_path specifies an input-only device, out_path is null.

-WD path

Specifies that path is to be used as the working directory pathname instead of the pathname established by the user id parameter.

{-EFN root / -EFN root?entry}

The name of a bound unit root segment to be loaded as the lead task (if not already loaded). The root segment name can be suffixed with ?entry, where entry is a symbolic start address within the root segment. If no suffix is given, the default start address of the root segment is assumed.

-ECL

Specifies that the lead task of the spawned task group will be the execution control language (ECL) processor.

-LRN n

Specifies the highest LRN that will be referred to by any task group. The minimum value that can be specified for n is the highest LRN used by the system task group; this is also the default if this argument is not specified.

-LFN n

Specifies the highest LFN used by any task in the spawned task group. If -LFN is not specified, n assumes the value 15.

-POOL id

id is a two-character ASCII identifier and is the name of the memory pool from which all memory required by the spawned task group is to be taken. This argument is required, and must name a pool defined by a CLM MEMPOOL directive.

-ARG

Indicates that additional arguments required by the spawned task group for its execution follow. These additional arguments are passed to the lead task of the spawned group to be used as necessary. If used, the -ARG control argument must appear last.

NOTE: In any invocation of the SG command, -EFN *or* -ECL, but not both, can be specified. If neither is specified, -ECL is assumed and the in_path parameter is required.

FUNCTION DESCRIPTION:
The SPAWN GROUP command combines the functionality of the CREATE GROUP, ENTER GROUP REQUEST, and DELETE GROUP commands. It implicitly causes the execution of these three functions in sequence; i.e., allocates and creates the data structures required to define and control the execution of the task group, places a request against the group, thereby activating it, and, when execution terminates, removes all controlling data structures and returns memory used by the task group to the appropriate memory pool.

The operator can use the SG command to create and activate a task group whose life span is dependent upon the actions of the task group itself, rather than upon any explicit action on the part of the operator. A task group using the ECL processor as its lead task is an example of such a group; it is initiated by the SG command, and it is terminated by an ECL BYE command entered by the user on whose behalf the task group is running. The duration of any task group may vary widely; the significant difference between a spawned task group and a created task group is that, in the former case, the operator need have no knowledge of when its execution terminates in order to delete the group and return its resources to the system. He does, however, have the ability at any time to determine the status of the group in question, through the use of the STATUS GROUP command.

Example:
An installation runs an application which is to begin at 10:00 A.M. each day, and lasts for an indeterminate period. The application updates a master file, accepting transaction input from a remote terminal device. It writes its output to a mass storage file for later conversion to hard copy. At the specified time the system operator enters the command

```
SG UM SMITH.REMOTE 10 >SPD>VIP06 -EFN UPMAST -OUT MSTRLST -POOL M1
```

The data structures defining and controlling task group UM are allocated and initialized. A request identified as SMITH.REMOTE is entered against the task group, establishing the group's working directory as >UDD>REMOTE>SMITH. A bound unit, UPMAST, contained in this working directory, is defined as the lead task of the group, and obtains its input data from the device whose pathname is >SPD>VIP06. It writes its output to a file, MSTRLST, also contained in the working directory. The base priority level at which the application runs is 10, and memory is obtained from the memory pool identified as M1.
It is assumed that when the user has completed the entry of transaction items he enters a unique termination code which is interpreted by the UPMAST program as a signal to issue a system service call to the monitor to terminate the execution of the application. When this occurs, the system deletes the task group and returns all of the group's resources to the system for use by other task groups.
When the application has terminated, the output written to the MSTRLST file can be transcribed to hard copy by entering a request to the batch task group to print the contents of the file.

STATUS GROUP

Command Name: STG
Display the status of the indicated task group.

FORMAT:

```
STG id [ctl_arg]
```

PARAMETER DESCRIPTION:

id

The identifier of the group whose status is requested. The batch task group identifier is $B.

[ctl arg]

One or more control arguments chosen from the following list.

-TASKS

Specifies that the statuses of all tasks in the indicated task group are to be listed. This is the default if no control arguments are present.

-FILES

Requests the names of all files that are currently associated with the indicated task group, their types, concurrencies and LFNs.

FUNCTION DESCRIPTION:

The STATUS GROUP command writes to the operator output file a summary of the current status of a task group. In addition to information pertinent to the group as a whole, two other categories of status information are displayed; that relating to tasks within the group and that relating to files currently associated with the group.

The following items provide status information relative to the task group as a whole:

- Task group identification
- Current state of the task group:
  - B = Batch, not rolled out
  - R = Batch, rolled out
  - S = Suspended
  - D = Dormant
  - A = Active
- Memory pool identification
- Current user identification
- Full pathname of the error output file
- Full pathname of the user output file

Task-specific status information consists of the following group of items for each task:

- Task logical resource number (if a created task) or the letters ST (if a spawned task)
- Task priority level
- Current state of the task:
  - D = Dormant
  - S = Suspended
  - W = Waiting
  - A = Active
- Task's bound unit name
- Full pathname of the command input file
- Full pathname of the user input file

If there are no tasks currently associated with the task group, a single item, NO TASKS, is returned.

File-specific information consists of the following group of items for each file:

- o Full pathname of the file
- o Concurrency of the file, represented by a decimal digit in the range 1 through 5. The significance of the digits, for the task group specified by the id parameter, and for other task groups, is as follows:

For Group id	*For Other Groups*
1 = Read Only	Read Only
2 = Read only	Read or Write
3 = Read or Write	No Read, No Write
4 = Read or Write	Read Only
5 = Read or Write	Read or Write

- o File type
- o Logical file number, if one is associated with the file, otherwise spaces
- o Open/closed status of the file

If there are no files currently associated with the task group, a single item, NO FILES, is returned. The task group status information is always returned when this command is used. The task-specific information is returned if no control arguments are given, or if explicitly requested by the -TASKS argument. If the -FILES argument is specified, the file-specific, but not the task-specific, information is given.

STATUS SYSTEM

Command Name: STS
Display general system status.

FORMAT:

STS [ctl_arg]

PARAMETER DESCRIPTION:

[ctl_arg]

One or more control arguments chosen from the following list.

-BA

Specifies that the set of devices available to the batch task group is to be listed. An indication of the active files on each device is given, as well as the volume identifier of the mounted volume on each device.

-ALL

Specifies that the same information as described under -BA above is to be listed for all devices.

-AVAIL

All devices which have no open files associated with them are to be listed, with the volume identifier of the mounted volume. If no control arguments are specified with the command, this is the default.

-SYMPD dev_name

The status of the specific device dev_name is to be listed, including the volume identifier of the mounted volume.

-DISABLED

All devices which are currently in a disabled state are listed.

-GROUP

All task groups are listed, including their pool identifiers and current request user id's.

-LBR

All entries on the batch request queue are listed, including each user id and pathname.

FUNCTION DESCRIPTION:

The STATUS command gives the system operator the ability at any time to ascertain the general status of the system with regard to task groups, their associated peripheral devices, memory pools, and/or entries in the batch request queue.

When the -GROUP control argument is used, the following status information is returned for each group:

- Task group identification
- Current state of the task group:
  - B = Batch, not rolled out
  - R = Batch, rolled out
  - S = Suspended
  - D = Dormant
  - A = Active
- Memory pool identification
- Current user identification

When the -LBR control argument is used, the following information is returned:

- Batch request's user identification
- Command/user input file pathname

This information is repeated for each request currently in the batch task group request queue, and reflects the values specified in the user_id and in_path parameters of each EBR command currently awaiting execution.

All of the remaining control arguments are related to the status of peripheral devices. When any of these is used, a display of the following form is returned:

$$b\left\{\begin{matrix}\text{DSKxxx}\\ \text{dev\_name}\end{matrix}\right\}\text{vol\_id}\left\{\begin{matrix}\text{D}\\ \text{nn}\end{matrix}\right\}$$

The significance of the elements in the above display format is shown below.

b

B if the device is accessible to the batch task group; otherwise spaces

DSKxxx

Device-unit name of a disk device, as specified in a CLM DEVICE directive

dev_name

Dev_name of a non-disk device, as specified in a CLM DEVICE directive

vol_id

Name of the volume mounted on DSKxxx or dev_name

D

Device is currently disabled

nn

00 if device contains no active files (i.e., device is available); or
$01\text{-}99_{10}$, indicating the number of currently active files on the device

If there are no devices satisfying the requested status, the command returns

NO DEVICES WITH STATUS REQUESTED

SUSPEND BATCH

Command Name: SSPB
Temporarily terminate the execution of the batch task group.

FORMAT:

SSPB

PARAMETER DESCRIPTION:
No parameters are required or permitted with this command.

FUNCTION DESCRIPTION:
The SUSPEND BATCH command causes the cessation of execution of any tasks that may be active in the batch task group, after completion of any outstanding input/output requests. The task group remains in the suspended state until it is reactivated by an ACTIVATE BATCH command. All controlling structures remain intact and no memory used by the group is returned to the memory pool during the suspended state. The suspended state of the batch task group does not preclude its being rolled out in the event that its memory is required by an online task group executing in an extendible full memory pool, and requesting additional memory.

SUSPEND GROUP

Command Name: SSPG
Temporarily terminate the execution of the specified online task group.

FORMAT:

SSPG id

PARAMETER DESCRIPTION:
id

The name of a task group previously activated which is to be suspended.

FUNCTION DESCRIPTION:
The suspend group command causes the cessation of execution of any tasks that may be active within the indicated task group, after completion of any outstanding input/output requests. The task group remains in the suspended state until reactivated by an ACTIVATE GROUP command specifying the same group id. All controlling data structures remain intact and no memory used by the task group is returned to the memory pool during the suspended state.

SECTION 4

EXECUTION CONTROL LANGUAGE

This section describes the elements of the Execution Control Language (ECL), by which a user exercises control over the MDT operating system. For the purpose of this section, a user is defined as any person who communicates with the operating system through a peripheral device capable of accepting ECL command lines and delivering them to the MDT operating system, i.e., a card reader or an MDC- or MCP-connected terminal. This device is known as a user terminal.

COMMANDS

ECL commands are read and interpreted by a system software component known as the ECL processor, which executes as the lead task in the batch task group, or can execute as the lead task in an online task group. Each command causes a task to be spawned within this task group to perform the requested function (e.g., create a task within an existing group, enter a group request, dump a file). When the execution of a command terminates, control is returned to the ECL processor, which is then capable of accepting another command.

Command Line Format

A command line to the ECL processor is a string of up to 127 ASCII characters whose general form is

$$arg_1 \ [arg_2 \ ... \ arg_n]$$

where arg_1 is the simple pathname of the bound unit that performs the command's function. It is the character string defined in the "Command Name" entry in each of the command descriptions in this section. Each subsequent arg entry is a parameter or control argument. The functions of parameters and control arguments are described in the following paragraphs.

Parameters

A parameter is a user-supplied character string which supplies a value for some command variable, such as the pathname of a file on which action is to be taken by a command, the identification of a task or task group, or the pathname of an input file required by a command. Parameters are positional in a command line, that is, their positions in the line indicate which variables the values are being supplied to. Some commands require no parameters; others accept one or more as indicated in the syntax in the individual command descriptions. In a few commands, a parameter may be optional, as indicated by its being enclosed in brackets, thus: [path].

Control Arguments

A control argument is used to specify some option applicable to a command, such as the pathname of an alternate input or output file. Control arguments are differentiated from parameters by the presence of a keyword, which is a fixed-form character string preceded by a hyphen, thus: -ECL. A keyword may be followed by a user-supplied value, as -IN path.

Control arguments, when included in a command syntax, are collectively represented by the term 'ctl_arg', and the parameter descriptions define the individual keywords applicable to the command. Unless otherwise noted, keywords are optional and can appear in any order in the command line. If all keywords are optional in a given command, the term ctl_arg is enclosed in brackets in the syntax definition; if an argument is required, the requirement is noted in the individual keyword description and the ctl_arg term is not enclosed in brackets.

Spaces in Command Lines

Arguments in command lines are separated from each other by spaces. Unless otherwise indicated, wherever a space appears in a command line syntax, it represents one or more space characters, or one or more horizontal tab characters, or any combination of these. Spaces can be embedded within an

argument by enclosing the argument in single (') or double (") quote characters. If the enclosing character is also required *within* the argument, it is represented by two successive characters, thus: "NAME=""SMITH""" AREA 203".

STANDARD ECL PROCESSOR FILES

Four files are always associated with the ECL processor. These are:

- o The Command Input file (COMMAND_IN)
- o The User Input file (USER_IN)
- o The User Output file (USER_OUT)
- o The Error Output file (ERROR_OUT)

The functions and characteristics of these files are described in the following paragraphs.

Command Input File

The command input file for the ECL processor is the file from which ECL command lines are read. More specifically, it is the device or file named by the in_path parameter when a request is entered against a task group in which the ECL processor is executing as the lead task. It can at times, however, be assigned temporarily to another device or file, as during the execution of the EC command. At the termination of execution of such a command, the command input file reverts to the original device or file.

User Input File

The user input file is the file from which a command function, during its execution, reads its own input. When a task group request has been processed, and as long as no alternate user input file is specified as a parameter in a subsequent command, the user input file remains the same as the command input file. At the termination of a command which names an alternate user input file, the user input file reverts to its initial assignment.

User Output File

The user output file is the file to which a task group writes its output. It is established by the -OUT control argument of an EBR or EGR command which activates the task group. Certain commands (e.g., the COBOL command) can temporarily reassign the user output file to a different device or file; when these commands terminate execution, user output reverts to its original destination.

The user output file can also be directed to another device through the use of the FILE OUT command. It remains assigned to this device until another FILE OUT command is processed, at which time it can be directed to yet another device, or back to the original device, at the user's discretion.

Error Output File

The error output file is the file to which the ECL processor and any commands invoked by it write information related to error conditions detected by them. The error output file is the same as the initial user output file; it cannot be reassigned by any ECL command or command parameter.

ECL COMMAND FORMATS AND DESCRIPTIONS

The remainder of this section comprises complete descriptions of the formats, parameters, control arguments, and functions of the execution control language commands. In cases in which the command formats are nontrivial, one or more illustrative examples are also given.

The command descriptions given on the following pages are arranged in alphabetic order to facilitate references to specific commands. A summary list of the commands, grouped by functional categories, is given below.

Task Group Creation and Deletion Commands

Function Name	*Command Name*
Abort Online Task Group	ABORT_GROUP
Create Online Task Group	CG
Delete Online Task Group	DG
Spawn Online Task Group	SG

Task Group Execution Commands

Function Name	*Command Name*
Enter Batch Request	EBR
Enter Group Request	EGR
Execution Command	EC
Terminate Group Request	BYE

File and Directory Control Commands

Function Name	*Command Name*
Associate Path and LFN	ASSOC
Change Working Directory	CWD
Create Directory	CD
Create File	CF
Dissociate Path and LFN	DISSOC
Change User Output File	FO
List Names	LS
List Search Rules	LSR
List Working Directory	LWD
Modify File	MF
Release (Delete) File	RL
Rename File	RN

Utility Commands

Function Name	*Command Name*
Compare Files	CPA
Copy File	CP
Create Volume	CV
Edit System Dump	DPEDIT
Export PAM File	EX_PAM
File Dump	FD
Import PAM File	IM_PAM
Print File	PR
Reset Bit Map	RS
Sort File	SORT

Program Preparation Activity Commands

Function Name	*Command Name*
Assemble Source Unit	ASSEM
COBOL Source Unit Compile	COBOL
Cross-Reference	XREF
Edit Source Unit	ED
FORTRAN Source Unit Compile	FORTRAN
Link Object Unit(s)	LINKER
Macro Preprocessor	MACROP
Patch Object/Bound Unit	PATCH
RPG Source Unit Compile	RPG

System and Status Commands

Function Name	*Command Name*
Modify External Switches	MSW
Ready Off	RDF
Ready On	RDN
Status Group	STG
Time Display	TIME

ABORT GROUP

Command Name: ABORT_GROUP
Suspend the indicated online task group and terminate it.

FORMAT:

ABORT_GROUP [id]

PARAMETER DESCRIPTION:
[id]

The group identification of a task group previously created by a CG command specifying the same id. If this parameter is omitted, the issuing task group is aborted.

FUNCTION DESCRIPTION:
The ABORT GROUP command causes the suspension and termination of an existing online task group, whether it is active or dormant. It removes all of the data structures which define and control the execution of the task group, and returns all memory used by the group to the appropriate memory pool. Any files which may have been open during the execution of the task group are closed. Any requests which may be pending against the group are cancelled. The action of the ABORT GROUP command is thus similar to the DELETE GROUP command, the difference being that the latter must wait until the task group becomes dormant, while the former takes effect as soon as all outstanding input or output orders are complete.
This command can be issued only from an online task group.

Example:
ABORT_GROUP AX

A task group identified as AX is terminated immediately.

ASSEMBLER

Command Name: ASSEM
Assemble the source program unit represented by the indicated file name, applying the specified options.

FORMAT:

ASSEM path [ctl_arg]

PARAMETER DESCRIPTION:
path

Specifies the name of the file containing the source unit to be assembled.

[ctl_arg]

One or more control arguments chosen from the following list.

{-NO_OBJ / -NO}

Indicates that the generation of the object text unit is to be supressed. If omitted, the object text unit is generated.

{-NO_LIST / -NL}

Indicates that the source listing is to be suppressed. If omitted, the source listing is written to the file path.L.

{-LIST_ERRS / -LE}

Specifies that the list file contains only statements which have assembly errors associated with them.

{-SAF / -LAF}

Indicates the addressing mode in which the program is to be assembled. -SAF indicates short (one-word) address form; -LAF indicates long (two-word) address form. If omitted, the program is assembled in the same mode as that in which the Assembler is running.

{-SIZE nn / -SZ nn}

Specifies the number (01 through 64) of 1024-word memory blocks that are to be used for the assembler's symbol table. If omitted, the available memory in the task group's memory pool minus 400_{10} words is used.

-COUT out_path

Indicates that listings which would normally be written to the file path.L are to be written to out path, in the working directory.

FUNCTION DESCRIPTION:

The ASSEMBLER command is used to invoke the GCOS 6 assembler component. Execution of the Assembler is normally intended for the batch task group, as is the execution of most of the other components used in a program development activity.

The path parameter can assume any of the acceptable forms of a pathname; a simple name indicates that a source program unit residing in the working directory is to be assembled. Wherever it exists, it must be suffixed with a .A suffix, indicating that it is an assembly language source unit. The path parameter must be given without the .A suffix; the Assembler appends the suffix prior to searching the directory for the source unit.

If the -COUT control argument is not specified, the source listing (if requested) is written to a file created by the Assembler in the working directory, having a file name of the form path.L. The path portion is the last or only element in the path parameter. The file can be subsequently listed on a line printer by using the PRINT utility command. If a different file is specified by using the -COUT argument, out_path is the name of the file containing the listing. The Assembler does not append a .L suffix to out_path.

The object text unit generated by the assembler is written to a file created by the assembler, whose name is of the form path.O and which is contained in the working directory.

If files of the form path.L and path.O already exist, they are overlaid by the output generated by the current assembly.

A full description of the operation and use of the Assembler is contained in the Assembly Language manual.

Example:

```
ASSEM MYPROG -SIZE 5 -COUT >SPD>LPT01
```

An assembly language source program, MYPROG.A, residing in the current working directory is to be assembled. The source listing and errors are to be written to the printer LPT01, and the object text unit is to be written to the file MYPROG.O in the working directory. If MYPROG.O already exists as a result of a previous assembly, it is overlaid with the new object text unit. Five 1024-word blocks of memory are to be used for symbol resolution during the assembly.

ASSOCIATE PATH

Command Name: ASSOC

Associate the specified pathname and logical file number.

FORMAT:

ASSOC lfn path

PARAMETER DESCRIPTION:

lfn

The logical file number by which a task is to refer to a file.

path

The pathname of the file to which the task is to refer.

FUNCTION DESCRIPTION:

The ASSOCIATE PATH command permits a task group to refer to files by the use of a standard interface known as a logical file number (LFN). The LFN serves as a "bridge" across which an input or output statement in a user program can gain access to an external file without the need to know its full pathname.

Conventions by which user files are identified and referred to in source programs are dependent upon the language processor by which the source program is compiled or assembled. Each processor relates an internal file identification by one means or another to a number (the LFN) which can be used in an ASSOC command to equate the internal file identification to an external pathname. In a COBOL program, for example, this number is derived from the ifn value in the ASSIGN clause of a SELECT statement.

The task group within which an ASSOC command is to be issued must have been created specifying (or defaulting to) an LFN parameter value large enough to include the highest LFN which is expected to be given in any ASSOC command issued during the life of the task group. This requires a knowledge of what programs are to be executed within the group and the numerical LFN values which these programs have generated.

The path parameter can specify a simple, relative or absolute pathname. If a simple name is specified, the file is assumed to reside in the user's working directory. No check is made at the time the ASSOC command is issued as to whether a file exists or not; this is only verified at the time the file is opened. If a relative pathname is given, the pathname is made absolute when the file is opened.

Example:

ASSOC 12 MYFILE

A file defined in a user program has been assigned a logical file number 12 by the language processor that compiled the program (e.g., the COBOL Compiler). A file, MYFILE, exists in the issuing task group's working directory. The ASSOC command relates the LFN (12), by which the program's input and output statements refer to the user file, to the external file whose pathname is >UDD>project>person>MYFILE.

Note that the symbolic name by which the file is identified and referred to in the program (e.g., INPUT-DATA) bears no relationship to the name by which it is referred to by the File System.

BYE (TERMINATE CURRENT GROUP REQUEST)

Command Name: BYE

Terminate the execution of the current request in the issuing task group.

FORMAT:

BYE

PARAMETER DESCRIPTION:

No parameters are required or permitted with this command.

FUNCTION DESCRIPTION:

The BYE command causes the cessation of execution of the issuing task group. It removes all defining and controlling data structures except those associated with the lead task and returns all associated

memory to the task group's memory pool. Any files that are open and in use by this task group are closed. At the conclusion of execution of the BYE command, the lead task processes the next request against this group, if any.

CHANGE WORKING DIRECTORY

Command Name: CWD
Change the default working directory to the specified path.

FORMAT:

CWD [path]

PARAMETER DESCRIPTION:
[path]
The path name of the new working directory. If this parameter is omitted, the path name >UDD>project>person, established by the task group's user id, is assumed.

FUNCTION DESCRIPTION:
The CHANGE WORKING DIRECTORY command enables the user to move his point of reference to some other directory level within his own project's directory or to some specified point within an entirely different directory. Moving the reference point in a directory enables a task to refer, using simple names, to entities in the directory at levels other than the >UDD>project>person level established when the task was activated initially, or to entities which exist in some other directory. The use of the CWD command without a path name moves the reference point back to the original >UDD>project>person level.
If a simple pathname (one that does not begin with a > sign) is given as a parameter, the effect is to change the reference point within the current directory hierarchy. It normally implies that there exist one or more directories subordinate to the >UDD>project>person level. That is, if a user issued a command CWD MANUALS, there is assumed to exist a directory path name >UDD>project>person>MANUALS within the hierarchy being used by this task. After the CWD command is executed, files that exist within the MANUALS subdirectory can be referred to by the task using simple file names.
It is also possible to traverse the hierarchy in the opposite direction, that is, in a direction toward the root. This is done by specifying as the parameter the single character < (less than sign). Thus it is possible to revert to the original directory level after having issued the CWD command described above by issuing a second command, CWD <. Each occurrence of the < sign moves the point of reference one level up (toward the root).
If an absolute pathname (one that begins with the > sign) is given as a parameter, the effect is to move the point of reference directly to the specified point in the named directory. This directory may or may not be the same as the one being used by the issuing task.

Example:
Assume the directory structure shown in the figure below. A task group whose user id is SMITH.AUTHORS is active and is at the default directory level >UDD>AUTHORS>SMITH, established when the task group was initiated.

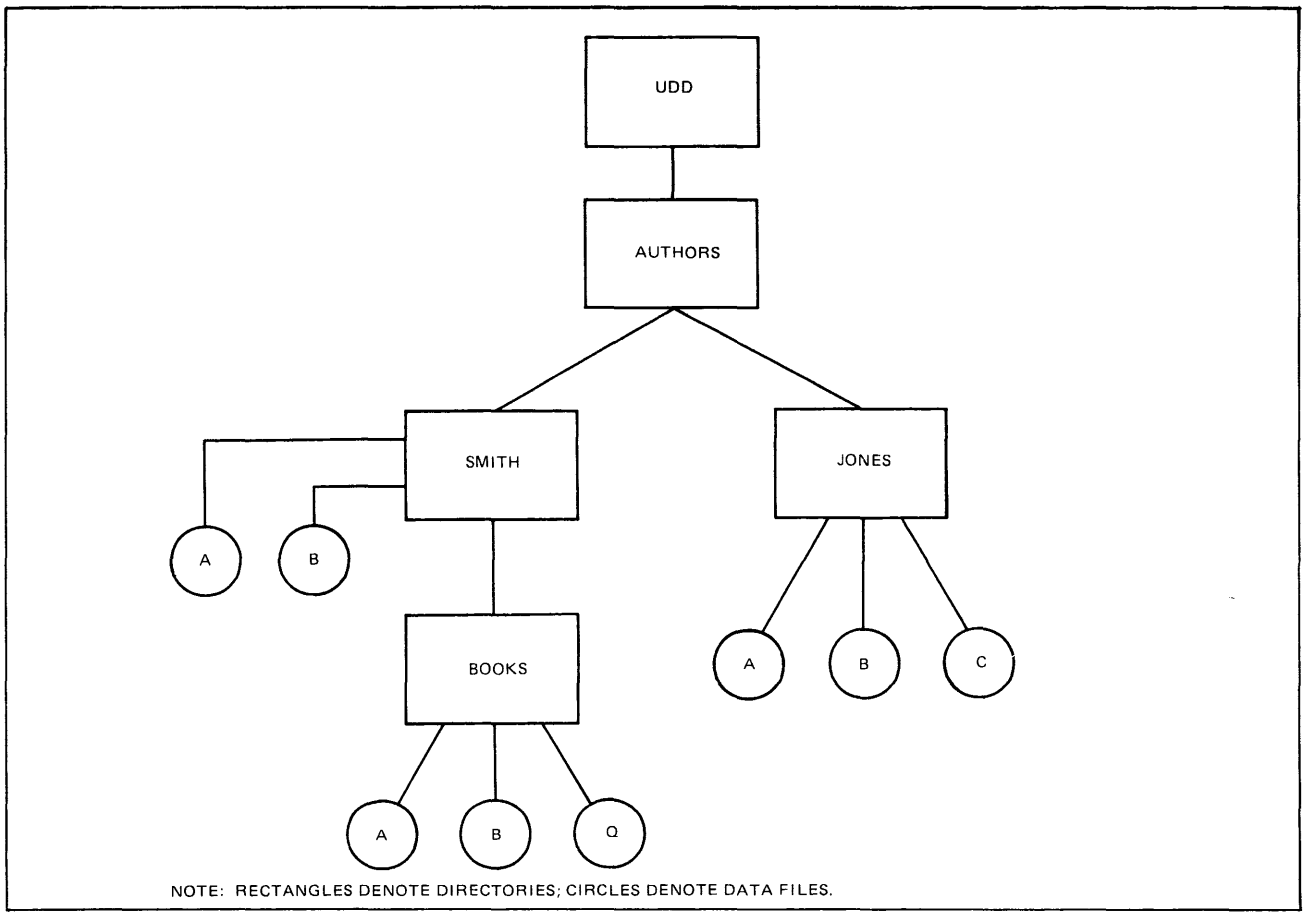


Figure 4-1. Typical Directory/File Structure

A sequence of CWD commands such as that shown below is issued. A description of the resulting action is given opposite each command.

Command	*Resulting Action*
CWD BOOKS	The point of reference is moved to the BOOKS subdirectory level (one level below the default SMITH level). Files named A, B, and Q can now be referred to by their simple names. The system supplies >UDD>AUTHORS> SMITH>BOOKS from the working directory in the construction of full pathnames for the three files.
CWD <	The point of reference is moved up one level, back to the original SMITH level (note that the same effect could have been obtained with a CWD command without the < sign). The files named A and B in the SMITH directory (not the same files as A and B at the BOOKS subdirectory) can now be referred to by simple names.
CWD>UDD>AUTHORS>JONES or CWD <JONES	The absolute form of the pathname moves the point of reference directly to the JONES directory level. The second form achieves the same result by moving up one level to AUTHORS and then down one level to JONES.

COBOL

Command Name: COBOL
Compile the COBOL source program unit represented by the indicated file name, applying the specified compiler options.

FORMAT:

COBOL path ctl_arg

PARAMETER DESCRIPTION:

path

Specifies the name of the file containing the source unit to be compiled.

ctl_arg

One or more control arguments chosen from the following list.

{-NO_OBJ / -NO}

Indicates that the generation of the object text unit is to be suppressed. If omitted, the object text unit is generated.

{-LIST_OBJ / -LO}

Indicates that a listing of the object text output is to be obtained. Object text is interspersed with source text in the listing. If omitted, object text is not listed.

{-LIST_ERRS / -LE}

Specifies that only those source lines containing compilation errors, together with their error codes, are to be listed. If omitted, and -NL is not specified, the complete source program is listed, followed by a listing of the error lines and codes.

{-NO_LIST / -NL}

Specifies that all listings are to be suppressed. If omitted, and -LE is not specified, the complete source unit is listed, followed by a listing of the error lines and codes.

{-SIZE nn / -SZ nn}

Designates the number of additional 1024-word blocks of memory to be used for tables by the compiler. The value of nn can be between 04 and 53, inclusive. If omitted, at least 3000 words of table space must be available.

-DB

Indicates that debugging lines are to be compiled as comments, ignoring the WITH DEBUGGING MODE clause in the source program.

-LD

Specifies that, in addition to the source text, errors, and object code (if specified) listings, a data map list is also to be produced.

-COUT out_path

Indicates that listings which would normally be written to the file path.L are to be written to out_path.

FUNCTION DESCRIPTION:
The COBOL command is used to invoke the GCOS 6 COBOL Compiler component. Execution of the compiler is normally intended for the batch task group, as is the execution of most of the other components used in a program development activity.

The path parameter can assume any of the acceptable forms of a pathname; a simple name indicates that a source program unit residing in the working directory is to be compiled. Wherever it exists, it must be suffixed with a .C suffix, indicating that it is a COBOL langauge source unit. The path parameter must be given *without* the .C suffix; the compiler appends the suffix prior to searching the directory for the source unit.
If the -COUT control argument is not specified, the requested listings are written to a file created by the compiler in the working directory, having a file name of the form path.L. The path portion is the last or only element specified in the path parameter. This file can be subsequently listed on a line printer by using the PRINT utility command. If a different file is specified by using the -COUT argument, the listings are written to a compiler-created file whose pathname is out path. The compiler does not append a .L suffix.
The object text unit generated by the compiler is written to a compiler-created file whose name is of the form path.O, and is contained in the working directory.
If files of the form path.L and path.O already exist in the current working directory, they are overlaid by the output generated by the current compilation.

Example:

COBOL CBPROG -NO_OBJ -LD -COUT >SPD>LPT01

A COBOL source program, CBPROG.C, is to be compiled. The source text file is located in the working directory. Listings are to include source statements, error diagnostics and a data map, and are to be written to the line printer LPT01. No object text unit is to be generated.

COMPARE

Command Name: CPA
Compare the contents of one file or volume with that of another file or volume.

FORMAT:

CPA path new [ctl_arg]

PARAMETER DESCRIPTION:

path
Indicates the name of the file or volume to be compared.

new
Indicates the name of the file or volume against which that specified by the path parameter is to be compared.

[ctl_arg]
One or more control arguments chosen from the following list.

{-VOLUME
-VOL}
Indicates that an entire volume is to be compared, a track at a time, with another. If this argument is specified, both the path and new parameters must be of the form >SPD> dev name>vol id.

-CI
Indicates that a compare by control interval is to be performed. This argument can be specified under any of the following conditions:

1. The path parameter represents a Series 60-compatible file and the new parameter represents either a file of the same type or a magnetic tape.
2. The path parameter represents a magnetic tape containing control intervals from a type "1" COPY (refer to the COPY command description), and the new parameter represents a Series 60-compatible file of the type copied to the tape.
3. Both path and new parameters represent magnetic tape files or volumes.

{-LIMIT nn
-LI nn}

Specifies that only nn records or control intervals are to be compared (if end of file is not encountered first).

{-FROM nn
-FM nn}

Specifies that the first nn records or control intervals of the file are to be bypassed before beginning the compare.

{-PRINT nn
-PR nn}

Specifies that only the first nn miscompared records are to be printed. The compare operation terminates when the end of the file or volume is encountered.

-NWD

This argument applies only to tape files and specifies that the tapes are not rewound at the end of the compare.

FUNCTION DESCRIPTION:
The COMPARE command compares two files or volumes, record by record or control interval by control interval, and, if specified by the -PR control argument, writes the contents of any miscompared records on the user output file. If a user is at an interactive terminal and his user output file is the terminal, he can direct the written output to another device, such as a line printer, by issuing an appropriate FILE OUT command prior to issuing the COMPARE command. At the termination of the CPA command, a message is issued to the user output file, giving the number of miscompared records or control intervals, if the number is nonzero.
If a volume compare is to be performed, then the path and new parameters must represent the pathnames of peripheral devices. The dev_name portion of the pathname is the symbolic name (e.g., DSK01) given to the device in question at system configuration. The vol_id portion of the parameters represents the identification of the volumes to be compared.

Example 1:

```
CPA FILEA FILEB
```

Compare two files in the working directory.
The full pathnames of FILEA and FILEB are constructed using elements of the working directory. The files are compared record by record and a summary message is issued, giving the number of miscompared records, if any.

Example 2:

```
FO >SPD>LPT01
CPA FILEA >UDD>BOOKS>JONES>FILEA -PR 20
```

FILEA in the working directory is compared to FILEA in the directory >UDD>BOOKS>JONES. The first 20 miscompared records are written to the line printer LPT01.

COPY

Command Name: CP
Copy a file or volume.

FORMAT:

```
CP path new [ctl_arg]
```

PARAMETER DESCRIPTION:

path

Specifies the name of the file or volume to be copied.

new

Specifies the new pathname of the file or volume being copied.

[ctl_arg]

One or more control arguments chosen from the following list.

{-VOLUME / -VOL}

Indicates that an entire volume is to be copied a track at a time. If this argument is specified the path parameter must be of the form >SPD>dev_name>vol_id, and the new parameter must be of the form >SPD>dev_name[>vol_id]. .

-CI

Indicates that a copy by control interval is to be performed. This argument can be specified under any of the following conditions:

1. The input is a Series 60-compatible file and the output is a file of the same type or a magnetic tape.
2. The input is a magnetic tape created by a copy under condition 1 above, and the output is a Series 60-compatible file of the same type as that which was copied to the tape.
3. Both the path and new parameters represent magnetic tape devices. In this case the copy will be to end of volume.

-NWD

This argument applies only to tape files and secifies that the tapes are not rewound at the end of the copy.

{-VERBATIM / -VRB}

This argument applies only to card input files and specifies that cards are to be read in verbatim mode at the control interval level.

FUNCTION DESCRIPTION:

The COPY command permits the creation of backup copies of files or volumes, either on magnetic tape or on another mass storage device. It can also be used to create copies of files in the same directory or in other directories.

The path and new parameters may express or imply the same directory portion of the file's pathname. If they do, then the file name portions of both must be different, and furthermore the file name portion of the new parameter must not already exist within the directory. If the path and new parameters represent different directories, then the file name portions of both may be the same, but the same requirement exists regarding the uniqueness of the file name in the directory represented by the new parameter.

If a volume copy is to be performed, then the path and new parameters must represent the pathnames of peripheral devices. The dev_name portion of the pathname is the symbolic name (e.g., DSK01) given to the device at system configuration. The vol_id portion of the path parameter is the volume identification of the volume being copied. If the new parameter names a magnetic tape device and the pathname includes the vol_id portion, the volume label is read and verified. The copied data then follows the volume label; i.e., the volume label is preserved. If the vol_id portion is omitted, copying begins at the current position on the tape (normally beginning of tape). In this case, the tape volume label, if any, is not preserved.

Example 1:

```
CP FILEA FILEB
```

Copy a file within the working directory. The full pathnames of FILEA and FILEB are constructed using elements of the working directory. The result of this copy is the existence of two identical files under different names.

Example 2:

CP FILEA >UDD>BOOKS>JONES>FILEA

Copy a file from the working directory to another directory on the same volume. FILEA in the working directory is copied to the directory >UDD>BOOKS>JONES, retaining the same name, FILEA, assuming that the file name does not already exist in that directory.

Example 3:

CP SUB_DIR1>FILEA^ VOL003>UDD>BOOKS>JONES>FILEB

Copy a file from a subdirectory in the working directory to a directory on another volume. FILEA, one directory level below the working directory, is copied to the directory >UDD>BOOKS>JONES on a volume whose volume id is VOL003. It is assigned the name FILEB in the new directory.

Example 4:

CP >SPD>DSK03>VOL001 >SPD>DSK05 -VOL

Copy the contents of one mass storage volume to another (like) mass storage volume. The contents of the volume VOL001, mounted on the device represented by symbolic device name DSK03, are copied to the volume mounted on the device represented by symbolic device name DSK05.

CREATE DIRECTORY

Command Name: CD
Create a new directory identified by the specified pathname.

FORMAT:

CD path

PARAMETER DESCRIPTION:
path
The pathname of the new directory to be created.

FUNCTION DESCRIPTION:
The CREATE DIRECTORY command can be used under any circumstances in which the creation of a new directory, or a subdirectory within an existing directory, is required. On a newly created volume, whose directory consists of only the root entry (volid) it can be used to introduce the UDD directory level, as well as any number of project- and user-level entries (see example 4, below). On a volume which already contains user directories, it can be used to introduce new user-level entries within a project-level directory, or new project-level entries within the UDD-level directory.
The form of the path entry of this command is the factor which determines the level of the directory being created. If it is a simple name, the name is concatenated with the entries constituting the working directory, resulting in a new directory one level below that of the working directory. A pathname consisting of more than one element results in the creation of the directory named by the last pathname element, and requires that all preceding directories named already exist (see examples 3 and 4, below).

Example 1:

CD SMITH1

Create a directory within the working directory. If the current working directory is >UDD>BOOKS>SMITH, the resulting directory is >UDD>BOOKS>SMITH>SMITH1.

Example 2:

```
CD<JONES
```

Create a new user-level directory at the same level as the working directory. If the working directory is >UDD>BOOKS>SMITH, the resulting new directory is >UDD>BOOKS>JONES.

Example 3:

```
CD<JONES
CD<JONES>JONES1
```

Create a new user-level directory at the same level as the working directory, and one subdirectory. If the working directory is >UDD>BOOKS>SMITH, the resulting directory is >UDD>BOOKS>JONES>JONES1. Note that two steps are required, since two directory levels are being created.

Example 4:

```
CD^ USER03>UDD
```

Create a new user directory directory on another volume which has only a volume-id, USER03. Additional project/user directories can be created on the new volume by issuing pairs of commands of the form

```
CD^USER03>UDD>project
CD^USER03>UDD>project>person
```

for each new directory desired. Or, if a command

```
CWD^USER03>UDD
```

is issued first, the additional project/user directories can be created using pairs of commands of the form

```
CD project
CD project>person
```

CREATE FILE

Command Name: CF
Create the specified mass storage file.

FORMAT:

```
CF path [ctl_arg]
```

PARAMETER DESCRIPTION:

path

Specifies the pathname of the file to be created.

[ctl_arg]

One or more control arguments chosen from the following list.

-F_REL

Creates a BES native fixed relative file without deletable records.

-N_REL

Creates a BES native fixed relative file with deletable records.

-SEQ

Creates a Series 60-compatible variable length record file with spanned records that is processed sequentially.

-REL

Creates a Series 60-compatible variable length record file that can be processed sequentially or relatively.

{-INDEX / -IX}

Creates a Series 60-compatible variable or fixed length record file which can be processed sequentially or index sequentially.

{-CI_SIZE n / -CSZ n}

The number of bytes in a control interval for -SEQ and -INDEX type files. The value of n must be a multiple of 256 bytes. If not specified, the default is 512 bytes.

{-REC_SIZE n / -RSZ n}

The number of bytes per record for -F_SEQ, -F_REL, and -N_REL type files. For -SEQ, -REL, and -INDEX type files it specifies the maximum record size in bytes. If not specified, the default is 256 bytes.

{-SIZE n / -SZ n}

The initial size of the file in units of control intervals for -SEQ, -REL, and -INDEX type files, or records for -F_REL and -N_REL type files. Default is no initial allocation.

{-INC_SIZE n / -ISZ n}

The number of units by which the file size is to be incremented whenever it must be expanded to accommodate more data. If not specified the value of n is the same as that specified for -SIZE. If -SIZE is not specified, n is set to 40 physical sectors.

{-MAX_SIZE n / F-MSZ n}

The maximum size which this file can attain, in units of control intervals for -SEQ, -REL, and -INDEX type files, or records for -F_REL and -N_REL type files. It must be set equal to the initial size, as specified by the -SIZE control argument, if a BES1-readable file is being created. If this argument is not specified, the file can expand to the physical limit of the volume.

{-KEY_OFFSET n / -KO n}

The byte offset of the first byte of the key field within the record. The first byte of a record is byte 1. This argument is required for -INDEX type files.

{-KEY_SIZE n / -KSZ N}

The number of bytes constituting the key field. This argument is required for -INDEX type files.

{-FILL_PC n / -FPC n}

The ratio of data bytes to total bytes to be put into each control interval when creating an -INDEX type file, expressed as a percentage. If not specified, the default value is 100.

FUNCTION DESCRIPTION:
The CREATE FILE command reserves space in the file system for the specified file in accordance with the control arguments supplied in the command. It establishes a pathname whose form is dependent upon the form of the path parameter and the elements of the working directory.
If a simple name is specified as the path parameter, it is concatenated with the elements of the working directory to form the full pathname of the file. If a relative name is given, any directories expressed or implied by that relative name must exist, as must any directories expressed if the path parameter is an absolute pathname.
The CF command, in effect, creates an "empty" file, which can be subsequently loaded by output statements or macro calls in user programs.
The initial shareability and permission attributes of the created file are such that the file may be referred to from both online and batch tasks, and may be read from and written to by any task. These attributes can be modified through the use of the MODIFY FILE command if different attributes (e.g., write protection) are desired.
The control arguments -F_REL, -N_REL, -SEQ, -REL, and -INDEX are mutually exclusive. If none is specified a -SEQ type file is created.

Example 1:

```
CF FILE01 -SEQ -CI_SIZE 1024 -SIZE 100
```

Create a file at the current level in the working directory. If the working directory is >UDD>BOOKS>JONES, the full pathname of the created file is >UDD>BOOKS>JONES>FILE01. It is a sequential file whose control interval size is 1024 bytes and whose initial size is 100 control intervals. It can be incremented in steps of 100 control intervals up to the physical limit of the volume (default values for -ISZ and -MSZ control arguments).

Example 2:

```
CF SUB_DIR1>MYFILE -IX -SIZE 50 -KO 9 -KSZ 6 -MSZ 200
```

Create a file in an existing directory one level below the current level in the working directory.
Given the same working directory as in the previous example, the full pathname of the created file is >UDD>BOOKS>JONES>SUB DIR1>MYFILE. It is an indexed file whose initial size is 50 control intervals of 512 bytes, and whose increment size and maximum size are 50 and 200 control intervals, respectively. The first byte of the record key is the ninth byte of the record (the first byte of a record is byte 1), and the key is six bytes long.

CREATE GROUP

Command Name: CG
Perform the initialization functions necessary to the initiation of an online task group.

FORMAT:

```
CG id phys_lvl [ctl_arg]
```

PARAMETER DESCRIPTION:

id

The group identification of the new task group. It is a two-character name that cannot have the $ as its first character.

phys_lvl

The base priority level relative to which all tasks in this task group will execute.

[ctl_arg]

One or more control arguments chosen from the following list.

{-EFN root
-EFN root?entry}

The name of a bound unit root segment to be loaded as the lead task if it is not already loaded. The root segment name can be suffixed with ?entry, where entry is a symbolic start address within the root segment. If not given, the start address established when the bound unit was linked is assumed.

-ECL

The root segment of the execution control language (ECL) processor is to be loaded as the lead task.

-LRN n

Specifies the highest logical resource number (LRN) which will be referred to by any task in the task group. The minimum value which can be specified for n is the highest LRN used by the system task group; this is also the default if this argument is not specified.

-LFN n

Specifies the highest logical file number used by any task in the task group. If -LFN is not specified, n assumes the value 15.

-POOL id

id is a two-character ASCII identifier and is the name of the memory pool from which all dynamic memory required by this task group is to be taken. If specified, id must have been defined by a CLM MEMPOOL directive. If not, the issuing task group's memory pool is used.

NOTE: In any invocation of the CG command, -EFN *or* -ECL, but not both, can be specified. If neither is specified, -ECL is assumed.

FUNCTION DESCRIPTION:

The CREATE GROUP command causes the initialization and allocation of all data structures used by the system to define and control the execution of the task group. It causes the loading of the root segment of the lead task of the task group. It does *not* cause the system to activate any task within the task group.

This command can be issued only from an online task group.

Example:

```
CG AX 15 -EFN MAIN_PG?ENTRY1 -LRN 8 -POOL A2
```

A task group identified as AX is created. The lead task of the group is the program MAIN_PG, whose execution is to be started at the symbolic address ENTRY1. No task in the group will execute at a priority level lower than 15, nor refer to a logical resource number higher than 8. Memory will be obtained from a pool identified as A2 at system configuration.

CREATE TASK

Command Name: CT

Perform the initialization functions necessary to the initiation of a task within the issuing task group.

FORMAT:

```
CT lrn rel_lvl ctl_arg
```

PARAMETER DESCRIPTION:

lrn

The logical resource number (LRN) by which the issuing task group can refer to the created task. It cannot exceed the value specified by the -LRN control argument in the CREATE GROUP command which created the group of which this task is a member.

rel_lvl

The priority level, relative to the task group's base priority level, at which the created task is to execute.

ctl_arg

One or more control arguments chosen from the following list.

{-EFN root
-EFN root?entry}

The name of the bound unit root segment to be loaded for execution. The root segment name can be suffixed with ?entry, where entry is a symbolic start address within the root segment. If no suffix is given, the default start address, established when the bound unit was linked, is assumed.

{-SHARE lrn [ssa]
-SHR lrn [ssa]}

The same bound unit is used as for the task identified by lrn. (This task must have been previously defined by a CREATE TASK command specifying this lrn.) ssa is the symbolic start address within the root segment of the task lrn. If none is given, the root segment's default start address, established when the shared bound unit was linked, is assumed.

NOTE: In any invocation of the CT command, -EFN *or* -SHARE, but not both, must be specified.

FUNCTION DESCRIPTION:

The CREATE TASK command causes the allocation and initialization of the data structures which define and control the execution of a task. It causes the loading of the root segment specified by the -EFN control argument. It does *not* activate the task (the ENTER TASK REQUEST command is required to perform activation).

One or more CT commands can be issued to create one or more tasks within the task group. These tasks can be requested for execution concurrently or serially by entering the appropriate control argument in the ETR command which is used to activate each task. Refer to the description of the ETR command.

Use of the -SHARE control argument requires that the bound unit represented by the accompanying logical resource number (LRN) be declared as sharable at the time the bound unit was linked. A sharable bound unit is loaded into the system's memory pool if there is sufficient memory available; otherwise it is loaded into the issuing task group's memory pool. In the latter case it effectively becomes nonsharable (for the current execution).

Example:

```
CT 10 02 -EFN PROG10
CT 11 03 -EFN PROG11
CT 12 02 -SHARE 10 ENTRY2
```

Three tasks are made known to the issuing task group. Their logical resource numbers (LRNs) are 10, 11, and 12. Task 10 is to execute at priority level 02 relative to the base priority level established when the task group was created. Task 11 is to execute at relative level 03, and task 12 is to execute at the same relative level as task 10. If the task group's base priority level was specified as 20, then the three tasks execute at physical priority levels of 22, 23, and 22, respectively. Task 12 is to share the same bound unit as task 10; however, execution of task 12 begins at a different point in the bound

unit, specified by the label ENTRY2 (task 10's entry point is the default entry point established when PROG10 was linked, and PROG10 must have been declared sharable when linked). Subsequent ENTER TASK REQUEST commands cause the execution of the above tasks to begin (refer to the description of the ETR command).

CREATE VOLUME

Command Name: CV
Create or modify a mass storage volume.

FORMAT:

CV path ctl_arg

PARAMETER DESCRIPTION:

path

The pathname of the device upon which the volume to be created is mounted. The form of the pathname is

>SPD>dev_name[>vol_id]

ctl_arg

Exactly one control argument from the following list.

{-FORMAT vol_id [x] | -FT vol_id [x]}

Assign vol_id as the volume id and major directory name. Preformat the volume by initializing all sectors to zeros, checking for bad tracks. The optional x character defines the format of a magnetic tape volume. The possible values of x are 1, 2, 3, or H, and specify the following formats:

1. American National Standard Institute level 1
2. American National Standard Institute level 2
3. American National Standard Institute level 3

H Honeywell derivative of American National Standard Institute level 3.

If used when formatting a disk volume, the optional x character is ignored.

{-BOOT x'hhhh' | -BT x'hhhh'}

Create bootstrap records and write them to volume-relative sectors 0 through 6. The existing volume id and major directory name are not modified. The x'hhhh' field defines certain bootstrap options as described in the function description.

{-MDUMP xx | -MD xx}

Create a memory dump bootstrap record and write it to volume-relative sector 0. The existing volume id and major directory name are not modified. xx specifies the number of 4096-word modules to be dumped.

{-RENAME y | -RN y}

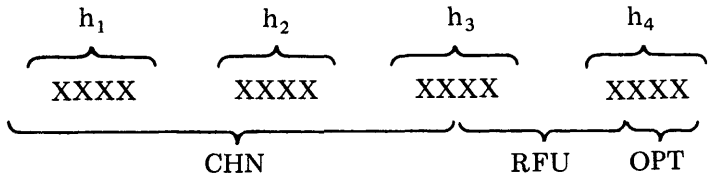
Change the volume id and major directory name to that specified by y. y is a one- to six-character ASCII string.

FUNCTION DESCRIPTION:

The CREATE VOLUME command initializes a mass storage volume in one of several ways. A previously unused volume can be assigned a volume identification through the use of the -FORMAT

control argument. This argument, in addition to initializing all tracks on the volume and verifying their integrity, writes a volume label record containing the volume identifier specified by the vol_id field in this argument. It also establishes this identifier as the volume major (root) directory name. Thus, if vol_id is given the value USER01, the volume label contains this value as the volume identifier, and the root directory pathname for this volume is ^USER01.

A volume which has already been assigned a volume identifier as described above can be supplied with a bootstrap routine in one of two forms. The -BOOT control argument causes a standard system bootstrap routine to be written on the volume. The x'hhhh' field is used to define the channel of the disk device containing the directive files and routines used during system initialization, and to define certain bootstrap and initalization options. The field consists of four hexadecimal digits whose bit configuration is broken down as follows:



CHN

Ten bits (bits 0 through 9) which specify the channel number of the initialization device (e.g., 0400, 1280). The fourth digit of the channel number is always zero, and the values that can be assumed by the third digit are 0, 4, 8, and C (hexadecimal).

RFU

These bits (bits 10 through 12) are reserved for future use and must be zero.

OPT

These bits (bits 13 through 15) establish the bootstrap/initialization options as follows:

If bit 13 = 1: Halt at the conclusion of the system bootstrap routine, and before entering the operating system initialization code. At this halt the D1 register can be set to a channel number as defined under CHN, above, or bits 14 and 15 can be set to specify other options.

If bit 14 = 1: Ignore any CLM USER directive file in the system initialization directory (SID) and use the Honeywell-supplied directive file on the device specified by CHN.

If bit 15 = 1: Bootstrap from the fixed cartridge disk device specified by CHN.

The -MDUMP control argument causes a special record, which bootstraps the memory dump routine, to be written on the volume. A file, DUMPFILE, is allocated with a sufficient number of sectors to contain the number of memory words specified by the xx field of the -MDUMP argument.

A volume already having a volume identifier can be given a new identifier through the use of the -RENAME control argument. This causes the volume identifier field of the volume header record, and the root directory name, to be changed to the identifier specified by the y field of this argument.

The CV command must specify the pathname of the peripheral device (cartridge disk or diskette) upon which the volume to be initialized is mounted. The dev_name portion of the path parameter is the symbolic name of this device as defined by a CLM DEVICE directive at system configuration. The vol_id field of the path parameter, if used, indicates that the volume already has a volume identifier, and that this identifier is to be checked for agreement with the specified identifier. If the two identifiers do not agree, an error message is issued and the command is terminated. The vol_id field of the path parameter does *not* assign an identifier or root directory name to the volume; this can only be done by using the -FORMAT control argument.

Example 1:

```
CV >SPD>DSK03 -FT USRDTA
```

A volume mounted on the device identified at system configuration as DSK03 is to be formatted and assigned the identifier and root directory name USRDTA. If this volume is to contain only user data (i.e., it is not to be used for system initialization or dumping of memory), no further initialization is required. That is, no bootstrap records need be created for this volume. Other directories can be established under the root directory USRDTA by subsequent use of the CREATE DIRECTORY command.

Example 2:

```
CV >SPD>DSK02 -FT DMPVOL
CV >SPD>DSK02>DMPVOL -MD 04
```

A volume mounted on the device identified as DSK02 is to be formatted and assigned the identifier and root directory name DMPVOL. This volume is to be used for dumping memory, and is therefore (by the second CV command) given a memory dump bootstrap record. Dumps are to contain four 4096-word modules of memory. The second command also specifies that the previously assigned volume identifier is to be verified prior to creation of the memory dump bootstrap record.

CROSS-REFERENCE PROGRAM

Command Name: XREF
Create a cross-reference listing for an assembly language or a macro source unit.

FORMAT:

XREF path{.A / .P} [ctl_arg]

PARAMETER DESCRIPTION:

path{.A / .P}

The pathname of the file containing the source unit to be cross-referenced. The last two characters must be a suffix; .A indicates assembly language source unit and .P indicates a macro source unit.

[ctl_arg]

One or more control arguments chosen from the following list.

-COUT out_path

Indicates that listings which would normally be written to the file path.L are to be written to out_path.

{-SIZE nn / -SZ nn}

Specifies the number of 1024-word memory blocks that are to be used for symbol table building. If not specified, nn assumes the value 01.

FUNCTION DESCRIPTION:
The XREF command is used to invoke the GCOS 6/MDT Cross-Reference component. Execution of the Cross-Reference program is normally intended for the batch task group, as is the execution of most of the other components used in a program development activity.
The path parameter can assume any of the acceptable forms of a pathname, although normally it would be a simple name, indicating that the source unit resides in the working directory. The path

parameter must be given with either the .A or the .P suffix to indicate which type of source unit is to be cross-referenced.
If the -COUT control argument is not specified, the cross-reference listing is written to a file created by the Cross-Reference program, having a file name of the form path.L, the path portion being identical to the last (or only) element of the path parameter. If a file with this name already exists as a result of having performed an assembly of the same source unit, the listing generated by the Cross-Reference program is appended to that file. The file can subsequently be listed on a line printer by using the PRINT utility command. If a different file is specified by using the -COUT argument, out_path is the name of the file containing the listing. The .L suffix is not appended.
Additional information concerning the Cross-Reference program is contained in the Program Preparation and Checkout manual.

DELETE GROUP

Command Name: DG
Delete an online task group definition previously created by a CREATE GROUP command.

FORMAT:

DG id

PARAMETER DESCRIPTION:
id

The group identification of a task group previously created by a CG command specifying the same id.

FUNCTION DESCRIPTION:
The DELETE GROUP command removes all of the data structures which were constructed by the CG command issued previously with this id. No more ENTER GROUP REQUEST commands can be issued for this task group after the DG command has been executed. The DG command takes effect immediately if the task group is dormant when the command is issued. If it is active (i.e., if its code is being executed and/or there are still requests in this task group's request queue), the DG command takes effect when execution terminates and there are no more requests in the queue.
When a task group is deleted, the memory occupied by the data structures defining the group, and any memory associated with the execution of the group, is returned to the appropriate memory pool and is available for use by other task groups.
This command can be issued only from an online task group.

DISSOCIATE PATH

Command Name: DISSOC
Break the association between the indicated logical file number and the external file name, established by a previous ASSOCIATE PATH command.

FORMAT:

DISSOC lfn

PARAMETER DESCRIPTION:
lfn

The logical file number whose association with an external file name is to be broken.

FUNCTION DESCRIPTION:
The DISSOCIATE PATH command is used when a task has no further need for the association between the specified logical file number (LFN) and the related external file name. It frees the LFN so that another task, which requires the same LFN to be associated with a different external file, can have this relation made by the use of another ASSOC command.

The DISSOC command has no effect on a file which is open at the time the command is issued. For example, if the DISSOC command follows an ETR command that did not specify the -WAIT control argument, and the task is still executing and using the file to be dissociated, the dissociation does not occur.

Example:

```
DISSOC 12
```

The external file associated with LFN 12 is no longer related to the LFN, and cannot be referred to by the task through the LFN.

DUMP EDIT

Command Name: DPEDIT
Transfer to a printer the contents of a previously written memory dump file.

FORMAT:

```
DPEDIT path [ctl_arg]
```

PARAMETER DESCRIPTION:

path

The pathname of the file containing the memory dump to be edited. The last element of the pathname must be the name DUMPFILE.

[ctl_arg]

One or more control arguments chosen from the following list.

{-FROM X'hhhh' / -FM X'hhhh'}

Specifies that the physical portion of the dump is to begin with location $hhhh_{16}$. If not specified, the dump begins at location zero.

-TO X'hhhh'

Specifies that the physical portion of the dump is to end at location $hhhh_{16}$. If not specified, the dump terminates when the end of DUMPFILE is reached.

{-NO_LOGICAL / -NL}

Indicates that the logical dump of the system control structures is to be omitted.

{-NO_PHYSICAL / -NP}

Indicates that the physical portion of the memory dump is to be omitted.

{-MEMORY / -MEM}

Specifies that current memory, rather than the memory image contained in DUMPFILE, is to be printed. If the -FROM and -TO arguments are not specified with the -MEM argument, the dump is from location zero to the physical end of memory.

FUNCTION DESCRIPTION:
The DUMP EDIT command causes the transfer to a hard-copy device of the contents of the memory dump file. This file, DUMPFILE, is allocated at the time a volume is created with a memory dump bootstrap record, and comprises sufficient sectors to contain the number of 4096-word memory modules specified when the volume was created. Refer to example 2 in the description of the CREATE VOLUME command.

The transfer of data from memory to the memory dump file must have been previously performed by a bootstrap operation which specified the channel to which the device containing the memory dump volume was attached.
The pathname specified by the path parameter is normally a full pathname of the form

^ vol_id>DUMPFILE

Thus, in the case of the dump volume created in the example referred to above, the form of the pathname would be

^ DMPVOL>DUMPFILE

The memory dump output produced by the DPEDIT command comprises a logical portion and a physical portion. The former consists of an edited printout of system control structures such as task control blocks, dedicated memory locations, and group control blocks. The physical portion consists of a memory image printout encompassing the memory locations explicitly or implicitly specified by the -FROM and -TO arguments or their defaults. This portion of the dump is printed in both hexadecimal and ASCII representation, with duplicate line suppression. Any hexadecimal digit pairs which have no printable ASCII equivalents are represented by ASCII period (.) characters.
Additional information concerning the Dump Edit utility program is contained in the *Program Preparation and Checkout* manual.

Example 1:

```
DPEDIT ^DUMPER>DUMPFILE -FROM X'0400' -NL
```

Print the contents of the memory dump file located on the volume DUMPER. Memory locations between 0400_{16} and the end of the dump file are printed. The logical portion of the dump is to be omitted. The memory dump is printed on whatever device is currently serving as the user output device (the user terminal if no FILE OUT commands have been previously issued).

Example 2:

```
FO >SPD>LPT01
DPEDIT ^DUMPER>DUMPFILE
```

Print the contents of the same dump file as in example 1 above. In this case, because of the FO command which precedes the DPEDIT command, the output is printed on a printer designated as LPT01. Both logical and physical portions of the dump are printed, and the dump encompasses locations zero through the end of the dump file.

EDITOR

Command Name: ED
Add, delete, or modify selected lines of a source unit file.

FORMAT:

ED[ctl_arg]

PARAMETER DESCRIPTION:
[ctl_arg]
One or more control arguments chosen from the following list.

-IN path
Specifies the file from which editor directives are to be read. If not specified, directives are obtained from the current user input file.

{ -LINE_LEN n }
{ -LL n }

Specifies the maximum line length to be acted upon by the Editor. If not specified, n assumes the value of 80.

FUNCTION DESCRIPTION:
The EDITOR command is used to invoke the GCOS 6 Text Editor component. Execution of the Editor is normally intended for the batch task group, as is the execution of most of the other components used in a program development activity.
A full description of the operation and use of the Editor is contained in the Program Preparation and Checkout manual.

ENTER BATCH REQUEST

Command Name: EBR
Enter a request for Execution Command Language (ECL) processor in the batch request queue.

FORMAT:

EBR user_id in_path[ctl_arg]

PARAMETER DESCRIPTION:
user_id

A field comprising two subfields in the form person.project, by which this request is identified. The user id subfields are also used by the system in the construction of absolute pathnames.

in_path

The name of the file from which the ECL processor is to read its commands.

[ctl_arg]

Only one control argument is recognized, and is described below.

-OUT out_path

Defines the path name of the file which is to receive user output and error output from the batch task group. If not specified, one of the following assumptions is made:
If in_path specifies a mass storage file, out path = in_path.AO
If in_path specifies an interactive terminal, out_path = in_path
If in_path specifies an input-only device, out_path is null.

-WD path

Specifies that path is to be used as the working directory pathname instead of the pathname established by the user_id parameter.

FUNCTION DESCRIPTION:
The ENTER BATCH REQUEST command initiates the execution of the ECL processor as the lead task in the batch task troup previously created by an OCL CREATE BATCH command. If the task group is dormant at the time the EBR command is issued, execution begins immediately. Otherwise, the request is queued for execution when the group becomes dormant; i.e., a previous EBR command has activated the task group and its code is still executing. The ECL processor obtains its commands from the file named in the in_path parameter. This means that the file must begin with an ECL command, although it may contain other items which the called ECL function may require for its execution (e.g., Editor directives).

Example:

EBR BROWN.LIBRARY CMND_IN

The batch task group is to be activated by a request identified as BROWN.LIBRARY. It will receive its input from and direct its output to files identified as CMND_IN and CMND_IN.AO, respectively.

ENTER GROUP REQUEST

Command Name: EGR
Activate the lead task of an online task group previously created by a CREATE GROUP command.

FORMAT:

EGR id user_id[in_path] [ctl_arg]

PARAMETER DESCRIPTION:

id

The group identification of a task group previously created by a CG command specifying the same id.

user_id

A field comprising two subfields in the form person.project, by which this request is identified. The user id subfields are also used by the system in the construction of absolute pathnames.

[in_path]

The name of the file from which commands and user input are to be read by the task group during execution. This argument is set to null if it is not specified. It is required if the CG command specified the control argument -ECL.

[ctl_arg]

One or more control arguments chosen from the following list.

-OUT out_path

Defines the path name of the file which is to receive user output and error output from the task group. If not specified, one of the following assumptions is made:
If in_path specifies a mass storage file, out_path = in_path.AO
If in path specifies an interactive terminal, out_path = in_path
If in_path is not specified, out_path is null
If in_path specifies an input-only device, out_path is null.

-WD path

Specifies that path is to be used as the working directory pathname instead of the pathname established by the user_id parameter.

-ARG

Indicates that additional arguments required by the task group during execution follow. These additional arguments are passed to the lead task to be used as necessary. If used, the -ARG control argument must appear last.

FUNCTION DESCRIPTION:

The ENTER GROUP REQUEST command initiates the execution of the lead task of a task group previously created by a CREATE GROUP command. If the task group is dormant at the time the EGR command is issued, execution begins immediately. Otherwise, the request is queued for execution when the group becomes dormant; i.e., a previous EGR command has activated this task group and its code is still executing. Execution begins at the point specified by the -EFN control argument of the CG command, if specified. If the -EFN control argument was not used (i.e., the lead

task is the ECL processor), execution begins by reading the file named by the in path parameter. This means that this file must begin with an ECL command, although it may contain other items which the called ECL function may require for its execution.

Example:

EGR AX SMITH.SERVICES MPG_DATA -ARG '07/12/76 1100AM'

The task group identified as AX in a previous CG command is to be activated. This request is identified as SMITH.SERVICES. The task group expects its input data to come from a file named MPG_DATA, in the working directory, and will write its output to a file named MPG_DATA.AO, also in the working directory. The lead task expects one argument, a date and time item. The item is enclosed in quotes because there is an embedded space, but it is to be interpreted as a single argument.

ENTER TASK REQUEST

Command Name: ETR
Allocate and initialize a task request block and place it on the request queue of the indicated task.

FORMAT:

ETR lrn [ctl_arg]

PARAMETER DESCRIPTION:

lrn

A logical resource number specified in a previous CREATE TASK command.

[ctl_arg]

One or more control arguments chosen from the following list.

-WAIT

Specifies that the ECL processor is to wait upon completion of the requested task before resuming execution.

-ARG

Indicates that additional arguments required by the requested task follow. These additional arguments are passed to the requested task in an extension of the task request block. If this control argument is used, it must be the last one specified.

FUNCTION DESCRIPTION:

The ENTER TASK REQUEST command is used to activate a task which was previously defined by a CREATE TASK command specifying the same logical resource number (LRN) as that named in this command.

The ETR command causes the construction of a GCOS 6 standard task request block (TRB), which consists of the elements described in Section 4 of the Monitor and I/O Services Macro Calls manual. Additional entries to accommodate task-specific parameters specified by the -ARG control argument are appended to the TRB as required.

Multiple tasks can be made to execute concurrently within a given task group by issuing multiple CT and ETR commands.

Tasks can also be made to execute serially; i.e., one task going to completion before a subsequent task begins execution. The -WAIT control argument is the mechanism that controls concurrency of execution. Judicious use of this argument can also result in a mixture of concurrent and serial execution (see example 3, below).

When all of the tasks created and requested have terminated, the structures constructed by the CT and ETR commands can be removed by issuing a BYE command. This removes all structures except those of the lead task (the ECL processor), after which the next group request if any, can be honored.

In each of the following examples three tasks are assumed to have been previously created by the CT commands shown in the example in the description of the CT command, namely:

```
CT 10 02 -EFN PROG10
CT 11 03 -EFN PROG11
CT 12 02 -SHARE 10 ENTRY2
```

Any other prerequisite commands (e.g., file creation, association of LFNs to pathnames) are also assumed to have been issued.

Example 1:
The three tasks are such that there are no dependencies among them, so they can be run concurrently. The following ETR commands are issued to activate them.

```
ETR 10
ETR 11
ETR 12
```

Example 2:
The three tasks are required to be executed in a particular sequence, determined by the order in which the ETR commands are issued. The following ETR commands are used to activate them.

```
ETR 10 -WAIT
ETR 12 -WAIT
ETR 11
```

In this case, execution of task 12 must await completion of task 10, and task 11 must likewise await completion of task 12. Since task 11 does not specify -WAIT, another (unrelated) activity can be initiated in parallel with the execution of task 11. Note, however, that if a BYE command follows the last ETR command, task 11 will probably not complete, since the BYE command takes effect even if a task within the task group is active.

Example 3:
Two of the tasks have a dependency between them and the third is independent of the other two. The following sequence of ETR commands can be used to activate them.

```
ETR 11
ETR 10 -WAIT
ETR 12
```

In this case, because task 11 does not specify -WAIT, both task 11 and task 10 are activated to run concurrently, but task 12 is dependent upon the completion of task 10. As in the previous example, another activity can be initiated concurrently with the execution of the third task.

EXECUTION COMMAND

Command Name: EC

Invoke the Execution Command (EC) processor to read ECL commands from a designated file.

FORMAT:

```
EC path
```

PARAMETER DESCRIPTION:
path

The name of a file, path.EC, containing ECL commands and EC directives.

FUNCTION DESCRIPTION:
The function of the EC processor is to read from a previously created file a series of ECL commands and EC directives. It provides a mechanism whereby a sequence of routinely performed ECL functions can be executed without the need for manually entering the functions through an interactive terminal.
The file path.EC is a file which has been previously created by use of the Editor. It contains one or more ECL commands and EC directives which are interpreted in sequence by the EC processor and acted upon as described in the following paragraphs.
When an ECL command is encountered by the EC processor, it is simply passed to the ECL processor for interpretation and execution. This means that the syntax of the command as read from the file path.EC must be identical to that which would have been entered from a terminal device if the function were requested manually. All parameters and control arguments must be supplied as specified in the individual ECL command descriptions.
When a command execution terminates, control is returned to the EC processor, which then reads the next line from the file.
The EC processor also recognizes several directives which are not passed to the ECL command processor, but are interpreted and acted upon by the EC processor itself. These directive lines are identified by a character string beginning with & and followed by a Δ (space or tab character). They provide control over certain operational aspects of the EC processor as well as a degree of control over the logic of execution of the series of ECL commands. Any directive other than those described below is treated as an &QΔ directive, except that a nonzero error status code is returned to the task which invoked the EC command.

The EC control directives are described in detail in the following paragraphs.

&Δ

This signifies a comment line and is not processed further. It is visible only by obtaining a listing of the EC file, and can be used, for example, to describe the function performed by the commands contained in the file.

&AΔ This directive signifies that the EC file is to be substituted for the user input stream defined by the in path parameter when the EGR or EBR command activating this task group execution was issued. This means that whenever the executing task refers to its user input file, the data which would normally be read from this file is obtained instead from the file being read by the EC processor.

&DΔ This directive restores the task group's user input file to that which was defined at task group activation.

&FΔ Command line printing is to be turned off; i.e., command lines are not to be written to the user out file. This is the default; command lines are not normally written to user out.

&NΔ Command line printing is to be turned on. Each command line read from the EC file is written to the user out file before being passed to the ECL command processor. The & directive lines are not written.

&PΔ The entire line, except for the &PΔ, is written to the user out file. Printing of &PΔ lines occurs regardless of whether command line printing is on or off.

&IFΔ This directive permits the interrogation by the EC processor of an error status code returned by the command executed immediately prior to the &IFΔ directive. For compatibility across all GCOS systems, it has the format:

&IFΔ[EQUALSΔ&STATUSΔ0]Δ&THENΔ&ELSEΔ&QUIT;

and is interpreted as follows:
If the error status code returned from the execution of the immediately preceding command line is zero (0) continue with the next command or directive line; otherwise quit.
The &IFΔ directive enables the EC processor to exit from any further processing of the EC file when an error condition resulting from the interpretation or execution of a command would make meaningless the execution of any subsequent commands.

&QΔ The execution of the current EC file is terminated, and control is returned to the invoking task. Implicit &QΔ directives may be executed, as described above, by invalid & directives or because of error status codes returned by the interpreation or execution of a command line. To ensure proper termination of the EC command, every EC file should have the &QΔ directive as its last line.

Example 1:
A user is developing a program named TEST. Several recursions of source unit correction, assembly, and link are required before the program is operational. The original source unit TEST.A has already been created using the Editor. An EC file PROG_DEV.EC has also been previously created, and contains the following commands and EC directives.

```
&ΔEDIT, ASSEMBLE, AND LINK PROGRAM 'TEST'
&PΔBEGIN EDITOR
ED
&IFΔ[EQUALSΔ&STATUSΔ0]Δ&THENΔ&ELSEΔ&QUIT;
&PΔBEGIN ASSEMBLY
ASSEM TEST -COUT >SPD>LPT01
&IFΔ[EQUALSΔ&STATUSΔ0]Δ&THENΔ&ELSEΔ&QUIT;
&PΔBEGIN LINK
LINKER TEST -COUT >SPD>LPT01
&IFΔ[EQUALSΔ&STATUSΔ0]Δ&THENΔ&ELSEΔ&QUIT;
&PΔLINK COMPLETE
&QΔ
```

In order to execute the correction, assembly and link sequence the user has only to enter the ECL command

EC PROG_DEV

The EC processor appends the .EC suffix to PROG_DEV and searches the current working directory for the resulting file name PROG_DEV.EC. Each command is preceded by an &PΔ directive which causes a typeout to the user_out file informing the user of his step-by-step progress through the sequence. The &IFΔ directives following each command line would cause an exit from the sequence if execution of the command resulted in an error reported by a non-zero error status code. No further &PΔ messages would be written to user_out, indicating to the user that some error condition has been detected.

Example 2:
Execution of the program 'TEST' created in example 1 above requires the creation of a work file for use by the program. This file is also to be deleted after the program is finished its execution. The following EC file, called EX_TEST.EC has been created to perform this sequence of functions:

```
&△EXECUTE PROGRAM 'TEST'
&P△CREATE WORK FILE 'TEST01'
CF TEST01 -SEQ -SZ 100
&P△EXECUTE 'TEST'
TEST -ARG arg1 ... argn
FD TEST01
RL TEST01
&P△EXECUTION OF 'TEST' COMPLETE
&Q△
```

In this case, the user enters the ECL command

EC EX_TEST

which, in the same manner as in example 1, invokes the EC processor and turns control over to the sequence of commands and directives contained in the EC file. The work file TEST01 is created, the program 'TEST' is invoked, supplying any arguments (arg_1 ...arg_n) which may be necessary to its execution, a dump of the work file is requested, and the file is then released (deleted).

EXPORT PAM FILE

Command Name: EX_PAM
Unload one or more MDT sequential files to a BES1 and BES2 partitioned file.

FORMAT:
EX_PAM path pam [mem_1] ... [-R]

PARAMETER DESCRIPTION:
path
The pathname of a directory containing one or more files to be exported.

pam
The name of the BES1 and BES2 partitioned file to which the sequential files are to be exported.

[mem_i]
The simple names of one or more files, immediately contained within path, to be copied to the partitioned file as members. If not specified, all files within path are copied.

[-R]
Indicates that if any member named by the mem_i parameter already exists in the file named by pam, it is to be replaced.

FUNCTION DESCRIPTION:
The EX_PAM command permits the transfer of sequential files contained within the MDT file system to BES1 and BES2 partitioned access method (PAM) files. Each sequential GCOS 6 file, when copied to the PAM file, becomes a member in the PAM file, and has its name entered into the member index portion of the PAM file.
The path parameter names the GCOS 6 file system directory which immediately contains the files to be transferred; i.e., if the named directory contains subdirectories which themselves contain files,

these latter files are not affected by the EX_PAM command. If no mem$_i$ parameters are given, each file which is immediately subordinate to the directory named in the path parameter is transferred.
The pam parameter names the BES1 and BES2 partitioned file which is to contain the transferred files. This file must have been previously created as a BES 1/2 offline activity. Three steps are involved in the creation of this file.

1. The volume which is to contain the PAM file must be initialized using the Initialize function of Utility Set 1. (If the volume has already been initialized and the file is to be added to the volume, this step is omitted.)
2. The file must be allocated using the Allocate function of Utility Set 1. Sufficient sectors must be allocated to contain the expected files and the accompanying member index.
3. The file allocated in step 2 must be initialized as a partitioned file using the Initialize function of Utility Set 1. The number of members specified must be large enough to accommodate the expected number of transferred files.

The user can refer to the BES2 Utility Programs manual (Order Number AU47) for full details regarding the allocation and initialization of partitioned files.
Because of the eight-character limit in the length of a BES1 and BES2 partitioned file member name, the first eight characters of each GCOS 6 file to be transferred must be unique. File names longer than eight characters are truncated to eight characters.

Example:

EX_PAM MYDIR MYPAM FILEA FILEB FILEC

Three files contained in the directory MYDIR are to be transferred to the partitioned file MYPAM. The files, FILEA, FILEB, and FILEC, are GCOS 6 sequential files and are added as members to the previously created PAM file as new members.

FILE DUMP

Command Name: FD
Transfer the contents of the specified area of a mass storage volume to the user output file.

FORMAT:

FD path[ctl_arg]

PARAMETER DESCRIPTION:

path

The pathname of the file or volume whose contents are to be dumped.

[ctl_arg]

One or more control arguments chosen from the following list:

{-FROM xx / -FM xx}

The first xx records are to be skipped before beginning the dump. xx can be a decimal number or a hexadecimal number in the form X'hhhh', where 'hhhh' represents four hexadecimal digits.

{-LIMIT nn / -LI nn}

Dump the number of records specified by nn. If end of file is encountered before nn records are dumped, the dump terminates at end of file. nn can be a decimal number or a hexadecimal number in the form X'hhhh', where 'hhhh' represents four hexadecimal digits.

-CI

The dump is to be taken at the control interval level. If the path parameter specifies a magnetic tape file or volume, the dump is taken at the physical block level.

-NWD

A magnetic tape volume is not to be rewound before opening it for the dump. This argument is valid only if -CI is also specified.

-BACK nn

A magnetic tape volume is to be backspaced nn blocks before dumping. This argument is valid only if -CI is also specified.

-HEX

Print only the hexadecimal representation of the file or volume content.

-ALPHA

Print only the ASCII representation of the file or volume content.

FUNCTION DESCRIPTION:
The FILE DUMP command permits the user to obtain a printed listing of a mass storage or magnetic tape file or volume. Whether a file or a volume is to be dumped is determined by the form of the path parameter. A mass storage file dump can be obtained by using any of the acceptable forms of a pathname. A mass storage volume dump or a dump of a magnetic tape file or volume is obtained by specifying the pathname in the form

>SPD>dev_name[>vol_id]

The dev_name portion of the path parameter is the symbolic device name assigned by a CLM DEVICE directive at system configuration. The vol_id portion, if used, specifies that the volume name is to be verified before initiating the dump.
The output from the FD command is written to whatever file or device is currently assigned as the user output file.

Example 1:

FD MYFILE

A user file named MYFILE is to be dumped in its entirety. The output is written in both hexadecimal and ASCII representation.

Example 2:

FD >SPD>MTU01 -NWD -BACK 5 -LIMIT 10 -CI

A portion of a magnetic tape volume is to be dumped. The volume is that defined at system configuration as MTU01. Ten physical blocks are to be dumped after backspacing the tape five blocks. The -NWD argument is used to prevent rewinding of the tape, and, in conjunction with the -BACK and -LIMIT arguments, effectively causes a dump of five blocks on either side of the tape's current position.

FILE OUT

Command Name: FO
Change the destination to which user output is sent.

FORMAT:

FO[path]

PARAMETER DESCRIPTION:
[path]

The name of the new user output file. If this parameter is omitted, the user output file reverts to that established at task group initiation.

FUNCTION DESCRIPTION:
The FILE OUT command defines a new device or file to which user output generated by a task is written. When a task group is initiated, the file which is to receive this output is established by the -OUT control argument of the ENTER GROUP REQUEST command. Error output is also written to the same file. The FO command makes it possible for a series of group requests to write their output information to separate files or devices. It does not affect the destination of error output; this is always written to the originally defined file. The use of the FO command with no argument resets the destination of user output to that of error output as defined in the EGR or EBR command.

Example:

FO REPORT_OUT

The output generated by the issuing task is to be redirected to a file named REPORT_OUT, in the working directory.

FORTRAN

Command Name: FORTRAN
Compile the FORTRAN source program unit represented by the indicated file name, applying the specified compilation options.

FORMAT:

FORTRAN path[ctl_arg]

PARAMETER DESCRIPTION:
path

Specifies the name of the file containing the source unit to be compiled.

[ctl_arg]

One or more control arguments chosen from the following list:

{-NO_OBJ / -NO}

Indicates that the generation of the object text is to be suppressed. If omitted, the object text unit is generated.

{-LIST_OBJ / -LO}

Indicates that a listing of the object text output is to be obtained. Object text is interspersed with source text in the listing. If omitted, object text is not listed.

{-LIST_ERRS / -LE}

Specifies that only those source lines containing compilation errors, together with their error codes, are to be listed. If omitted, and -NL is not specified, the complete source program is listed, followed by a listing of the error lines and codes.

{-NO_LIST / -NL}

Specifies that all listings are to be suppressed. If omitted, and -LE is not specified, the complete source unit is listed, followed by a listing of the error lines and codes.

{-SIZE nn}
{-SZ nn }

Designates the number of 1024-word blocks of memory to be used for tables by the compiler. The value of nn can be between 02 and 20, inclusive. If omitted, the available memory in the task group's memory pool, up to approximately 1700 words, is used. At least 1024 words must be available.

-AS

Indicates that the output from the compiler is a file of assembly language source statements which can be used as input to the Assembler. This file has a pathname of the form path.A, where path is the same as that specified in the FORTRAN command.

-HS

Specifies that the source unit is in Hollerith code created by an 026 keypunch, a Type H200/2000, or a Type H716 central processor. If omitted, standard MDT code is assumed.

-SI

Indicates that the compiler is to generate short form integers and logical variables, each being one word. If not specified, two-word integers and variables are generated.

-UC

Specifies that the generation of embedded links to subroutines referred to by CALL statements is to be suppressed. If not specified, links are generated.

-UZ

Specifies that the generation of embedded links to system subroutines (those beginning with the letters ZF) are to be suppressed. If not specified, these links are generated.

-WRK n

Designates the size of object time work space for FORTRAN main program. The value of n can be from 1 to 9999_{10} inclusive. If not specified, a value of 325_{10} is assumed.

-COUT out_path

Indicates that the listings which would normally be written to the file path.L are to be written to out_path.

FUNCTION DESCRIPTION:

The FORTRAN command is used to invoke to GCOS 6 FORTRAN Compiler component. Execution of the compiler is normally intended for the batch task group, as is the execution of most of the other components used in a program development activity.

The path parameter can assume any of the acceptable forms of a pathname, although normally it would be a simple name, indicating that a source program unit which resides in the working directory is to be compiled. Wherever it exists, it must be suffixed with a .F suffix, indicating that it is a FORTRAN language source unit. The path parameter must be given *without* the .F suffix; the compiler appends the suffix prior to searching the directory for the source unit.

If the -COUT control argument is not specified, the source and object listings (if specified) are written to a file created by the compiler in the working directory, having a file name of the form path.L, the path portion being the last (or only) element specified in the path parameter. This file can be subsequently listed on a line printer by using the PRINT utility command. If a different file is specified by using the -COUT argument, the listings are written to the file whose pathname is out_path. The compiler does *not* append a .L suffix.

The object text unit generated by the compiler is written to a file whose name is of the form path.O, and is contained in the working directory.

If files of the form path.L and path.O already exist in the working directory, they are overlaid by the output generated by the current compilation.

Example:

```
FORTRAN FTPROG -LIST_OBJ -SI -COUT >SPD>LPT01
```

A FORTRAN source program, FTPROG.F, residing in the working directory, is to be compiled. The source and error listings are to be written to the printer LPT01, and the object text unit is to be written to the file FTPROG.O in the working directory. Short-form integer and logical variables are to be generated.

IMPORT PAM FILE

Command Name: IM_PAM
Transfer one or more BES1 and BES2 partitioned file members to the GCOS 6 file system.

FORMAT:

```
IM_PAM pam path [mem_i]...[-R]
```

PARAMETER DESCRIPTION:

pam

The name of the BES1 and BES2 partitioned file from which members are to be transferred.

path

The name of a GCOS 6 directory into which the BES1 and BES2 file members are to be transferred.

[mem$_i$]

The names of one or more partitioned file members which are to be transferred to the GCOS 6 file system.

[-R]

Indicates that if a file named by the mem$_i$ parameter already exists in the directory named by path, it is to be replaced.

FUNCTION DESCRIPTION:
The IM_PAM command permits the transfer of one or more BES1 and BES2 partitioned access (PAM) file members into the GCOS 6 file system. A PAM file member, when transferred, becomes a GCOS 6 variable sequential file contained within the directory specified by the path parameter.
The pam parameter names the PAM file which contains the members to be transferred. This file has been previously created using BES1 and BES2 offline procedures.
The path parameter names the GCOS 6 file system directory which is to immediately contain the files transferred to it. It must have been previously created through the use of the ECL Create Directory command.
Each of the mem$_i$ parameters, if any are specified, names a member of the PAM file, specified by the pam parameter, which is to be transferred to the GCOS 6 file system. If no mem$_i$ parameters are specified, every member contained in the file is transferred and becomes a GCOS 6 sequential file.
If the -R control argument is not specified, and if a GCOS 6 file whose name is specified by a mem$_i$ parameter already exists in the file system, an error message is issued.

Example:

```
IM_PAM MYPAM MYDIR MEM01 MEM02 MEM03 -R
```

Three PAM file members, MEM01, MEM02, and MEM03, contained in the partitioned file MYPAM, are to be transferred into the GCOS 6 file system directory MYDIR. If the file system already contains any files whose names are the same as those of the specified members, they are to be replaced.

LINKER

Command Name: LINKER
Create a bound unit from one or more object text units, applying the specified options.

FORMAT:

LINKER [name] [ctl_arg]

PARAMETER DESCRIPTION:

[name]

Pathname of the created bound unit file. The pathname can be a simple, relative or absolute name. If the specified file already exists, it is overlaid with the new bound unit. If not specified, no bound unit is created, only a list file is created in the current working directory.

[ctl_arg]

One or more control arguments chosen from the following list:

-IN path

Specifies the pathname of the file containing the linker directives. The file can be read from a disk device, a card reader, or a terminal device. If not specified, the file named in the in_path parameter of this task group's EGR or EBR command is used.

-COUT out_path

Specifies the name of the file to which the map listing will be written. It can be written to a disk device, a printer, or a terminal device. If not specified, the listing is written to the file name.M in the working directory, overlaying any existing file by that name.

{-SAF / -LAF}

Indicates the addressing mode in which the bound unit is to execute; -SAF indicates short (one-word) address form, -LAF indicates long (two-word) address form. If not specified, -SAF is assumed.

{-SIZE nn / -SZ nn}

Designates the number of 1024-word memory modules available for the Linker symbol table. The minimum value of nn must be 01. If not specified, the available memory in the task group's memory pool is used.

FUNCTION DESCRIPTION:
A complete description of the function of the Linker, its controlling directives, and examples of use are contained in the Program Preparation and Checkout manual.

LIST NAMES

Command Name: LS
Display the names of one or more elements contained in the specified directory, along with their types, attributes, and sizes.

FORMAT:

LS [ctl_arg] [entry_name]

PARAMETER DESCRIPTION:

[entry_name]

Specifies the name of the entry to be displayed. If this parameter is omitted, all entries of the type(s) specified by control arguments are displayed.

[ctl_arg]

One or more control arguments chosen from the following list:

-PN path

Specifies the directory from which entries are to be listed. If this argument is omitted, entries contained in the working directory are listed.

-FILE

Indicates that only file entries are to be displayed. If none of the control arguments -FILE, -DIR, or -ALL are specified, -FILE is the default.

-DIR

Indicates that only directory entries (i.e., directories subordinate to the specified directory path) are to be displayed.

-ALL

Indicates that both file and directory entries subordinate to the specified directory are to be displayed.

{-DETAIL
-DTL }

Specifies that the file type and attributes of each entry are to be displayed.

FUNCTION DESCRIPTION:

The LIST NAMES command permits the user to obtain a listing of the file and/or directory entries contained within a given directory. It also displays various attributes of file entries: file type, starting sector, number of sectors, and record length.

The following list gives the possible file type designators and their meanings.

Type	*Meaning*
R1	BES fixed relative, static allocation, no deletable records.
R2	BES fixed relative, dynamic allocation, no deletable records.
R4	BES fixed relative, static allocation, deletable records.
R5	BES fixed relative, dynamic alloction, deletable records.
D	Directory.
S	Variable sequential.
R	Relative.
ID	Indexed (data area).
I	Indexed (index area).
RD	Random.
*	Organization not recognized.

Example:

```
LS -PN ^Z00B00 START_UP.EC
```

List the attributes of the file START_UP.EC, contained in the directory Z00B00. The following display is returned to the user output file.

```
DIRECTORY:  ^Z00B00

                              STARTING  NUMBER OF  RECORD
  ENTRY NAME   TYPE           SECTOR    SECTORS    LENGTH
**************************************************************

  START_UP.EC    S             35A          10        100
**************************************************************
```

LIST SEARCH RULES

Command Name: LSR
Display the search rules currently defined for the issuing task group.

FORMAT:

LSR

PARAMETER DESCRIPTION:
No parameters are required or permitted with this command.

FUNCTION DESCRIPTION:
The LIST SEARCH RULES command writes to the user output file the full pathnames of the directories used by the system task group in its search for bound units.
The search rules define three directory pathnames and the sequence in which they are used during a search. The first of these is the system task group's working directory. The second is the system directory LIB1, whose pathname is >SYSLIB1. The third is the system directory LIB2, whose pathname is also >SYSLIB1. The pathnames associated with LIB1 and LIB2 can be changed through the use of the CHANGE SYSTEM DIRECTORY command. The pathnames returned by the LSR command always reflect the current directory pathnames.

Example:

Assume that the system task group's initial working directory is ^SYSVOL and that no CWD or CSD commands have been issued. The LSR command returns

```
^SYSVOL
^SYSVOL>SYSLIB1
^SYSVOL>SYSLIB1
```

Assume now that a CSD NEW_DIR -LIB2 command has been executed at some point prior to the issuing of the LSR command. The LSR command now returns

```
^SYSVOL
^SYSVOL>SYSLIB1
^SYSVOL>NEW_DIR
```

LIST WORKING DIRECTORY

Command Name: LWD
List the full pathname of the current working directory.

FORMAT:

LWD

PARAMETER DESCRIPTION:
No parameters are required or permitted with this command.

FUNCTION DESCRIPTION:
The LIST WORKING DIRECTORY command is used to obtain the full pathname of the working directory currently being used by the issuing task group. It is at times useful to be able to determine

the identity of the working directory after having made several changes of working directories through the use of CHANGE WORKING DIRECTORY commands. The LWD command causes the full pathname of the working directory to be written to the user output file in the form

^ vol_id>dir_1 ...

The ellipsis indicates that one or more subordinate levels may be included in the pathname of the working directory, depending on the nature of previously-issued CWD commands. Also, again depending on previous CWD commands, the person and/or project entries may not be included in the path.

Example:
Assume that a task group's initial working directory is ^ SYSVOL>UDD>A1>JOE, as established at task group initiation.
A CWD EC_DIR command has been previously issued. The LWD command returns

^ SYSVOL>UDD>A1>JOE>EC_DIR

If, starting with this working directory, a CWD < command is issued, a subsequent LWD command would return

^ SYSVOL>UDD>A1>JOE

MACRO PREPROCESSOR

Command Name: MACROP
Expand assembly language macro calls and %INCLUDE statements into assembly language source statements, applying the indicated options.

FORMAT:

MACROP path [ctl_arg]

PARAMETER DESCRIPTION:

path

The pathname of the file which contains the unexpanded source statements. The file's .P suffix must not be included in the path parameter; it is appended by the macro preprocessor prior to searching for the source unit.

[ctl_arg]

One or more control arguments chosen from the following list:

{-INCLUDE_CONTROLS, -IC}

Indicates that the Macro Preprocessor is to incorporate as comment statements in the expanded source output all macro control statements and inline macro definitions. If this argument is omitted, these statements are not included.

{-MACRO_CALLS, -MC}

Indicates that the Macro Proprocessor is to incorporate as comment statements in the expanded source output all macro call statements. If this argument is not specified, these statements are omitted.

{-SIZE nn / -SZ nn}

Designates the maximum number of 1024-word blocks of memory that the Macro Preprocessor is to use for work space. If this argument is not specified, seven-eighths of the task group's memory pool, or available pool memory minus 400_{10} words, whichever is smaller, is used.

FUNCTION DESCRIPTION:
The MACROP command is used to invoke the GCOS 6 Macro Preprocessor component. Execution of the Macro Preprocessor is normally intended for the batch task group, as is the execution of most of the other components used in a program development activity.
A full description of the operation and use of the Macro Preprocessor is contained in the Assembly Language manual.

MESSAGE

Command Name: MSG
Send a message from a user command device to the operator terminal.

FORMAT:

MSG message

PARAMETER DESCRIPTION:
message

The message to be sent. If it contains embedded blanks, it must be enclosed in double quotes (") or apostrophes (').

FUNCTION DESCRIPTION:
The MSG command is used whenever it is necessary for a task group to convey some item of information or a request for operator action to the system operator. The source of the message is whatever file or device is designated as command input for the sending task group at the time the message is sent; the message is displayed to the operator on the operator terminal. The operator can respond to the task group to indicate that a requested action has been taken.

Example:

MSG "PLEASE ABORT BATCH REQUEST"

Send a message to the operator requesting an abort of the current batch request. The operator responds by entering an ABR OCL command, and can then inform the batch user that his request has been honored.

MODIFY EXTERNAL SWITCHES

Command Name: MSW
Modify selected external switches associated with the issuing task group.

FORMAT:

MSW ctl_arg

PARAMETER DESCRIPTION:
ctl_arg
One or more control arguments chosen from the following list:

-ON $S_i[S_i]$...

Set the external switch indicated by S_i ON. Each S_i is a hexadecimal digit from 0 throuth F.

-OFF Si[Si]...

Set the external switch indicated by S_i OFF. Each S_i is a hexadecimal digit from 0 through F.

-ALL v

Set all switches to the value v. The value v can be either ON or OFF.

FUNCTION DESCRIPTION:
The MODIFY EXTERNAL SWITCHES command enables the issuing task group to modify the external switches by which it can control its execution. An external switch can be thought of as a hardware switch on a control panel, which can be set on or off manually by an operator. There is a separate switch word associated with each task group created, giving each group the capability of addressing 16 switches. A user program can contain instructions or statements which interrogate the settings of one or more of these switches, and can use these settings to control the execution logic of the program.

Example:

```
MSW -ON 25 -OFF 7B
```

In the issuing task group, external switch numbers 2 and 5 are to be set ON, and external switch numbers 7 and B are to be set OFF.

MODIFY FILE

Command Name: MF
Modify the attributes of the specified file.

FORMAT:

```
MF path ctl_arg
```

PARAMETER DESCRIPTION:
path

The pathname of a file whose attributes are to be changed.

ctl_arg

One or more control arguments chosen from the following list. At least one is required as there are no defaults.

{-SHARE / -SHR}

Specifies that the named file is to be made accessible to the batch task group.

{-NONSHARE / -NS}

Specifies that the named file is to be made inaccessible to the batch task group.

{-READ / -RD}

Specifies that no users are given permission to write to the named file; only reading is permitted.

{-WRITE / -WR}

Specifies that users are permitted access to the named file in the output, update, or extend mode.

NOTE: The arguments within the argument pairs -SHARE and -NONSHARE, and -READ and -WRITE, are mutually exclusive.

The MODIFY FILE command allows the accessibility and permission attributes of a file to be modified. When a file is first created (refer to the CREATE FILE command in this section), it is accessible to both online and batch task groups. It can also be read from and written to by any task. Its initial attributes are thus SHARE/WRITE.
If a file is made inaccessible to the batch task group (through the use of the -NS control argument), no access of any kind by the batch task group is permitted. Furthermore, directories can be given the -NS attribute; in this case the directory and all subdirectories and files contained within it are inaccessible to the batch task group.
Another kind of protection can be given a file by the use of the -RD control argument. This argument makes the file a read-only file, preventing any task groups, online or batch, from writing to the file. It can still be read by online tasks and, unless the -NS argument has also been specified, by batch tasks as well.
Attributes assigned to nondisk files by this command remain in effect only for the current initialization of the sytem. If the system is reinitialized, attributes for these files revert to SHARE/WRITE.
The MF command can be issued only from an online task group.

Example:

```
MF >UDD>PROJ1>USERA>FILE01 -NS
```

A file is to be made inaccessible to the batch task group. It remains accessible for writing by online tasks if it was previously accessible.

PATCH

Command Name: PATCH
Patch an object or image text file.

FORMAT:

```
PATCH path [ctl_arg]
```

PARAMETER DESCRIPTION:

path

The pathname of the object or image text file to be patched. An object text pathname must end with the .O suffix; no suffix is used for an image text file unless one was assigned when the image text unit was linked.

[ctl_arg]

Only one control argument is recognized, and is described below.

-IN path

The pathname of the file which contains the patch directives. If not specified, the directives are read from the current user input file.

FUNCTION DESCRIPTION:
The PATCH command permits the selective modification of an object or image (bound unit) text file, in accordance with directives submitted to the PATCH processor. A complete description of the directives and operation of the PATCH command, as well as several examples of its use, appear in the Program Preparation and Checkout manual.

PRINT

Command Name: PR
Print the contents of the indicated file.

FORMAT:

PR path [ctl_arg]

PARAMETER DESCRIPTION:

path

The pathname of the file whose contents are to be printed.

[ctl arg]

One or more control arguments chosen from the following list:

{-LIMIT nn / -LI nn}

Specifies the number of records to be printed if end of file is not encountered before the value of nn is satisfied. If not specified, all records in the file are printed.

{-COPIES n / -CP n}

Specifies the number of copies to be printed; i.e., the number of times the file is to be printed for this invocation. Default is 1.

{-SPACE n / -SP n}

This argument indicates that the file is not a true print file with print control characters in its records. Each record is printed on one or more print lines. The value of n specifies the line spacing between records, and can be either 1 or 2. 1 specifies single spacing (no blank line). 2 specifies double spacing (one blank line). The default value is 1.

{-FORTRAN / -FT}

The print file was created by a FORTRAN object program and has print control characters of the FORTRAN type.

{-FROM nn / -FM nn}

Indicates that the first nn records of the file are to be skipped before beginning to print. If not specified, printing starts at beginning of file.

{-LINE LEN nn / -LL nn}

Specifies the number of characters to be printed per line. If a longer line is read from the file, it is folded at the indicated print position. If not specified, the value of nn is 68.

-RL

Specifies that, at the completion of printing, the file is to be released.

FUNCTION DESCRIPTION:

The PRINT command is used to write to the current user output file the contents of a file formatted according to MDT print file conventions. Such a file contains, for each record to be printed, a printer forms control byte. This byte is in the first character position of each record, and serves to control the line spacing associated with each print line, as well as head-of-form spacing.
Print files written by the various language processors are suffixed with a .L unless otherwise directed

by the processor's -COUT control argument. This suffix must be included in the pathname specified by the path parameter when the PRINT command is used to print these types of files. These files always contain forms control bytes. User programs which write files destined to be printed using this command are responsible for supplying the appropriate forms control bytes in their output records. Files written by user programs are not required to be terminated with the .L suffix.
Print files written by FORTRAN object programs utilize a special set of forms control bytes. If the PRINT command includes the -FT control argument, these bytes are translated into equivalent standard forms control actions before the line is printed. If the -FT control argument is not specified for these files, resulting form spacing will probably not reflect that which was intended by the programmer.
Theoretically, any file can be printed by using the PRINT command. However, since the first byte of each record is interpreted as a forms control indicator, the line spacing which results from the printing of a nonprint file is unspecified. The -SP control argument, in addition to specifying the spacing between records, also negates the interpretation of the first byte as a control byte. Each record is printed on as many single-spaced lines as are required and the line spacing between records then occurs as specified by the -SP argument.
The user can request the printing of only a part of a file by the appropriate combination of -FM and -LI control arguments, which define, respectively, the point in the file at which printing is to begin and the number of lines to be printed.
When the output of the PRINT command is directed to a high-speed printer, the use of the -LL control argument specifying the physical line length of the printer is recommended, since the length of an output record whose ultimate destination is such a device is likely to be longer than the default 68 characters. If the argument is not specified, each such line will be folded at the 68th character.

Example 1:

```
PR TABLIST -CP 2 -FT
```

Two copies of the print file written by a FORTRAN program are to be printed. The file contains print lines less than 68 characters long; hence, the -LL argument is not required. The printed output is written on whatever device is currently associated with the user output file.

Example 2:

```
PR COBPRINT -LL 132
```

The print file from a program which writes 132-character print records is to be printed. If the current user output device is not a line printer, the PR command can be preceded by an FO (FILE OUT) command naming a line printer (LPTnn) as the output device.

READY OFF

Command Name: RDF
Suppress the 'ready' message printed at the completion of each ECL command.

FORMAT:

```
RDF
```

PARAMETER DESCRIPTION:
No parameters are required or permitted with this command.

FUNCTION DESCRIPTION:
The READY OFF command suppresses the printing of a message issued by the system at the completion of execution of each ECL command. The message informs the user that the system is prepared to accept another command.
If the RDF command is issued from within an EC file when execution of the EC file is completed, the

system reverts to the ON/OFF state which was in effect when the EC command was invoked.
The initial state of the ready function at the conclusion of task group initiation is OFF.

READY ON

Command Name: RDN
Activate the printing of the ready message at the completion of each ECL command.

FORMAT:

RDN

PARAMETER DESCRIPTION:
No parameters are required or permitted with this command.

FUNCTION DESCRIPTION:
The READY ON command activates the printing of a message issued by the system at the completion of execution of each ECL command. The message informs the user that the ECL processor is prepared to accept another command.
If the RDN command is issued from within an EC file, when execution of the EC file is completed the system reverts to the ON/OFF state which was in effect when the EC command was invoked.
The initial state of the ready function at the conclusion of task group initiation is OFF.

RELEASE

Command Name: RL
Release the space occupied by the named directory or file to the file system.

FORMAT:

$$RL\left[\begin{matrix}\text{-FILE}\\\text{-DIR}\end{matrix}\right]\{path\}...$$

PARAMETER DESCRIPTION:
path
Specifies the pathnames of one or more file system entries to be released.
-FILE
Indicates that the entries to be released are files. This is the default if neither -FILE nor -DIR is specified (see Example 2, below).
-DIR
Indicates that the entries to be released are directories.

FUNCTION DESCRIPTION:
The RELEASE command, when issued to release a file, removes all of the file's attributes from the directory within which it is immediately contained. All of the space which was allocated to the file is returned to the file system. If the file is open at the time the RL command is issued, the file is not released. If the file is reserved for use by another task group (i.e., another task group has issued an ASSOC command specifying this pathname) the file is released after that task group has closed the file and issued a DISSOC command.
When used to release a directory, the RL command removes all of the directory's attributes from the immediately superior directory. All of the space which was allocated to the directory is returned to the file system. The directory to be release must be "empty"; i.e., it cannot contain entries representing subdirectories or files. If it is not empty, it is not deleted.

Example 1:

RL -FILE FILE01

The file, FILE01, in the working directory is released if it is not in use or reserved by another task.

Example 2:

RL SUB_DIR1>FILE02

The file, FILE02, in a directory SUB_DIR1, immediately subordinate to the working directory, is released if it is not in use or reserved by another task.

Example 3:

RL -DIR SUB_DIR1

The directory, SUB_DIR1 immediately subordinate to the working directory, is released, provided it is empty.

RENAME

Command Name: RENAME
Assign a new name to an existing file.

FORMAT:

RENAME oldname newname

PARAMETER DESCRIPTION:

oldname

The present pathname of the file to be renamed.

newname

A simple name unique within the directory containing oldname.

FUNCTION DESCRIPTION:

The RENAME command can be used at any time that it is desired to change the name of an existing file.

The oldname parameter can be a simple, relative, or absolute pathname. The only requirement is that the specified file exist in the expressed or implied directory. If a simple name is given, the file must exist in the working directory. If a relative or absolute pathname is given, the file must exist in the directory derived from the given pathname.

Whatever directory is established by the oldname parameter is the one in which the file will reside under its new name. The new file name must be one which does not already exist in that directory, in accordance with the requirement that, within a given directory, all file names must be unique. It is not possible to rename a file in one directory, and simultaneously establish its location in another directory.

Example 1:

Assume a working directory >UDD>BOOKS>SMITH, and that in this directory there is a file AB. The command

RENAME AB CD

changes the pathname of the affected file from

>UDD>BOOKS>SMITH>AB to >UDD>BOOKS>SMITH>CD.

Example 2:
Assume that within the working directory in example 1 is a subdirectory CHANGES, which contains a file AB_CHANGES. The command

RENAME CHANGES>AB_CHANGES CD_CHANGES

implies the directory >UDD>BOOKS>SMITH>CHANGES, since the oldname parameter is in the form of a relative pathname. The pathname of the file within this directory is changed to >UDD>BOOKS>SMITH>CHANGES>CD_CHANGES.

RESET MAP

Command Name: RS
Reset the bit map representing the available sectors on a volume.

FORMAT:

RS path

PARAMETER DESCRIPTION:
path
Specifies the name of the volume whose bit map is to be reset. The form of path is >SPD>dev_name[>vol_id]. If vol_id is present, the volume name is verified.

FUNCTION DESCRIPTION:
The RESET MAP command constructs a new available-sector map on a volume whose map may have become inconsistent with the actual available sectors through a hardware or software failure or an abort.
Under normal conditions the file system maintains a map which represents the locations of available blocks of disk storage on a volume. The map for each volume is stored on the volume itself. It reflects information contained in directories and subdirectories relating to the locations and extents of the files residing on the volume.
At any given time, the map may also reflect storage used by temporary work files in use by a system component such as a compiler. The locations and extents of such files are known to the system, but not represented in directories. If one of these temporary files is in use at the time of a system failure or an abort, the map will continue to represent the storage used by the file as unavailable, but the system, upon reinitialization, will have lost its information about the location and extent of the file. The space occupied by the temporary file thus becomes permanently unavailable. The RS command restores this space to an available status.
The RS command interrogates all directories and subdirectories on a volume for file location and extent information, and, using this information, constructs a new map. Since the information regarding temporary files is not contained in these directories, their allocations are not represented in the new map, and the space they occupied at the time of the failure or abort is thus made available again.

RPG

Command Name: RPG
Compile the RPG source program unit represented by the indicated file name, applying the specified compiler options.

FORMAT:

RPG path [ctl_arg]

PARAMETER DESCRIPTION:

path

Specifies the name of the file containing the source unit to be compiled.

[ctl_arg]

One or more control arguments chosen from the following list:

{NO_OBJ / -NO}

Indicates that the generation of the object text unit is to be suppressed. If omitted, the object text unit is generated.

{-LIST_OBJ / -LO}

Indicates that a listing of the object text output is to be obtained. Object text is interspersed with source text in the listing. If omitted, object text is not listed.

{-NO_LIST / -NL}

Specifies that all listings are to be suppressed. If omitted, the complete source unit is listed, followed by a listing of the error lines and codes.

{-SIZE nn / -SZ nn}

Designates the number of 1024-word blocks of memory to be used for tables by the compiler. The value of nn can be between 04 to 28, inclusive. If omitted, the assumed value of nn is 03.

-COUT out_path

Indicates that listings which would normally be written to the file path.L are to be written to out_path.

FUNCTION DESCRIPTION:

The RPG command is used to invoke the GCOS 6 RPG Compiler component. Execution of the compiler is normally intended for the batch task group, as is the execution of most of the other components used in a program development activity.

The path parameter can assume any of the acceptable forms of a pathname, although normally it would be a simple name, indicating that a source program unit which resides in the current working directory is to be compiled. Wherever it exists, it must be suffixed with a .R suffix, indicating that it is an RPG language source unit. The path parameter must be given *without* the .R suffix; the compiler appends the suffix prior to searching the directory for the source unit.

If the -COUT control argument is not specified, the specified listings are written to a file created by the compiler in the working directory having a file name of the form path.L. The path portion is the last (or only) element specified in the path parameter. This file can be subsequently listed on a line printer by using the PRINT utility command. If a different file is specified by using the -COUT argument, the listings are written to the file whose pathname is out path. The compiler does not append a .L suffix.

The object text unit generated by the compiler is written to a file whose name is of the form path.O, and is contained in the working directory.

If files of the form path.L and path.O already exist in the working directory, they are overlaid by the output generated by the current compilation.

Example:

```
RPG RGPROG -COUT RGPROG_LST -LIST_OBJ
```

An RPG source program, RGPROG.R, located in the working directory, is to be compiled. Listings are to include source statements, error diagnostics and object code and are to be written to a file named RGPROG_LIST, in the working directory.

SORT FILE

Command Name: SORT
Sort the records in a mass storage file.

FORMAT:

SORT [ctl_arg]

PARAMETER DESCRIPTION:
[ctl_arg]
One or more control arguments chosen from the following list:

-IN path
Specifies the name of the file containing the sort descriptors for this sort. If not specified the user input file is used.

{-SIZE n / -SZ n}

Indicates the number of 1024-word memory modules to be available to the sort. The value of n can be from 8 to 56_{10}, inclusive. If not specified the default value is 8.

-PD
Indicates that a listing of the sort description is to be produced.

FUNCTION DESCRIPTION:
The SORT command provides the capability of sorting a data file according to specifications supplied in a sort descriptor file.
A complete description of the operation and use of the sort component is contained in the Sort manual.

SPAWN GROUP

Command Name: SG
Create, request the execution of, and then delete a task group.

FORMAT:

SG id user_id phys_lvl [in_path] [ctl_arg]

PARAMETER DESCRIPTION:
id
The group identification of the task group to be spawned. It is a two-character name that cannot have the $ as its first character.

user_id
A field comprising two subfields in the form person.project, by which the requested execution of this task group is identified. The user_id subfields are also used to establish the working directory for this request.

phys_lvl
The priority level relative to which all tasks within this task group will execute.

in_path
The name of the file from which commands and user input are to be read by the task group during its execution. The file name is set to null if the in_path parameter is not specified. in_path must be specified if the control argument -ECL (see below) is used or implied.

[ctl_arg]

One or more control arguments chosen from the following list:

-OUT out_path
Defines the pathname of the file which is to receive user output from the task group. If not specified, one of the following assumptions is made:
If in_path specifies a mass storage file, out_path = in_path.AO
If in_path specifies an interactive terminal, out_path = in_path
If in_path is not specified, out_path is null
If in_path specifies an input-only device, out_path is null.

-WD path
Specifies that path is to be used as the working directory pathname instead of the pathname established by the user id parameter.

{-EFN root
-EFN root?entry}
The name of a bound unit root entry which is to be loaded as the lead task. The root segment name can be suffixed with ?entry, where entry is a symbolic start address within the root segment. If not given, the start address established when the bound unit was linked is assumed.

-ECL
The root segment of the execution control language (ECL) processor is to be loaded as the lead task.

-LRN n
Specifies the highest logical resource number (LRN) which will be referred to by any task in the task group. The minimum value which can be specified for n is the highest LRN used by the system task group; this is also the default if this argument is not specified.

-LFN n
Specifies the highest logical file number used by any task in the spawned task group. If -LFN is not specified, n assumes the value 15.

-POOL id
id is a two-character ASCII identifier and is the name of the memory pool from which all memory required by the spawned task group is to be taken. If specified, id must have been defined by a CLM MEMPOOL directive. If not, the issuing task group's memory pool is used.

-ARG
Indicates that additional arguments required by the spawned task group for its execution follow. These additional arguments are passed to the lead task of the spawned group to be used as necessary. If used, the -ARG control argument must appear last.

NOTE: In any invocation of the SG command, -EFN *or* ECL, but not both, can be specified. If neither is specified, -ECL is assumed and the in_path parameter is required.

FUNCTION DESCRIPTION:
The SPAWN GROUP command combines the functionality of the CREATE GROUP, ENTER GROUP REQUEST, and DELETE GROUP commands. It implicitly causes the execution of these three functions in sequence, i.e., allocates and creates the data structures required to define and control the execution of the task group, places a request against the group, thereby activating it, and, when execution terminates, removes all controlling data structures and returns memory used by the task group to the appropriate memory pool.
Because of the sequencing of the functions described above the SG command relieves the user of the issuing task group of the need to be aware of when the spawned task group terminates. The user need take no explicit action to return the terminating group's resources to the system to make them available for use by other task groups. A user may, for example, spawn a task group for another user

who wishes to use the Editor or perform a file dump. This task group exists only for the length of time required to perform its function; when it terminates it is deleted automatically.
The issuing task group can itself be a spawned task group, spawned either by an OCL command issued by the system operator or by an ECL command issued by another online task group. In either case, it has the ECL processor as its lead task.
The SG command can be issued only by an online task group.

SPAWN TASK

Command Name: ST
Create, request the execution of, and then delete a task within the issuing task group.

FORMAT:

ST rel_lvl ctl_arg

PARAMETER DESCRIPTION:

rel_lvl

The priority level, relative to the task group's base priority level, at which the spawned task is to execute.

ctl_arg

One or more control arguments chosen from the following list:

{-EFN root
-EFN root?entry}

The name of a bound unit root segment which is to be loaded for execution. The root segment name can be suffixed with ?entry, where entry is a symbolic start address within the root segment. If no suffix is given, the default start address, established when the bound unit was linked, is assumed.

{-SHARE lrn [ssa]
-SHR lrn [ssa]}

The same bound unit is used as for the task identified by lrn. (This task must have been previously defined by a CREATE TASK command specifying this lrn.) ssa is the symbolic start address within the root segment of the task lrn. If none is given, the default start address of the root segment lrn established when it was linked, is assumed.

-WAIT

Specifies that the task issuing this command is to await completion of the spawned task before resuming execution.

-ARG

Indicates that additional arguments required by the spawned task follow. These additional arguments are passed to the spawned task in an extension of the task request block. If this control argument is used, it must be the last one specified.

NOTE: In any invocation of the ST command, -EFN *or* -SHARE, but not both, must be specified.

FUNCTION DESCRIPTION:
The SPAWN TASK command combines the functions of the CREATE TASK and ENTER TASK REQUEST commands in that it constructs all of the requisite structures for the execution of the task, and then activates it. It performs one additional function -- when the task becomes inactive, it deletes the task.
A spawned task becomes dormant when it issues a TERMINATE Monitor service call and there are no additional requests on the task's request queue. At this time, all controlling data structures associated

with the spawned task are removed, and memory occupied by them is returned to the task group's memory pool.
A spawned task is not assigned a logical resource number. It is therefore "local" to (i.e., visible only to) the spawning task. It cannot be requested or referred to by any other task, nor can its memory space or code be shared. It can, however, share that of another task which was assigned an LRN by means of a previously issued CREATE TASK command. The -SHARE control argument indicates that this sharing is to occur.
Multiple tasks can be made to execute concurrently within a given task group by issuing multiple ST commands. Tasks can also be made to execute serially; i.e., one task going to completion before a subsequent task begins execution. The -WAIT control argument is the mechanism which controls concurrency of execution. Judicious use of this argument can also result in a mixture of concurrent and serial execution (see example 3, below).

Example 1:
Three tasks which have no dependencies among them are to be executed. They can be activated concurrently by issuing the following commands:

```
ST 02 -EFN PROGA
ST 03 -EFN PROGB
ST 04 -SHARE 10
```

Each of the first two spawned tasks executes its own bound unit in its own memory space. The third shares the code and memory space of a previously created task identified by logical resource number 10. If the task group's base priority level was specified as 20 when the group was created, the three tasks execute at physical priority levels 22, 23, and 24, respectively.

Example 2:
The three tasks above have dependencies among them which require them to be executed serially. They are activated by the following commands:

```
ST 02 -EFN PROGA -WAIT
ST 03 -EFN PROGB -WAIT
ST 04 -SHARE 10
```

Each task in this case awaits the completion of the preceding task. Since the third task does not specify -WAIT, another activity can be initiated to run concurrently with it.

Example 3:
The first two of the three tasks are unrelated, but there is a dependency between the second and third tasks. The following commands can be used:

```
ST 02 -EFN PROGA
ST 03 -EFN PROGB -WAIT
ST 04 -SHARE 10
```

This sequence causes the first two tasks to be activated to run concurrently. Since the second task specifies the -WAIT argument, it must terminate execution before the third task can begin. The first task may or may not still be running at this time. As in the previous example, another activity can be initiated to execute concurrently with the third task.

STATUS GROUP

Command Name: STG

Display the status of the issuing task group.

FORMAT:

STG [ctl_arg]

PARAMETER DESCRIPTION:

[ctl_arg]

One or more control arguments chosen from the following list:

-TASKS

Specifies that the statuses of all tasks in the indicated task group are to be listed. This is the default if no control arguments are present.

-FILES

Requests the names of all files that are currently associated with the indicated task group, their types, concurrencies and LFNs.

FUNCTION DESCRIPTION:

The STATUS GROUP command writes to the user output file a summary of the current status of the issuing task group. In addition to information pertinent to the group as a whole, two other categories of status information are displayed: that relating to tasks within the group and that relating to files currently associated with the group.

The following items provide status information relative to the task group as a whole:

- Task group identification
- Current state of the task group:
  - B = Batch, not rolled out
  - R = Batch, rolled out
  - S = Suspended
  - D = Dormant
  - A = Active
- Memory pool identification
- Current user identification
- Full pathname of the error output file
- Full pathname of the user output file

Task-specific status information consists of the following group of items for each task:

- Task logical resource number (if a created task) or the letters ST (if a spawned task)
- Task priority level
- Current state of the task:
  - D = Dormant
  - S = Suspended
  - W = Waiting
  - A = Active
- Task's bound unit name
- Full pathname of the command input file
- Full pathname of the user input file

If there are no tasks currently associated with the task group, a single item, NO TASKS, is returned.

File-specific information consists of the following group of items for each file:

- o Full pathname of the file
- o Concurrency of the file, represented by a decimal digit in the range 1 through 5. The significance of the digits, for the issuing task group and for other task groups, is as follows:

For Issuing Group	*For Other Groups*
1 = Read Only	Read Only
2 = Read Only	Read or Write
3 = Read or Write	No Read, No Write
4 = Read or Write	Read Only
5 = Read or Write	Read or Write

- o File type
- o Logical file number, if one is associated with the file, otherwise spaces
- o Open/closed status of the file

If there are no files currently associated with the task group, a single item, NO FILES, is returned. The group status information is always returned when this command is used. The task-specific information is returned if no control arguments are given, or if explicitly requested by the -TASKS argument. If the -FILES argument is specified, the file-specific, but not the task-specific, information is given.

TIME

Command Name: TIME
Display the current date and time in ASCII format.

FORMAT:

TIME

PARAMETER DESCRIPTION:
No parameters are required or permitted with this command.

FUNCTION DESCRIPTION:
The TIME command returns the current date and time of day in an ASCII character string of the form

yyyy/mm/dd hhmm:ss.mmm

yyyy is the current year
mm is the current month
dd is the current day within the month
hhmm is the time in hours and minutes
ss is the current second within the minute
mmm is the current millisecond

Use of the TIME command presupposes that at some earlier time (usually at system initialization) the system operator has issued an OCL SET DATE (SD) command to establish the current date and time within the operating system. The accuracy of the information returned by the TIME command is totally dependent upon the accuracy of the data entered in the SD command.

SECTION 5

ERROR, STATUS, AND INFORMATIONAL MESSAGES

The MDT system software issues error and status messages through the operator's terminal, or on th ERROR OUT file for a particular task group. In systems that do not have an operator's termina configured, CLM error messages appear in the control panel registers. In interactive mode, whe commands are submitted through the operator's terminal (or another terminal) error messages are issue in response to the command that has just been entered.

MESSAGE CODES

Error messages are prefixed by a six-digit hexadecimal code. They appear in the following format:

(xxyyzz) message

xx
The code of the component that *reports* the error

yy
The code of the component that *detects* the error

zz
The code of the error type within the "yy" category.

Depending upon a particular situation, an error detected by one component (yy code) might be *reported* by any of several different components (xx code). (The reporting component is the one that calls the Error Handler System.) Consequently the messages in the following lists are in order by the code of the *detecting* component. The component codes are listed in Table 5-1. If the error reported by a component was also detected by that component the xx and yy values will be the same. For example, during the execution of the Assembler, the value 1010zz is appropriate for conditions specific to the Assembler itself, such as "symbol table overflow." However, a value of 1002zz indicates a "file not found" condition detected by the File Manager but reported by the Assembler.

TABLE 5-1. COMPONENT CODES

Code	Component
01	Physical I/O
02	File, Data, and Storage Manager
03	Trap Manager
04	Clock Manager
05	Semaphore Functions
06	Memory Manager
08	Monitor
0B	CLM Communications
10	Assembler
11	Linker
12	Utility Programs
13	Configuration Load Manager
14	FORTRAN Compiler
15	FORTRAN Run-time Routines
16	Loader
17	System EC/ECL/OCL Commands
18	Cross-Reference Program (XREF)

TABLE 5-1 (CONT). COMPONENT CODES

Code	Component
19	Editor
21	Patch
22	Communications File Transmission Program
23	Macro Preprocessor
24	Export/Import PAM File Program
25	Dump Edit (DPEDIT) Program
26	COBOL Compiler
27	COBOL Run-time Routines
28	RPG Compiler
29	RPG Run-time Routines
31	Sort
34	Multiline Communications Processor (MLCP) Dump Routine
35-7F	Reserved for software use
80-EE	User defined
F0-FF	Reserved for software use

PHYSICAL I/O MESSAGES (xx01)

For assembly language programs that reference Physical I/O through monitor calls, the error code for physical I/O error messages is the rightmost byte of the return status for the request for an I/O transfer. The return status is contained in register R1. The address of the input/output request block (IORB) used by the driver when the error was detected must be contained in register B4 when the Error Handler System is called.

The format of the physical I/O error messages is as follows.

xx01zz lev cccc sswd dswd

lev
: Level number of driver (two character positions).

cccc
: Channel number (four character positions).

sswd
: Software status word (four character positions). (See Table 5-4 in the Monitor and I/O Calls manual.)

dswd
: Device specific word (four character positions displayed for disk devices only). (See Table 5-12 in the Monitor and I/O Calls manual.)

xx0101 SPECIFIED IORB IS BUSY

The in-use bit of the IORB is set to busy.

xx0102 INVALID LOGICAL RESOURCE NUMBER (LRN)

This message may be due to a configuration error. Correct configuration or task code.

xx0103 ILLEGAL WAIT

Program logic error in task code.

xx0104 INVALID PARAMETER(S)

Invalid parameter(s) passed by component to driver (internal error) or program logic error in task code.

xx0105 DEVICE NOT READY

Prepare device so that it can be used.

xx0106 DEVICE TIMEOUT

Program should retry later.

xx0107 HARDWARE ERROR

Check status bits in software status word (sswd).

xx0108 DEVICE DISABLED

This message may result from a program logic error.

xx0109 FILE MARK ENCOUNTERED

xx010A CONTROLLER UNAVAILABLE

Run controller test and verification.

xx010B DEVICE UNAVAILABLE

This message indicates interruption of the physical connection to a terminal after the connect has been made; e.g., a line drop.

xx010C INCONSISTENT REQUEST

Examples of inconsistent requests are:

A request for connect when the connect has already been made.

A request for disconnect when the connect has not been made.

xx010D MAGNETIC TAPE EOT MARKER DETECTED

FILE SYSTEM MESSAGES (xx02)

For assembly language programs that reference the File System through monitor calls the error code for the File System messages is the rightmost byte of the return status of the call for File, Data, or Storage Management Service. Hardware register R2 must contain the appropriate logical file number (LFN) at the time the Error Handler System Service is invoked.

xx0201 lfn ILLEGAL PATHNAME

Illegal pathname for file management function (ASFIL, GTFIL, RMFIL, CRFIL, RLFIL, RNFIL, GIFIL, CRDIR, RLDIR, CWDIR, XPATH). Possible causes: causes:

Illegal characters or level separators

Length exceeds 57 characters

Space character is absent

Invalid file type (rename of directory or temporary file)

File (pathname) is not a directory on change-working-directory

xx0203 lfn ILLEGAL FUNCTION

Illegal file management function. The subfunction code is illegal for the type of file referenced.

Illegal data management function (RDREC, WRREC, DLREC, RWREC). The subfunction code is illegal for the type of file referenced. For example the user issues a read-with-primary-key call to a sequential file. Although the read-with-key is a legal parameter, the fact that it is a primary key makes it illegal.

Illegal storage management function (RDBLK, WRBLK). The subfunction code is illegal for the type of file referenced. For example write with TM or read with EOT, TM, BOT, or SPACE mode to a disk file.

xx0204 lfn FILE BUSY

For file management function TSFIL: If the LFN is bidirectional, a write order is queued but not completed. If the LFN is unidirectional, a read or write (depending on the direction that characterizes the LFN) order is queued but not completed. Note that a read is always queued for input terminal devices anticipatory read). This call will not return a busy code if the LFN is bidirectional and is busy because an anticipatory read is queued.

If this is the first time a test-file call is issued following an open-file, this message indicates that the connect has not been completed.

For file management function TIFIL: An input request (read) function is queued but not completed. Note that an anticipatory read is always queued for input terminal devices. In this case the busy indication means that there is no data ready to be moved into the user's record area by a read-record call. A read-record issued while the terminal is busy will cause the task to stall until input is received from the terminal.

If this is the first time a test-file call is issued following an open-file, this message indicates that the connect has not been completed.

For file management function TOFIL: An output request (write) is queued but not completed.

If this is the first time a test-file call is issued following an open-file, this message indicates that the connect has not been completed.

xx0205 lfn ILLEGAL PARAMETER

For file management functions (ASFIL, DSFIL, GTFIL, RMFIL, CRFIL, RLFIL, RNFIL, OPFIL, CLFIL, GIFIL, TSFIL, TIFIL, TOFIL, RLDIR, CWDIR, GWDIR, XPATH):

The file information block (FIB) or parameter structure block (PSB) pointer (loaded into register B4) contains a null value.

A pathname is not specified and the LFN specified is not associated.

An unknown file type has been specified.

For data management functions (RDREC, WRREC, DLREC, RWREC). The FIB pointer contains a null value.

For storage management function (RDBLK, WRBLK, WTBLK). The FIB pointer contains a null value.

For file management function CRFIL with the following file types:

Unified files:
The control inverval (CI) size is not a multiple of 256.

Fixed Relative files:
The CI size is not a multiple of 128.

Unified Indexed files:
The number of key descriptors is not one.
The key descriptor pointer is null.
The number of key components is not one.
The key component data type is not C.
Creating a temporary file.
The CI size is not large enough to hold at least two index entries ((CI size - 6)/(key size + 6) is less than 2).
The key is not located within the record.

Unified Relative and Index files:
The record size plus overhead per CI exceeds CI size.

All files:
Initial allocation exceeds maximum specified.
Initial or incremental allocation exceeds extent limit (8191 physical sectors).

xx0206 lfn UNKNOWN OR ILLEGAL LFN

For file management functions (ASFIL, DSFIL, GTFIL, RMFIL, CRFIL, RLFIL, RNFIL, OPFIL, CLFIL, GIFIL, TSFIL, TIFIL, TOFIL): The LFN is not in the legal range for this task group or the LRN was not previously associated with a pathname by an ASFIL, GTFIL, or CRFIL call.

For data management function (RDREC, WRREC, RWREC, DLREC). The LFN is not valid in all system control structures.

For storage management function (RDBLK, WRBLK, WTBLK). The LFN is not in the legal range for this task group or there is an inconsistency in LFN values in system control structures.

x0207 lfn FILE NOT OPEN

For file management function (DLFIL, TSFIL).
For data management function (RDREC, WRREC, RWREC, DLREC).
For storage management function (RDBLK, WRBLK, WTBLK).

For all these functions, the LFN in the FIB or PSB indicates a file that is known and attached to a task group but one that is not currently opened. The file must be opened before the desired function can be performed.

xx0208 lfn FILE ALREADY OPEN

For file management function (GTFIL, RMFIL, CRFIL, RLFIL, OPFIL). The LFN in the FIB or PSB designates a file that is known, attached to a task group, and already open under the specified LFN.

xx0209 lfn PATHNAME NOT FOUND

For file management function (GTFIL, RMFIL, RLFIL, RNFIL, OPFIL, GIFIL, CRFIL, RLDIR, CWDIR, CRDIR.) The pathname specified is syntactically correct but the file which it describes is not present on any volumes accessible to the requesting task.

For file management function CRFIL, some superior directory is not found.

xx020A lfn ADDRESS OUT OF FILE

For data management function (RDREC, WRREC, RWREC, DLREC). The supplied key (which is either a control interval and line number, or a relative record number)

is outside the file boundaries. For example, the file contains 1000 records and the user issues a RDREC for record number 2000.

For storage management function (RDBLK, WRBLK, WTBLK). The supplied key is outside the file boundaries.

xx020C lfn VOLUME NOT FOUND

For file management function (GTFIL, RMFIL, CRFIL, RLFIL, RNFIL, OPFIL, GIFIL, CRDIR, RLDIR, CWDIR). A syntactically correct pathname has been specified that identifies a volume that is not currently mounted.

xx020E lfn RECORD NOT FOUND

For data management function (RDREC, WRREC, DLREC). Record specified by key is deleted or it never existed.

xx0210 lfn LFN ALREADY ASSOCIATED

For file management function (ASFIL). A request has been made to associate an LFN which is already associated with a pathname. The current LFN must be disassociated before another association can be performed.

xx0211 lfn LFN NOT AVAILABLE

For file management function (GTFIL, CRFIL). The requested function specifies that an LFN is to be established (search for a currently unused LFN) and there are no more in the pool available to the task group. An existing LFN must be relinquished or the LFN pool must be enlarged.

xx0212 lfn FILE ALREADY EXISTS

For file management function CRFIL. The request to create a file specifies one that already exists in the containing directory.

For file management function RNFIL. The request to rename a file uses a new name that already names a file in the containing directory.

For file management function CRDIR. The request to create a directory specifies one that already exists in the containing directory.

xx0213 lfn UNABLE TO PROVIDE REQUESTED LEVEL OF CONCURRENCY CONTROL

For file management functions (GTFIL, OPFIL): The requested level of concurrency control conflicts with current active usage of specified file.

For file management function RNFIL, the file is already open or the file is a directory.

For file managment function RLDIR the directory is currently in use e.g., it is the current working directory of the same task.

xx0214 lfn BAD PROGRAM VIEW OF FILE

For file management function OPFIL, the user visibility to the file specified in program view (FIB) does not agree with the information stored in the file directory.

xx0215 lfn FILE SPACE NOT AVAILABLE

For file management functions (CRFIL, CRDIR): There is not sufficient file space on the specified volume to create the file/directory specified.

For storage management function WRBLK, a write has been issued that requires an additional extent to be allocated but there is not enough space on the volume for another extent.

xx0217 lfn ACCESS VIOLATION

For data management function (RDREC, WRREC, RWREC, DLREC). An attempt to access the file conflicts with:

Concurrency control established when the file was assigned to the task group (by GET-FILE primitive/ECL)

Processing rules specified in the FIB

For example, the user issues WRREC when he has only read access or shared reac with no write concurrency control.

For storage management function (RDBLK, WRBLK,). An attempt to access the file conflicts with:

Concurrency control established when the file was assigned the task group (by GET-FILE primitive/ECL)

Processing rules specified in the FIB

For example, the user issues WRBLK when he has only read access or shared read with no write concurrency control.

xx0219 lfn NO CURRENT RECORD POINTER

For data management function (RDREC, WRREC). A possible cause is that a previous function left the read/write current pointer at or beyond the end of file.

For data management function (RWREC, DLREC). A key is not supplied in this function call. This means rewrite or delete the "current" record and the previous function was not a read i.e., there is no current record.

xx021A lfn RECORD LENGTH ERROR

For data management function RDREC. For variable length record files, the record read is larger than the user record area. For fixed length record files, the record read is not the same size as the user record area.

For data management function WRREC. Record to be written is larger than the maximum record size for the relative files.

For data management function RWREC. For all files, except relative files containing variable length records an attempt was made to change the size of a logical record during the rewrite function. For relative files having variable length records, the new record length exceeds the maximum record length declared for the file. The record is not rewritten.

xx021B lfn DUPLICATE KEY

For data management function WRREC. For an indexed file, a record with the same key value already exists in the file. For a relative file, an attempt was made to write with a simple key or relative key to an active record. The record is not written.

xx021C lfn KEY OUT OF SEQUENCE

For data management function WRREC. During loading of an indexed file ("renew" mode) a record was not in ascending sequence by key value.

xx021D lfn KEY CHANGE ERROR

For data management function RWREC. For indexed files nn attempt was made to change the record's key value during the rewrite function. The record is not rewritten.

xx021E lfn KEY LENGTH ERROR

For data management function (RDREC, WRREC, RWREC, DLREC). Key size does not match the size of the key as defined for the file.

xx021F lfn END OF FILE

For data management function RDREC. End of file was reached. No record was read.

For storage management function RDBLK. There is not sufficient data remaining in the file to fill the specified buffer. What is left in the file is delivered.

For storage management function WRBLK. There is insufficient space in the file to contain the data specified for writing.

xx0220 lfn DIRECTORY NOT EMPTY

For file management function RLDIR, a request was made to delete a directory that contains a file of subordinate directory entries. All files and subordinate directory entries must be deleted separately before the directory can be deleted.

xx0222 lfn NO CURRENT WORKING DIRECTORY

For file management functions (ASFIL, GTFIL, RMFIL, CRFIL, RLFIL, RNFIL, OPFIL, GIFIL, CRDIR, RLDIR, GWDIR, XPATH): A relative pathname was specified as input and there is no current working directory for the task.

xx0223 lfn FILE SPACE LIMIT REACHED OR FILE NOT EXPANDABLE

For file management functions (CRFIL, CRDIR) A superior directory may require expansion as a result of adding entries to describe the file or directory. If the superior directory is not expandable, or is already expanded to the limit an error occurs. Note that the root or volume major directory is not expandable.

xx0231 lfn END OF TAPE

For storage management function WRBLK, the physical end of the tape has been reached.

TRAP HANDLER MESSAGES (xx03)

Each error code in the following messages except codes 1F through 21, is the number of the trap vector. Error codes 1F through 21 pertain to improper use of user trap functions.

xx0301 MONITOR CALL

This trap indicates a system software problem.

xx0302 BRK INSTRUCTION

This trap is normally handled by the Debug program.

xx0303 SCIENTIFIC FLOATING-POINT INSTRUCTION NOT IN HARDWARE

Trap 3 occurs if the instruction is a scientific floating-point instruction. If the SIP Simulator is present, it serves as a trap handler for trap 3, and the trap is not visible to the user. If the SIP and the SIP Simulator are not present, the user must provide a trap handler or the task will be aborted.

xx0304 UNRECOGNIZED INSTRUCTION - SIP SIMULATOR PRESENT

See explanation for message xx0305.

xx0305 INSTRUCTIONS (OTHER THAN FLOATING-POINT INSTRUCTIONS) NOT IN HARDWARE

If the instruction is a scientific branch instruction, the SIP Simulator if present, serves as a trap handler. If the instruction is a scientific branch instruction and the SIP Simulator is not present, the task is aborted unless the user provides a trap handler for it.

If the SIP Simulator is present, all other unrecognized instructions including double precision instructions on a single precision simulator produce a trap to trap 4. If trap 5 occurs and the SIP Simulator is not present the task will be aborted unless the user provides a trap handler.

xx0306 INTEGER ARITHMETIC OVERFLOW

This trap occurs when the overflow bit in the I-register is set to 1 as a result of an operation on an R-register while the M-register "overflow trap enable" bit for this R-register is set to 1.

xx0307 SCIENTIFIC DIVIDE BY ZERO

This trap occurs when an SDV (scientific divide) instruction is encountered that has a divisor of zero.

xx0308 SCIENTIFIC EXPONENT OVERFLOW

This trap occurs during the execution of a scientific instruction if exponential overflow takes place.

xx03OD UNPRIVILEGED USE OF PRIVILEGED OPERATION

This trap occurs when the central processor attempts to execute a privileged instruction while running in unprivileged mode.

xx03OF REFERENCE TO UNAVAILABLE RESOURCE

This trap occurs when the central processor attempts to process an instruction and one of the following conditions exists. (1) The effective address developed is outside specified limits. (2) An input/output instruction contains an improper channel number. (3) A WDTN (Watchdog Timer ON) or WDTF (Watchdog Timer OFF) instruction occurs when the watchdog timer is not installed.

xx0310 CPU DETECTED PROGRAM LOGIC ERROR

This trap occurs when (1) the central processor attempts to execute an RTT (Return From Trap) instruction normally issued by a trap handler and a trap save area to be dequeued cannot be found, or (2) the central processor attempts to execute an instruction that illegally contains a register address syllable.

xx0311 MEMORY OR MEGABUS ERROR

This trap occurs when an uncorrectable memory error or a megabus parity error is detected.

xx0313 SCIENTIFIC EXPONENT UNDERFLOW (IF ENABLED)

This trap results from an operation that generates a characteristic value of 128 too large while the EUM enable bit in the SIP trap mask register (M5) is set to 1.

xx0314 SIP DETECTED PROGRAM ERROR

This trap occurs when program errors are detected by the SIP. Note that program errors detected by the CPU activate trap vector 16.

xx0315 SCIENTIFIC SIGNIFICANCE ERROR

This trap results from an operation in which an integer is truncated during a floating-point to integer conversion while the SE enable bit in the SIP trap mask register (M5) is set to 1.

xx0316 SCIENTIFIC PRECISION ERROR

This trap results from an operation in which the nonzero portion of a fraction is truncated while the PE enable bit in the SIP trap mask register (M5) is set to 1.

xx03IF ILLEGAL TRAP ADDRESS
Address of trap handling routine is invalid.

xx0320 ILLEGAL TRAP NUMBER
Trap requested is not user class trap.

xx0321 TRAP ADDRESS NOT SET
Trap handler entry not connected.

CLOCK MANAGER MESSAGES (xx04)

xx0401 ILLEGAL DATE, TIME, OR INTERVAL VALUE

xx0402 INVALID RECEIVING FIELD LENGTH
Invalid receiving field length in conversion to external date/time.

xx0403 INVALID BASIC TIMER SPECIFIED
Occurs only on requests for Clock Manager services.

xx0404 REFERENCED CLOCK REQUEST BLOCK (CRB) NOT CONNECTED TO BASIC TIMER
Occurs only on cancellation of requests for Clock Manager services.

xx0405 INVALID CLOCK REQUEST BLOCK (CRB) FORMAT

xx0406 CLOCK REQUEST BLOCK (CRB) DEQUEUED
CRB dequeued by another task while wait in progress.

xx0407 INVALID EXTERNAL DATE

xx0408 INVALID EXTERNAL TIME

SEMAPHORE FUNCTION MESSAGES (xx05)

xx0501 RESOURCE UNAVAILABLE

xx0502 UNDEFINED SEMAPHORE OR SEMAPHORE REQUEST BLOCK (SRB)

xx0503 DUPLICATE SEMAPHORE NAME

MEMORY MANAGER MESSAGES (xx06)

xx0601 ILLEGAL MEMORY SIZE OR MEMORY POOL

xx0602 NO SPACE
Insufficient space in memory pool for memory requested.

xx0603 BLOCK RETURNED IS OUT OF MEMORY
Block returned by the batch queue's task group is not within its own memory pool, or block returned by any other task group is not within the managed memory.

xx0607 INVALID WAIT ARGUMENT

MONITOR ERROR MESSAGES (xx08)

xx0801	REQUEST BLOCK BUSY (CRB, IORB, IRB, RB, or SRB)
xx0802	INVALID LRN
xx0803	INVALID WAIT
xx0804	DUPLICATE GROUP-ID
xx0805	UNBALANCED DELIMITERS IN COMMAND LINE
xx0806	INVALID GROUP-ID
xx0807	INVALID MEMORY POOL-ID
xx0808	INVALID LEVEL
xx0809	ILLEGAL HIGH LRN
xx080A	ILLEGAL HIGH LFN
xx080B	ILLEGAL USER-ID
xx080C	UNRESOLVED SYMBOLIC START ADDRESS
xx080D	GROUP NOT SUSPENDED
xx080E	REFERENCED ABSENTEE REQUEST DOES NOT EXIST
xx080F	STREAM (COMMAND IN, USER IN, USER OUT, OR ERROR OUT) UNDEFINED
xx0810	ILLEGAL MAJOR FUNCTION CODE
xx0811	ILLEGAL MINOR FUNCTION CODE
xx0812	INVALID LRN (TASK ABORTED OR BEING DELETED)
xx0813	DUPLICATE LRN
xx0814	NO REQUEST TO BE DEQUEUED
xx0815	NO FILE DESCRIPTOR BLOCK (FDB) OR NO DATA DESCRIPTOR BLOCK (DDB) DEFINED
xx0816	NO WORK AREA DEFINED (SYSTEM SOFTWARE ERROR)

CLM COMMUNICATIONS ERROR MESSAGES (xx0B)

The communications extension to the Configuration Load Manager (CLM) checks the user's description of the hardware and software and reports any inconsistencies as shown by the following messages. The format of these messages is given below.

```
xx0Bzz      3E
[additional information]
[command]
```

First line

xx indicates the component that reports the message

OB indicates the message is detected by the communications extension

zz is the error code

3E is the level on which the CLM runs

Second line

The additional information, if any, depends on the error as shown in the listing below.

Third line

Where applicable the directive in which the error occurs is reproduced. If a directive is printed but there is no additional information the directive appears on the second line.

xx0B01	COMM DIRECTIVE MUST PRECEDE DESIGNATED DIRECTIVE
directive	TTY, VIP, BSC or LPHn directive is given before a COMM directive.
xx0B02	COMM DIRECTIVE ALREADY GIVEN
interrupt level directive	The interrupt level is that specified in the first COMM directive.
xx0B03[1]	DIRECTIVE IS MISSING
	COMM directive is given but no TTY, VIP, BSC, or LPHn directive is given.
xx0B04[1]	LOADING ERROR
$CRTSK or loader error	Error in loading communications supervisor/MLCP driver bound unit.
xx0B10	MODEM NUMBER OR LPH NUMBER OUT OF RANGE
modem or LPH number directive	Modem number in MODEM is not in the range 3 through 15 or LPH number in LPHDEF is not in the range 0 through 3.
xx0B11	LRN OUT OF RANGE
lrn directive	LRN in TTY, VIP (screen or ROP), BSC, LPHn or STATION directive is not in the range 0 through 255.
xx0B12	INTERRUPT LEVEL OR REQUEST LEVEL OUT OF RANGE
level directive	Interrupt level in COMM directive is not in the range 0 through 62, or request level in TTY, VIP, BSC, or LPHn directive is not in the range interrupt level +1 through 62.
xx0B13	INVALID CHANNEL NUMBER
channel number directive	Low order 6 bits of channel number in TTY, VIP, BSC, or LPHn directive are not al zeros.
xx0B14	MODEM NUMBER UNDEFINED OR OUT OF RANGE
modem number directive	Modem number in TTY, VIP, BSC, or LPHn directive is not in the range 0 through 2, or is not defined in a MODEM directive.
xx0B15	INVALID SPEED VALUE
speed directive	Speed in TTY or LPHn directive is not 50, 75, 110, 134, 150, 300, 600, 900, 1200, 1800, 2400, 3600, 7200, or 9600.

[1] After this message, the system halts.

xx0B16	POLL ADDRESS OUT OF RANGE
poll address directive	Poll address in VIP directive is not in the range 0 through 31.
xx0B17	INVALID FIRST CHARACTER OF CONTROL/TRIBUTARY
first character directive	First character of control/tributary in VIP directive is not C or T.
xx0B18	INVALID ROP TYPE
directive	ROP type in VIP directive is not TTY33, TTY35, TN100 TN150, TN300, or TN1200.
xx0B19	DESIGNATED ROP FORM FEED IS INVALID
first two characters directive	First two characters of ROP form feed in VIP are not F0 or N0.
xx0B1A	FIRST CHARACTER OF PRIMARY/SECONDARY INVALID
first character directive	First character of primary/secondary in BSC directive is not P or S.
xx0B1B	FIRST TWO CHARACTERS OF CHARACTER SET INVALID
first two characters directive	First two characters of character set in BSC directive are not AS, EB, or TE.
xx0B1C	FIRST TWO CHARACTERS OF FDX/HDX INVALID
first two characters directive	First two characters of FDX/HDX in LPHn directive are not FD or HD.
xx0B21	DUPLICATE LRN
lrn directive	LRN in TTY, VIP (screen or ROP), BSC, LPHn or STATION directive is the same as the lrn for another device or task.
xx0B22	INVALID INTERRUPT LEVEL OR REQUEST LEVEL
level directive	Interrupt level in COMM directive or request level in TTY, VIP, BSC, or LPHn directive has been specified for a noncommunications device or for a task.
xx0B23	INVALID CHANNEL NUMBER
channel number directive	Channel number in TTY, VIP, BSC, or LPHn directive is the same as the channel number of a noncommunications device, or Two communications devices have the same channel number but are separated in the CLM file by a communications device with a different channel number, or The channel number is the same as the channel number of a communications device using a different line protocol handler, a different modem type, or a different speed, or The channel number is the same as the channel number of another TTY or BSC device.

xx0B26	CHANNEL NUMBER INCOMPATIBLE WITH POLL ADDRESS
poll address directive	Channel number and poll address in a VIP directive are the same as the channel number and poll address in another VIP directive, or The channel number in a VIP directive in which a poll address is specified is the same as the channel number in another VIP directive in which no poll address is specified.
xx0B27	CPU CANNOT BE A TRIBUTARY ON A POLLED LINE
directive	Control/tributary in a VIP directive in which a poll address is specified is T. The CPU may not be a tributary station on a polled line.
xx0B28	INVALID ROP TYPE OR FORM FEED
directive	ROP type or form feed is specified in a VIP directive in which no LRN is specified for an ROP.
xx0B33	LINE ADAPTOR OR DEVICE ERROR
channel number directive	No line adaptor or device at channel specified in TTY, VIP, BSC, or LPHn directive.
xx0B33	LINE ADAPTOR OR DEVICE ERROR
ID directive	ID of line adaptor or device at channel specified is not x'21xx' or ID of line adaptor at channel specified in a TTY directive or an LPHn directive in which a speed is specified is not x'2108', x'2110', or x'2118', or ID of line adaptor at channel specified in VIP or BSC directive is not x'2158'.
xx0B40[2]	FUNCTION NUMBER OUT OF RANGE
	Initialization subroutine called ZGQISB with function number not in the range 0 through 4.
xx0B41[2]	CCP REQUIRES TOO MUCH MEMORY
	LPH initialization tried to load a CCP into MLCP memory and the CCP did not fit into the 3072-byte area for CCP's.
xx0B48[2]	CCP OR LCT CANNOT BE LOADED - MLCP REPEATEDLY BUSY
	LPH initialization tried to load a CCP or LCT into MLCP memory and the MLCP was repeatedly busy.
xx0B49[2]	CCP OR LCT CANNOT BE LOADED - UNCORRECTED MEMORY ERROR
	LPH initialization tried to load a CCP or LCT into MLCP memory and the MLCP detected an uncorrected main memory error.
xx0B4A[2]	CCP OR LCT CANNOT BE LOADED - INCORRECT PARITY
	LPH initialization tried to load a CCP or LCT into MLCP memory and the MLCP detected incorrect parity for a character on the megabus.
xx0B4C[2]	CCP OR LCT CANNOT BE LOADED - INVALID ADDRESS
	LPH initialization tried to load a CCP or LCT into MLCP memory at an invalid main memory address.

[2] After this message the system halts

ASSEMBLER MESSAGES (xx10)

xx1007 — INVALID CONTROL ARGUMENT

Reenter ASSEM command using valid control argument.

xx100A — INSUFFICIENT STARTING MEMORY

Rerun in a pool of larger size, or reinitialize increasing the size of the current pool.

xx100B — INVALID - SIZE ARGUMENT

Reenter command using a valid - SIZE argument.

xx100C — FILE NAME NOT DESIGNATED

The first (or only) argument of the ASSEM command must designate the file name of the source module to be assembled. Reenter the command.

xx100D — SYMBOL TABLE OVERFLOW

Rewrite the source module so that some labels are temporary rather than permanent, or

Rewrite the source module so that it is several smaller modules, or

If additional memory is available, change the -SIZE argument and reenter the command.

The following messages are uncoded:

ASSEM vvrr — This message appears when the assembler is turned on; where vv is the version and rr the revision.

mmmm ERR COUNT — This message appears when the assembler is finished; mmmm is the number of errors.

LINKER MESSAGES (xx11)

Text messages from the linker are sent to the ERROR OUT and the LIST file (determined by the -COUT argument in the LINKER ECL command). If the Monitor reports a system error that results in termination of the Linker, an error message in the form 11yyzz is sent to ERROR OUT before termination. Such errors will usually be data management errors producing messages in the form 1102zz. Text messages generated by the Linker appear without numeric identification.

**LINK TERMINATED

Linker execution is terminated. The link was unsuccessful. This message appears after each of the following messages that result in termination of the Linker.

OVERLAY overlay name HAS BEEN MULTIPLY DEFINED

Each overlay name must be unique. This message indicates that a name has been used twice as an overlay name or as both an overlay name and an external value definition. The link is terminated. Correct the overlay name and relink.

IST ERR

The name specified in the IST directive is not found in the Linker symbol table. This message is for information only.

NO WORK FILE - NO LINK OUTPUT

Not enough work space is available in the working directory to allow for the initial allocation of LNKWRK.W, or an I/O error has occurred on the LNKWRK.W file on the working directory. The link is terminated. Relink with more space in the working directory.

PATHNAME TOO LONG

The pathname of the input object file including the automatically appended .O is too long. The link is terminated. Check the working directory and filenames and relink.

CMD ERR

The linker directive is invalid. Correct the directive. Linking continues.

TBL OV

The space allocated to the Linker symbol table is too small. The link is terminated. Relink increasing the value of the – SIZE argument in the ECL command.

RT/OV TOO BIG

The root or overlay being linked is greater than 64K words or it has an invalid start address. The link is terminated.

DATA SPACE ON OUTPUT FL EXCEEDED

There is not enough room on the output device to hold the bound unit. The link is terminated. No bound unit is created. Relink using another directory or disk for the output file.

RD ERR - OUTFL

A read I/O error in the bound unit file has occurred. The link is terminated. Relink using another directory or disk for the output file.

* NO LINK OUTPUT

The link has not been successfully completed. There is no executable bound unit.

NO LINKER COMMAND FILE

The Linker command file as specified in the ECL command (-IN) does not exist. The link is terminated. Resubmit the LINKER ECL command with the corrected -IN argument.

NO LIST FL - NO MAPS

The file specified in the -COUT argument of the ECL command does not exist. The link will continue but no list file output will be created.

INV SZ PARAMETER

The specified size of the ECL -SIZE argument was not in the range of 1 to 32K inclusive. The link is terminated.

UTILITY PROGRAMS MESSAGES (xx12)

xx1201 ILLEGAL PATHNAME

If the directive is COPY/COMPARE, the pathname is a directory and not a file.
If the directive is LIST NAMES, the pathname is not a directory.

xx1203 ILLEGAL NUMBER OF ARGUMENTS

xx1204 ILLEGAL ARGUMENT LENGTH

xx1207 ARGUMENT NOT RECOGNIZED

xx1209 ILLEGAL COMBINATION OF ARGUMENTS

xx120A REQUIRED OUTPUT DOES NOT EXIST
For COMPARE file directive, output file must exist.
For COPY directive, output volume must exist i.e., be initialized.

xx120F ALLOCATED SPACE EXCEEDED
On COPY directive, end of output is reached before end of input.

xx1211 REDUNDANT ARGUMENT

xx1212 REQUIRED ARGUMENT MISSING

xx121A DEVICE TYPE ILLEGAL FOR REQUESTED FUNCTION

xx121B FILES/VOLUMES DO NOT COMPARE

CONFIGURATION LOAD MANAGEMENT ERROR MESSAGES (xx13)

Error messages generated by the Configuration Load Manager have the format:

```
(xx13zz) hh
[s] [msg]
```

xx is the error number
hh is the level of the task group in which the CLM is operating
s and msg are secondary messages.

If s and msg are missing, the second line of the error notice is omitted, if s is missing (but msg is present) spaces are substituted for s. The meaning of s and msg depends on the message as explained in the error message listing below.

For all messages that begin with the word CMD, the faulty directive statement is printed on ERROR OUT and the next directive, only is read from COMMAND IN. The operator then has the option of correcting the directive or bypassing it (by typing an asterisk followed by a carriage return). The next directive after the operator action will be read from USER IN.

If an error notification occurs in the execution of the CLM and there is no operator's terminal configured, the system will halt with the following register contents:

($R1) = Primary error number (13zz)
($R2) = Secondary error number, if applicable
($B3) = Pointer to directive buffer (If zz is a CMD error)
($B4) = Pointer to secondary text buffer (Null if no secondary text)

Processing cannot continue after this type of halt.

xx1301 CMD DIRECTIVE INVALID
The directive has been misspelled, or it does not begin in column 1 of the line.

xx1302 CMD PARAMETER REQUIRES DECIMAL DIGIT
s Nondecimal digit specified where decimal is required. The parameter number is s.

xx1303 CMD PARAMETER REQUIRES SMALLER DECIMAL NUMBER
s The parameter number is s.

xx1304 CMD PARAMETER REQUIRES HEX DIGIT

s Nonhexadecimal digit specified where hexadecimal is required. The parameter number is s.

xx1305 CMD PARAMETER REQUIRES SMALLER HEX NUMBER

s The parameter number is s.

xx1306 CMD INCLUDES A PARAMETER ERROR

s

The parameter number is s. This message is issued if:

1. A terminal apostrophe has been omitted,
2. 64 characters have been collected or the end of line has been reached,
3. A string beginning with an alphabetic character rather than an apostrophe is longer than 64 characters.

xx1309 HALT. CANNOT LOAD CLM COMMAND DIRECTORY

s msg The load status is s. The name of the directory is msg. To try again to load, press RUN and EXECUTE. To continue without a retry, change $R1 to 0 and then press RUN and EXECUTE.

xx130A CANNOT READ COMMAND FROM USER IN

s Only the next command is read from COMMAND IN. An s of 021F indicates that no QUIT command was encountered before the end of file.

xx130F CMD ERROR DUE TO MISSING OR FAULTY PARAMETER

A required parameter is missing or the wrong type of parameter is used. The types are numeric and alphanumeric.

xx1310 CMD (SYS) INCLUDES INVALID SIP PARAMETER

xx1311 CMD (SYS) INCLUDES INVALID OLAN PARAMETER

xx1312 CMD (DEVICE) ERROR

s Cannot assign operator's terminal; s is the error code.

xx1313 CMD (SYS) CONFLICTS WITH PREVIOUS COMMAND

A preceding MEMPOOL command specified an exlusive type pool requiring all of the pool area (SIZE was *).

xx1314 MEMORY AREA REQUESTED IS TOO LARGE

msg A memory pool set requests total memory area which is too large for the area available; msg gives the boundaries of the area (in multiples of 32 words). Processing cannot continue after this error.

xx1315 CMD (MEMPOOL) INCLUDES AN INVALID POOL NUMBER

The number may be a duplicate of a previous one.

xx1316 CMD (MEMPOOL) A POOL TYPE PREVIOUSLY ASSIGNED

This directive specifies a B type pool, and a previous directive already has defined one.

xx1317 CMD (RESOLA) SPECIFIES INVALID OVERLAY

s The overlay whose name is given in parameter number s, is not floatable. CLM continues processing succeeding parameters. (This is a system error.)

xx1318 CMD (RESOLA) SPECIFIES OVERLAY NOT IN DIRECTORY

s The overlay, whose name is given in parameter number s, cannot be found in the system overlay directory. CLM continues processing the succeeding parameters.

xx1319 CMD (SYS) SPECIFIES INVALID HZ PARAMETER

The hz parameter is not 50 or 60 (or null).

xx131A CMD (SYS) SPECIFIES INVALID SCAN PARAMETER

The scan parameter must be one of the following millisecond values:

hz = 60(or null)	*hz = 50*
8	10
16	20
	25
33	
50 (or null)	50 (or null)
100	100

xx131F CMD (MEMPOOL) SPECIFIES TOO MANY POOLS

Only one pool can be specified in an S- or B-type set.

xx1320 CMD (MEMPOOL) SPECIFIES INVALID POOL NUMBER

There is a null pool number but the type is not S or B; or the pool number is not alphanumeric.

xx1321 CANNOT GET BLOCK REQUESTED

s s = FFFF—Zero size block requested
s = 06zz—See memory management errors.
CLM processing cannot continue after this error.

xx1322 CMD (CLMIN) SPECIFIES INVALID PATHNAME

s s = 02zz—See file management errors.

xx1323 CMD (MEMPOOL) OMITS POOL SIZE

Pool size must be specified.

xx1324 CMD (DEVICE)SPECIFIES INVALID DEVICE TYPE

Type must be one of the strings described in Section 2, of the System Control manual.

xx1325 CMD SPECIFIES LRN GREATER THAN 255

xx1326 CMD SPECIFIES LEVEL GREATER THAN 62

xx1327 CMD SPECIFIES LEVEL LESS THAN 5

xx1328 CMD SPECIFIES LRN PREVIOUSLY ASSIGNED

The LRN has already been assigned to another device of a different type.

xx1329 CMD SPECIFIES DUPLICATE LEVEL PARAMETER

xx132A CMD (DEVICE) SPECIFIES DUPLICATE CHANNEL

The channel specified has already been assigned to another device.

xx132B LEVEL ALREADY RESERVED

xx132C CMD (RESOLA) SPECIFIES OVERLAY THAT IS ALREADY PERMANENT

s Overlay whose name is given in parameter number s, is already permanent.

xx132D CMD (DEVICE) PREVIOUSLY REFERENCED

xx132E CMD (DEVICE) SPECIFIES WRONG LRN

LRN must be 0 with default KSR channel number.

xx132F CMD SPECIFIES ASCII NAME THAT IS TOO LONG

A file, module, or symbol name contains too many characters.

xx1330 CMD (DEVICE) SPECIFIES DUPLICATE FILE NAME

xx1331 CMD (DEVICE) SPECIFIES INCORRECT EXTEND PARAMETER

xx1332 ROLLOUT FILE CANNOT BE CREATED

s Rollout file for batch area cannot be created. Batch area will be destroyed if rollout occurs.
s = 02zz–See file management errors.

xx1333 CMD (MEMPOOL) MEMORY POOL DEFINITIONS IN ERROR

Memory pool definitions are specified subsequent to a pool definition with * size.

xx1334 CMD (MEMPOOL) SPECIFIES INVALID POOL TYPE

Pool type must be S, B, E, or null.

xx1336 HALT. ERROR IN LOADING CLM OVERLAY

s msg s is the load status; msg is the name of the CLM root whose overlay cannot be loaded. CLM cannot be continued.

xx1337 ERROR IN LOADING PERMANENT SYSTEM OVERLAY

s msg s is the load status; msg is the overlay number (hexadecimal). The overlay is not permanently loaded.

xx1340 HALT. CANNOT LOAD ROOT OF BOUND UNIT

s msg Cannot load root of bound unit specified in an LDBU directive.
s is the load status; msg is the name of the bound unit. To try to load again, press RUN and EXECUTE. To bypass the load, set $R1 to zero, then press RUN and EXECUTE.

xx1345 CMD (DEVICE) SPECIFIES DUPLICATE DEVICE TYPE OR UNIT

xx1346 CMD (DEVICE) SPECIFIES INVALID BUFFER PARAMETER

xx1347 CMD (DEVICE) SPECIFIES INVALID SPD

SPD directory cannot be located. This is a system software error.

xx134B INITIALIZATION SUBROUTINE ERROR

s Error detected during load initialization. s is the error code of the initialization subroutine. Processing cannot continue.

xx134F CMD (DEVICE) SPECIFIES DRIVER THAT CANNOT BE LOADED

S is the load status.

FORTRAN COMPILER MESSAGES (xx14)

If the Monitor reports a system error that results in termination of the compiler, an error message in the form 14yyzz is issued before termination. Such errors will usually be data management errors producing messages in the form 1402zz.

xx1407 INVALID ARGUMENT

An invalid argument was specified in the load command line. Control returns to the ECL Processor. Reinvoke the FORTRAN Compiler, specifying a correct argument in the load command line.

xx1412 PROGRAM NAME NOT DESIGNATED

A program name was not designated in the command line. Control returns to the ECL Processor. Enter a new command line specifying a program name.

The following messages are uncoded:

FORTRAN vvrr

This message appears when the compiler is turned on; vv denotes the version rr denotes the revision.

mmmm ERR COUNT

This message appears when the compilation is finished mmmm denotes the number of errors.

FORTRAN RUNTIME INPUT/OUTPUT ROUTINE MESSAGES (xx15)

xx1501 RECORD LENGTH EXCESSIVE

Record length exceeds available buffer space.

xx1502 RECORD LENGTHS MISMATCHED

Record length specified in FORTRAN OPEN statement does not match actual physical record length.

xx1503 END OF FILE REACHED

End of file reached, but no end path is specified in FORTRAN program.

xx1504 RECORD TYPE CONFLICT

Type of actual record is not the same as specified type (formatted vs unformatted).

xx1505 INVALID COUNT PARAMETER

Invalid count parameter specified in BACKSPACE statement.

xx1506 INPUT/OUTPUT LIST DEMANDS ARE EXCESSIVE

xx1508 UNFORMATTED WRITE WITHOUT AN IOLIST ITEM

xx1509 EXCEEDS 16 LFN'S
Input/output list demands exceed record length.

xx1521 ILLEGAL FORMAT CHARACTER

xx1522 ILLEGAL FORMAT CHARACTER SEQUENCE

xx1523 UNEQUAL NUMBER OF MATCHING PARENTHESES

xx1524 INTEGER CONSTANT MISSING
Integer constant missing from Hollerith type descriptor.

xx1525 INPUT OF HOLLERITH TYPE DATA IS ILLEGAL

xx1526 INPUT OF APOSTROPHE TYPE DATA IS ILLEGAL

xx1527 INVALID DATA

xx1528 INTEGER CONSTANT IN ERROR
Integer constant in error for X-field descriptor.

xx1529 DATA TYPE CONFLICT
Data type does not correspond with indicated data.

xx152A INTEGER WIDTH IS ZERO

xx152B LOGICAL FIELD CONTAINS BLANK CHARACTERS

xx152C LOGICAL FIELD IS NOT EITHER TRUE OR FALSE

xx1152D INTEGER VALUE TOO LARGE

xx152E ILLEGAL VALUE FOR EXPONENT

xx152F FORMAT INTEGER TOO LARGE

xx1530 TOO MANY EMBEDDED PARENTHESES

xx1531 ACCESS NOT COMPATIBLE FOR LFN

xx1541 VALUE OF EXPONENT EXCEEDS MAXIMUM RANGES

xx1542 ZERO ARGUMENT FOR INTRINSIC FUNCTION ALOG, DALOG

xx1543 NEGATIVE ARGUMENT FOR INTRINSIC FUNCTION ALOG, DALOG

xx1544 ARGUMENT TOO LARGE FOR INTRINSIC FUNCTION SIN, DSIN

xx1545 NEGATIVE ARGUMENT FOR INTRINSIC FUNCTION SQRT, DSQRT

xx1546 ORIGIN CANNOT BE AN ARGUMENT FOR $ATAN_2$, $DATAN_2$

xx1547 SECOND ARGUMENT FOR MOD FUNCTION CANNOT BE ZERO
The second argument for the intrinsic function MOD cannot be zero.

xx1548 SECOND ARGUMENT FOR AMOD, DAMOD FUNCTIONS CANNOT BE ZERO
The second argument for the intrinsic functions AMOD, DAMOD cannot be zero.

xx1549 SECOND ARGUMENT FOR ISIGN FUNCTION CANNOT BE ZERO
The second argument for the intrinsic function ISIGN cannot be zero.

xx154A SECOND ARGUMENT FOR SIGN, DSIGN FUNCTIONS CANNOT BE ZERO
The second argument for the intrinsic functions SIGN, DSIGN cannot be zero.

LOADER MESSAGES (xx16)

xx1601 ILLEGAL OVERLAY-id

xx1602 ILLEGAL PARAMETER

xx1603 INVALID LOAD ADDRESS SPECIFICATION

xx1604 INVALID START ADDRESS SPECIFICATION

xx1605 RELOCATION ERROR

xx1607 MEDIA ERROR

xx1608 SYMBOL RESOLUTION ERROR

xx1609 BOUND UNIT NOT FOUND

xx160A INSUFFICIENT MEMORY

xx160B ILLEGAL OVERLAY NESTING

xx160C OVERLAY SIZE EXCEEDS AREA SIZE

xx160D BOUND UNIT ENTRY POINT UNDEFINED

xx160E BOUND UNIT CANNOT EXECUTE IN USER TASK GROUP

xx160F BOUND UNIT CANNOT EXECUTE IN SYSTEM TASK GROUP

EC/ECL/OCL COMMAND MESSAGES (xx17)

xx1701 ILLEGAL EC DIRECTIVE

xx1702 ILLEGAL NUMBER OF ARGUMENTS

xx1703 NON-NUMERIC CHARACTER IN NUMERIC ARGUMENT

xx1704 ILLEGAL ARGUMENT LENGTH

xx1705 ILLEGAL GROUP-ID

xx1706 COMMAND NOT DEFINED IN OCL

xx1707 ARGUMENT NOT RECOGNIZED

xx1708 UNEXPECTED NULL ARGUMENT

xx1709 ILLEGAL COMBINATION OF ARGUMENTS

xx170A COMMAND NOT DEFINED IN ECL

xx170B ILLEGAL CHARACTER IN SYMBOLIC START ADDRESS (CT,ST)

xx170C SYMBOLIC START ADDRESS NOT FOUND (CT,ST)

xx170D SYMBOLIC DEVICE NAME NOT DEFINED (RAS,STS)

xx170E NO MOUNT PENDING FOR REQUESTED VOLUME OR DEVICE (RAS (-CANCEL))

xx170F NONHEX CHARACTER IN A HEX ARGUMENT

xx1710 COMMAND NOT VALID IN A COMMAND FILE FOR THE BATCH TASK GROUP

xx1711 REDUNDANT ARGUMENT

xx1712 REQUIRED ARGUMENT MISSING

xx1713 ILLEGAL LRN FOR SYMBOLIC DEVICE (RAS (-SWAP))

xx1714 DEVICE NOT OFFLINE (RAS (-SWAP))

xx1715 DEVICES TO BE SWAPPED NOT ALIKE (RAS (-SWAP))

CROSS REFERENCE PROGRAM (XREF) MESSAGES (xx18)

xx1807 ARGUMENT NOT RECOGNIZED

xx180F MEMORY FULL

No more symbols can be cross-referenced. Current symbols are processed.

xx1812 REQUIRED ARGUMENT MISSING

EDITOR MESSAGES (xx19)

Editor Initialization Messages

ERROR IN REQUEST PARAMETER

The ECL ED command recognizes only two load time parameters, -LINE LEN and -IN. The allowable line length is from 20 to 255 inclusive. -IN must be followed by a legal pathname. This message is issued if there is an error in either of these parameters. Messages generated by the editor are typed on the operator's terminal without numeric identification. The number assigned to editor messages is 19.

NO WORK SPACE ALLOCATED

The space in the current working directory is not sufficient for the temporary work files required by the editor.

CANNOT LOAD OVERLAY

- The memory available for loading the user's dynamic storage is insufficient, or
- The overlay cannot be found.

For memory space, 2K words are requested; 1K words will be accepted and used. The editor cannot run with less than 1K words of additonal storage.

Editor Addressing Messages

BUFFER EMPTY

An attempt was made to reference a specified line when the buffer is empty. (Only "$", ".", and "0" are legal addresses within an empty buffer and then only with a READ, APPEND, or INSERT directive.)

ADDRESS OUT OF BUFFER

A reference was made to a line that does not exist; for example, an address of 20 when there are less than 20 lines in the buffer, or an address of .+5 when the current line is less than five lines from the last line in the buffer.

ADDRESS WRAP-AROUND

An attempt was made to address a series of lines in which the line number of the second line addressed is less than that of the first (e.g., $,1).

SEARCH FAILED

The editor cannot find the expression specified in the directive.

ERROR IN REGULAR EXPRESSION

A regular expression used as an address is not properly delimited.

// UNDEFINED

A null expression is specified but a regular expression was not previously specified.

SYNTAX ERROR

Delimiters have been improperly used; request is not recognizable.

INVALID USE OF * IN REGULAR EXPRESSION

The asterisk has been used as the first character of a regular expression. To specify an asterisk as a data character, precede it with the characters !C; i.e., !C*. In a regular expression, the asterisk means "any number of the preceding character."

Edit Directive Messages

MODIFIED BUFFERS EXIST, QUIT DEFERRED

The contents of the buffer have been modified but have not been written to a file at the time the QUIT directive was entered. If you want to save the contents of the buffer, you must enter a WRITE directive. If not enter another QUIT to effect an exit from the editor.

NO PATHNAME GIVEN

No pathname was specified in a current READ or WRITE directive or in a previous READ or WRITE directive.

TRUNC AFTER xxxxx CHARS

The line contains more characters than the number specified in the -LIN LEN argument of the ECL ED command or in the default value (80) of that argument.

0 LEN-DEL

The line length became zero as the result of the specified substitution.

SUBSTITUTION FAILED

The editor cannot find the specified string of characters that is to be replaced.

ERROR IN COMMAND LINE

An invalid ECL command was entered in the EXECUTE directive.

Editor Messages Pertaining to Auxiliary Buffers

TOO MANY BUFFERS

An attempt was made to open a sixth buffer. The editor allows a maximum of five.

ERROR IN BUFFER NAME

A buffer name may consist of a single character or as many as 16 characters. If more than one character is used, they must be enclosed in parentheses. This message is issued if there are too many characters or if the parentheses are missing.

COPY TO CURRENT BUFFER REJECTED

A MOVE or COPY directive has specified a copy from the current buffer to the same buffer.

PATCH MESSAGES (xx21)

xx2103 ILLEGAL HEX CHARACTER

Illegal hexadecimal character specified for address or value.

xx2107 ILLEGAL INPUT PARAMETER

Correct parameter and reenter directive.

xx2112[3] NO SLASH

A slash (/) must be specified before an address field.

xx2180 SEGMENT NOT FOUND

The segment identified in the patch id cannot be found. Control returns to the Monitor.

xx2181 NO PATCH ON FILE

This message is issued when the directive Eliminate Patch (EP) or List Patches (LP) applies to a file that has never been patched. Control returns to the Monitor.

xx2182[3] DUPLICATE PATCH ID

A patch having the specified id already exists.

xx2183[3] PARAMETER TABLE OVERFLOW

Additional parameters cannot be accepted.

xx2184[3] NO ROOM TO VERIFY TAB

xx2185 ADDRESS OUT OF BOUNDS

msg Address specified in msg in not within specified segment. Control returns to the Monitor.

xx2186[3] PATCH NOT FOUND

Patch specified in Eliminate Patch (EP) directive cannot be found.

---

[3] Directive that contains error is deleted; preceding directives are executed.

xx2188[3] NO ROOM TO EXTEND SLOW LOAD SECTION
Control returns to the Monitor.

xx2189 NOT ENOUGH MEMORY IN MEMORY POOL
Control returns to the Monitor.

xx2190 PATCH EXCEEDS 256 BYTES
Control returns to the Monitor.

xx2191 WRONG FILE TYPE BEING PATCHED
Control returns to the Monitor.

COMMUNICATIONS FILE TRANSMISSION PROGRAM MESSAGES (xx22)

xx2201 ECL COMMAND OR ARGUMENT ERROR

xx2202 PARAMETER CARD ERROR DETECTED BY LEVEL 6

xx2203 WRITE ERROR DETECTED FROM IDENTIFICATION RECORD TRANSMISSION

xx2204 READ ERROR DETECTED FROM PARAMETER CARD RECEIPT

xx2205 RECORD SIZE MISCOMPARE BETWEEN PARAMETER CARD AND FILE INFORMATION BLOCK

xx2210 EOF RECORD NOT SENT BY LEVEL 66

xx2211 UNEXPECTED MESSAGE RECEIVED FROM LEVEL 66

xx2221 CONNECT FAILED

xx2280 PARAMETER CARD ERROR DETECTED BY LEVEL 66

xx2281 FILE TRUNCATED DURING RESTART

xx2282 RECORD SIZE ERROR

xx2283 BLOCK SIZE ERROR

xx2284 BLOCK SERIAL NUMBER ERROR

xx2285 LEVEL 66 UNRECOVERABLE I/O ERROR

xx2286 REMOTE ABORTS TRAN66

xx2287 UNEXPECTED ABORT

xx2288 BLOCK OVERFLOW ERROR

xx2289 LINE DISCONNECT

xx228A TRANSMISSION SEQUENCE NUMBER ERROR DETECTED IN LEVEL 66

xx22B1 TRANSMISSION SEQUENCE NUMBER ERROR DETECTED IN LEVEL 6

[3] Directive that contains error is deleted; preceding directives are executed.

xx22B2 INCORRECT MEDIA CODE FROM LEVEL 66

xx22B3 INCORRECT RECORD SEQUENCE NUMBER DETECTED IN LEVEL 6

xx22B4 FILE BUFFER OVERFLOW

MACRO PREPROCESSOR MESSAGES (xx23)

xx2307 INVALID CONTROL ARGUMENT
Invalid control argument in MACROP command.

xx230A INSUFFICIENT STARTING MEMORY
Rerun in a pool of larger size, or reinitialize increasing the size of the current pool.

xx230B INVALID -SIZE ARGUMENT
Reenter command using a valid -SIZE (1 through 64) argument.

xx230C PATHNAME IS MISSING
Pathname of unexpanded source file is missing from MACROP command.

xx230D WORK SPACE EXHAUSTED
All available work space has been exhausted.

xx230E NO ENDM STATEMENT
No ENDM statement in a library macro file or in an inline macro definition.

The following messages are uncoded:

MACROP vvrr	This message appears when the Macro Preoprocessor is turned on; where vv is the version and rr the revision.
mmmm ERR COUNT	This message appears when the Macro Preoprocessor is finished; mmmm is the number of errors.

EXPORT/IMPORT PAM FILE PROGRAM MESSAGES (xx24)

xx2404 ARGUMENT LENGTH IS ILLEGAL

xx2412 REQUIRED ARGUMENT IS MISSING

xx2413 IMPORT PATHNAME FILE IS NOT VARIABLE SEQUENTIAL

xx2414 UNRECOGNIZED DEVICE TYPE

xx2415 FILE TO BE EXPORTED IS NOT VARIABLE SEQUENTIAL

xx2421 DCB HAS BEEN DESTROYED
A device control block (DCB) must be available.

xx2422 INVALID MEMBER NAME
Specified input member cannot be found or specified output member is a duplicate.

xx2423 DCB STATUS INCORRECT

The device control block status is incorrect.

xx2424 END OF DATA SPACE

xx2426 INVALID RECORD LENGTH

xx242C FILE IS NOT PARTITIONED

DUMP EDIT (DPEDIT) ERROR MESSAGES (xx25)

Dump Edit reports fatal run-time errors to ERROR OUT through the system's Error Handler. Such errors terminate the dump procedure and cause an immediate return to the ECL Processor.

xx2503 NONNUMERIC CHARACTER IN NUMERIC ARGUMENT

This message is issued if a nonnumeric character is encountered during processing of the positional parameters within the -TO and -FROM arguments.

xx2507 ARGUMENT NOT RECOGNIZED

xx2512 REQUIRED ARGUMENT MISSING

This message is issued when the pathname of the dump file is omitted and -MEM has not been specified. It may also result from a missing argument during processing of the positional parameters within the -TO and -FROM arguments.

xx2513 ADDRESS MODE INCOMPATIBILITY

This message is issued when the address mode (SAF or LAF) of the dump file differs from the address mode of DPEDIT.

xx2514 DUMPFILE IS INCORRECT FILE TYPE

This message is issued when the external dump file is not a relative file without deletable records.

xx2515 DUMPFILE IS INCOMPLETE

This message is issued when the external dump file does not contain -1.

COBOL COMPILER MESSAGES (xx26)

xx2601 (argument) INVALID ARGUMENT

The displayed argument is not recognized as valid in the ECL for invoking COBOL. It is ignored and processing continues.

xx2602 INVALID SIZE SPECIFIED.

The -SIZE argument is in error and is ignored. This argument must be in the range 04 through 64. The compiler uses the default size.

xx2603 MISSING ARGUMENT

The required argument following -SIZE or -COUT is missing. Processing continues.

xx2604 TOO FEW ARGUMENTS

Insufficient arguments are supplied in the ECL for invoking COBOL. Compilation terminates and the compiler must be reinvoked.

xx2605 REQUESTED MEMORY NOT AVAILABLE

Less than 3K words of memory requested in the -SIZE argument are available. The compiler will use the available amount of memory.

2612nn LFN {01, 02, 03, 05, 06} COMPILER FILE PROBLEM

The value of nn represents an error detected by the File Manager. A typical value of nn is 09 (pathname not found in the file system). The compiler files are as follows:

LFN 01 - Source
LFN 02 - Listing
LFN 03 - Object
LFN 05 - Labels work file
LFN 06 - COBWRK work file

After this message, the compilation is terminated.

2616nn OVERLAY xx COMPILER LOADING PROBLEM

A loader error has occurred during loading of the compiler overlay represented by xx. The value of nn is one of the error codes produced by the loader.

The following messages are uncoded:

COBOL vvrr — This message appears when the compiler is turned on; vv is the version, rr the revision.

mmmm ERR COUNT — This message appears when the compiler is finished; mmmm is the number of errors.

MESSAGES ISSUED BY COBOL RUN-TIME ROUTINES (xx27)

xx2701 CALL ERR IN xxxxxx, or
CANCEL ERR IN xxxxxx

Overlay does not exist when COBOL program xxxxxx attempted to call or cancel it. Program execution is terminated.

xx2702 CALL ERR IN xxxxxx, or
CANCEL ERR IN xxxxxx

Overlay has been called but has not exited; attempt by COBOL program xxxxxx to call or cancel it is illegal. Program execution is terminated.

xx2703 CALL ERR IN xxxxxx

Overlay conflicts with resident overlay when COBOL program xxxxxx attempted to call it. Program execution is terminated.

xx2705 CALL ERR IN xxxxxx

Overlay loader error occurs when COBOL program xxxxxx attempts to call an overlay. Program execution is terminated.

xx2707 CALL ERR IN xxxxxx

Memory manager error occurs when COBOL program xxxxxx attempts to call an overlay. Program execution is terminated.

xx270A CONV ERR ON SOURCE LINE nnnnnn

The result of converting a numeric field to binary is either negative or exceeds 32,767. The line number of the source line where the error occurred is represented by nnnnnn. The condition can result from an identifier being converted in one of the following cases

1. PERFORM n TIMES
2. SET statement
3. Subscripting
4. Relative Key in I/O statement

SORT ERROR MESSAGES (xx31)

Error messages generated by the Sort program have the format:

(31yyzz)
message

Certain sort error messages are followed by a secondary message with the format:

(3131FF)
secondary message

313121 PARAMETER SYNTAX ERROR

Syntax error detected in ECL SORT command line or sort description. This message is followed by a secondary message. The text of the secondary message can be a phrase that identifies the error (for example, FIELD DESCRIPTION). Alternatively it can be a string of characters the first word of which indicates where the error was found; the remaining words in the string are those which Sort was unable to scan intelligently. An ellipsis (...) at the end of the string indicates that there are too many unintelligible words to be contained on one line.

A separate message is generated for each syntax error. Sort will be terminated once the full sort description has been checked.

313122 REQUIRED PARAMETER MISSING

Required parameter missing from sort description. This message is followed by a secondary message identifying the missing parameter. Sort will be terminated once the full sort description has been checked.

313123 TOO MANY PARAMETERS

An excessive number of parameters has been specified in the sort description. The message is followed by a secondary message identifying the error (for example, KEY FIELDS). Sort will be terminated once the full sort description has been checked.

313124 RECORD TOO SMALL, REC NUMBER nnnnnn

A variable-length record (identified by record number nnnnnn relative to the beginning of the file) has been read that is too short to support the specified key fields. The record is bypassed and the sort continues unless the next nine successive records read are also too short.

313125 INSUFFICIENT MEMORY

Insufficient memory available to support the size of record to be sorted. Sort is terminated.

313126 VALUE OR LENGTH INCONSISTENCY

A specified parameter value is inconsistent with another sort characteristic (for example, a key field position is outside the record). Sort will be terminated once the full sort description has been checked.

313127 WORKFILE TOO SMALL, WORK REC WRITTEN nnnnnn

Workfile space available is insufficient to complete the sort. File-relative record number nnnnnn represents the number of the last work record written.

313128 SEQUENCE ERROR

Sequence error detected during writing of output file. The out-of-sequence record is not written to output file. Output file is closed. Sort is terminated.

313129 DATA GAIN

Inconsistency detected between number of records read and number being written to output file. Excess records are not transferred to output file. Output file is closed. Sort is terminated.

313130 DATA LOSS

A loss of data is detected: fewer records are written to the output file than were read from the input file. Output file is closed. Sort is terminated.

313131 INCOMPLETE COMMENT

Incomplete comment detected in sort description; comment delimiters (/) did not occur in pairs. Sort is terminated.

Each of the following error conditions as reported by Sort occurred during an input/output operation on the file indicated in the message. For the explanation of the specific error that has occurred see the appropriate message listing under the applicable category code (yy) number. In these messages, SD refers to the sort description file specified in the control argument -IN_PATH in the ECL command SORT. In all cases, the sort is terminated.

31yyzz FILE NOT FOUND {INPUT | OUTPUT | WORK | SD}

The file indicated in the message cannot be found.

31yyzz OPEN ERROR {INPUT | OUTPUT | WORK | SD}

An error occurred during the process of opening the indicated file.

31yyzz READ ERROR {INPUT | WORK | SD}

An error occurred during the process of reading a record from the indicated file.

31yyzz WRITE ERROR {WORK / OUTPUT}

An error occurred during the process of writing a record to the indicated file.

31yyzz CLOSE ERROR {INPUT / WORK / OUTPUT / SD}

An error occurred during the process of closing the indicated file.

31yyzz SYSTEM ERROR

Sort-requested system service has not been executed successfully.

MULTILINE COMMUNICATIONS PROCESSOR (DUMCP) ERROR MESSAGES (xx34)

xx3401 INVALID OUTPUT DEVICE SPECIFIED

A device other than a local terminal or printer was specified as the output device. Respecify all of the required Linker directives, including a valid channel number in the VDEF DMPOUT directive.

xx3402 INVALID CHANNEL NUMBER

The channel number entered for block mode read is invalid. Respecify all of the required Linker directives including a valid channel number in the VDEF RDMLCP directive.

APPENDIX A

ECL AND OCL COMMANDS

The ECL and OCL commands in this appendix are arranged in alphabetic sequence by the command code. The information presented includes the command code, the command name, a character (E for ECL, O for OCL) designating the type of command, and the command format. Separate formats are given for variations of a command.

Command name, positional parameters, keywords, and arguments are separated by one or more spaces or horizontal tabs. In the SD command slashes separate year, month, and day; a colon separates minutes and seconds. Optional arguments are enclosed in brackets [ ]. The presence of an illegal keyword, or the specification of a wrong value for an argument in a command will cause the command to be rejected. In addition, some commands require a fixed number of arguments and some allow no arguments. If the number of arguments for such commands does not meet the requirements, the command is rejected. For example, a BYE command is rejected if it includes any arguments.

Name	*Function*	*ECL/ OCL*	*Format*
ABORT_BATCH	Abort Batch	O	ABORT_BATCH
ABR	Abort Batch Request	O	ABR
ACTB	Activate Batch	O	ACTB
ACTG	Activate Group	O	ACTG id
AG	Abort Group	E	AG [id]
AG	Abort Group	O	AG id
AGR	Abort Group Request	O	AGR id
ASSEM	Assembler	E	ASSEM path [-NO_OBJ -NO_LIST -LIST_ERRS -LAF -SIZE nn -COUT out_path]
ASSOC	Associate Path	E	ASSOC lfn path
BYE	Terminate Current Request	E	BYE
CB	Create Batch	O	CB phys_lvl -LRN n -LFN n
CD	Create Directory	E	CD path
CF	Create Fixed Relative File (without deletable records)	E	CF path -F_REL [-REC_SIZE n -SIZE n -INC_SIZE nnn -MAX_SIZE nnn -LFN lfn]
CF	Create Fixed Relative File (with deletable records)	E	CF path -N_REEL [-REC_SIZE n -SIZE n -INC_SIZE nnn -MAX_SIZE nnn -LFN lfn]
CF	Create Sequential File	E	CF path [-SEQ] [-CI_SIZE n -SIZE n -REC_SIZE nnn -INC_SIZE nnn -MAX_SIZE nnn -LFN lfn]
CF	Create Relative File (with variable records)	E	CF path -REL [-CI_SIZE n -SIZE n -REC_SIZE nnn-INC SIZE nnn -MAX_SIZE nnn -LFN lfn]
CF	Create Index File	E	CF path -INDEX [-CI_SIZE n -SIZE n -REC_SIZE nnn -INC_SIZE nnn -MAX_SIZE nnn -LFN lfn -KEY_SIZE nnn -FILL_PC nnn]
CG	Create ECL Group	E	CG id phys_lvl [-ECL] [-LRN n -LFN n -POOL id]

Name	*Funciton*	*ECL/ OCL*	*Format*
CG	Create ECL Group	O	CG id phys_lvl [-ECL] [-LRN n -LFN n] -POOL id
CG	Create Group	E	CG id phys lvl -EFN root [?entry] [-LRN n -LFN n -POOL id]
CG	Create Group	O	CG id phys_lvl -EFN root [?entry] [-LRN n -LFN n] -POOL id
COBOL	COBOLCompiler	E	COBOL path [-LIST_OBJ -NO_OBJ -NO_LIST -LIST_ERRS -LAF -SIZE nn -COUT out_path]
CP	Copy File	E	CP path new [-VOL -CI -VERBATIM -NWD]
CPA	Compare Volume	E	CPA path new -VOLUME [-PRINT x -NWD]
CPA	Compare File	E	CPA path new [-CI -LIMIT nn -FROM n -PRINT n -NWD]
CSD	Change System Directory	O	CSD [path] [-LIB x]
CT	Create Task	E	CT lrn rel_lvl -EFN root [?entry]
CT	Create Task (shared)	E	CT lrn rel lvl -SHARE lrn [ssa]
CV	Create Volume (bootstrap)	E	CV path -BOOT X'hhhh'
CV	Create Volume (Mdump record)	E	CV path -MDUMP nn
CV	Create Volume (format)	E	CV path -FORMAT vol-id [x]
CV	Rename Volume	E	CV path -RENAME y
CWD	Change Working Directory	E/O	CWD [path]
DB	Delete Batch	O	DB
DG	Delete Group	E	DG [id]
DG	Operator Delete Group	O	DG id
DISSOC	Dissociate Path	E	DISSOC lfn
DPEDIT	Dump Edit	E	DPEDIT path [-FROM X'xxxx' -TO X'xxxx' -NO_LOGICAL -NO_PHYSICAL -MEMORY]
EBR	Enter Batch Request	E	EBR in_path [-OUT out_path -WD path]
EBR	Operator Enter Batch Request	O	EBR user_id in_path [-OUT out_path -WD path]
EC	Execution Command	E/O	&Δcomment EC statement E/O &A attach user in E/O &D detach user_in E/O &FΔ turn off EC printing E/O &NΔ turn on EC printing E/O &PΔ message to operator E/O &QΔ terminate execution of EC file
ED	Editor	E	ED [-IN path -LINE_LEN n]
EGR	Enter Group Request	E	EGR id [-in_path -ARG -WD path -OUT out_path]
EGR	Enter ECL Group Request	E	EGR id in_path [-ARG -WD path -OUT out_path]

Name	*Function*	*ECL/ OCL*	*Format*
EGR	Operator Enter Group Request	O	EGR id user_id [in_path -ARG -WD path -OUT out_path]
EGR	Operator Enter ECL Group Request	O	EGR id user_id in_path [-ARG -WD path -OUT out_path]
ETR	Enter Task Request	E	ETR lrn [-WAIT -ARG]
EX_PAM	Export PAM File	E	EX_PAM path pam [mem$_i$]...[-R]
FD	File Dump	E	FD path [-LIMIT xx -FROM n -CI -NWD -BACK n -HEX -ALPHA]
FO	File Out	E/O	FO [path]
FORTRAN	FORTRAN Compiler	E	FORTRAN path [-LIST_OBJ -NO_OBJ -NO_LIST -LIST_ERRS -LAF -SIZE nn -COUT out_path]
IM_PAM	Import PAM file	E	IM_PAM pam new [mem$_i$]...[-R]
LINKER	Linker	E	LINKER path [-IN path -COUT out_path -SIZE nn -LAF -SLAF]
LS	List Directory Names	E	LS -DIR [-ALL -DETAIL -PN path] [name]
LS	List File Names	E	LS [-FILE] [-ALL -DETAIL -PN path] [name]
LSR	List Search Rules	E/O	LSR
LWD	List Working Directory	E/O	LWD
MACROP	Macro Preprocessor	E	MACROP path [-INCLUDE CONTROLS -MACRO_CALLS -SIZE nn]
MF	Modify File	E/O	MF -SHARE -READ
MF	Modify File	E/O	MF -SHARE -WRITE
MF	Modify File	E/O	MF -NONSHARE -READ
MF	Modify File	E/O	MF -NONSHARE -WRITE
MSG	Send Message to Operator	E	MSG "message"
MSW	Modify External Switches	E	MSW [-ON xx ... -OFF xx ...]
MSW	Turn on all External Switches	E	MSW - ALL ON
MSW	Turn off all External Switches	E	MSW -ALL OFF
MSW	Modify External Switches	O	MSW id [-ON xx ... -OFF xx ...]
MSW	Turn on All External Switches	O	MSW id -ALL ON
MSW	Turn off all External Switches	O	MSW id -ALL OFF
PATCH	Patch Object File	E	PATCH name [-IN path]
PR	Print File	E	PR path [-RELEASE -LIMIT nn -COPIES x -SPACE s -FORTRAN -FROM n -LINE_LEN nn]
RAS	Reassign	O	RAS [-SWAP dev_name1 dev_name2 -CANCEL name]
RDF	Ready Off	E/O	RDF
RDN	Ready On	E/O	RDN
RENAME	Rename File	E	RENAME old_path new_name
RL	Release file	E	RL -FILE path

Name	*Funciton*	*ECL/ OCL*	*Format*
RL	Release Directory	E	RL -DIR path
RPG	RPG Compiler	E	RPG path [-LIST_OBJ -NO_OBJ -NO_LIST -LIST_ERRS -LAF -SIZE nn -COUT out_path]
RS	Reset Map	E	RS path
SD	Set Date	O	SD 'yyyy/mm/dd/ hh[mm[:ss]]'
SG	Spawn ECL Group	E	SG id phys_lvl in_path [-ECL] [-OUT out_path -lrn n -LFN n -POOL id -WD path]
SG	Spawn Group	E	SG id phys_lvl [in_path] -EFN root [?entry] [-OUT out_path -LRN n -LFN n -POOL id -WD path ARG]
SG	Operator Spawn ECL Group	O	SG id user_id phys_lvl in_path [-ECL] [-OUT out_path -LRN n -LFN n -WD path] -POOL id
SG	Operator Spawn Group	O	SG id user_id phys_lvl [in_path] -EFN root_entry [-OUT out_path -LRN n -LFN n -WD path -ARG] -POOL id
SORT	Sort Program	E	SORT [-IN path -SIZE nn -PD]
SSPB	Suspend Batch	O	SSPB
SSPG	Suspend Group	O	SSPG id
ST	Spawn Task	E	ST rel_lvl -EFN root[?entry] [-WAIT -ARG]
ST	Spawn Task (shared)	E	ST rel_lvl -SHARE lrn [ssa] [-WAIT -ARG]
STG	Task Group Status	E	STG [-TASKS -FILES]
STG	Operator Task Group Status	O	STG id [-TASKS -FILES]
STS	System Status	O	STS [-BA -ALL -AVAIL -SYMPD nnn -DISABLED -GROUP -LBR]
TIME	Display Current Time	E/O	TIME
XREF	List Symbol Usage	E	XREF path [-COUT out_path -SIZE nn]

APPENDIX B

FILE TRANSMISSION

The file transmission (TRAN) programs provide for the transmission of files between Level 6 GCOS 6/MDT and Level 66 GCOS systems. Two programs are required: TRAN6, which runs on the Level 6 system, and TRAN66, which runs on the Level 66 system under GCOS with a 355 FNP under control of the Remote Terminal Supervisor (GRTS). The format of the transmitted file is ASCII on the Level 6 system and BCD system standard format (SSF) on the Level 66 system, with ASCII as the transmitted data medium. Files are transferred using a point-to-point (nonpolled) synchronous VIP protocol (refer to the *Series 60 (Level 66/6000) Remote Terminal Supervisor (GRTS)* manual, Order No. DD40). The TRAN6 program is invoked from the Level 6 using an ECL command. The TRAN66 program is invoked from the Level 66 using a procedure recognized by the GCOS operating system and executes as a standard GCOS direct access program.

FUNCTIONAL DESCRIPTION

The TRAN programs allow high-speed source data file transmission between the Level 6 and Level 66 systems. They perform a series of communications functions including:

- Support of Level 6 and Level 66 disk files
- Error recovery and file repositioning
- Extensive interactive analysis:
  - Parameter card validity testing
  - Data compression during transmission
  - Verification of transfer unit sequence number (TUSN)
  - Maintaining logical record sequence numbers for recovery
  - Operator notification of inconsistencies between stated and actual parameter values
- Conversion of files to or from Level 6 to Level 66 system standard format

TRAN6 establishes access connection with Level 66 through a communications procedure, and once the connection is complete, the TRAN programs initiate transfer in accordance with the calling parameters passed to the system from the Level 6 invocation.

The TRAN programs are designed to satisfy the following functions for successful data file transmission:

- Conversion and reconversion of data from Level 66 SSF to Level 6 ASCII format
- Compression of data to variable length record format
- Validity checking on all parameter card and system required arguments
- Deletion of duplicate transmission blocks using the TUSN
- Error indication for abort conditions
- Transmission activity status indication

Normal Termination

The TRAN programs use an end-of-file (EOF) designation to signify transmission of the final record of a data file. The normal termination sequence is then initiated. A message buffer indicating the successful transfer is formatted and transmitted by the Level 66 system to the Level 6 system. Upon receipt of the buffer, the status of the buffer is tested. A successful status is followed by a logical disconnect, a release of buffer and temporary file storage space, the closing of all open files, and program termination.

Abnormal Termination

An abnormal termination of the transfer of a file can be caused by inability to connect, inadequate file or buffer space, unrecoverable I/O or other transmission errors, or parameter card errors. The TRAN programs react to an abort condition by closing all Level 6 and Level 66 files to allow for transmission at a later time, and by signaling the operator of the abort status.

Error codes generated during the transmission activity indicate the type of inconsistency encountered. The messages include:

- Command error – TRAN6 detected an error in a calling argument
- Parameter card error (TRAN6) – Parameter card error detected by TRAN66
- Restart end of file error – TRAN66 reached an end of file before the restart record was encountered
- Unrecoverable I/O error – Indicates an unrecoverable hardware error on a Level 66 peripheral
- Parameter card error (TRAN66) – Parameter card error detected on the Level 66
- Unrecoverable transmission error – Excessive number of NAKs

TRAN Error Codes

The following error codes and their meanings apply to the TRAN programs.

2201	ECL command or argument error
2202	Parameter card error detected by Level 6
2203	Write error detected from identification record transmission
2204	Read error detected from parameter card receipt
2205	Record size miscompare between parameter card and file information block
2210	EOF record not sent by Level 66
2211	Unexpected message received from Level 66
2221	Connect failed
2280	Parameter card error detected by Level 66
2281	File truncated during restart
2282	Record size error
2283	Block size error
2284	Block serial number error
2285	Level 66 unrecoverable I/O error
2286	Remote aborts TRAN66
2287	Unexpected abort
2288	Block overflow error
2289	Line disconnect
228A	Transmission sequence number error detected in Level 66
22B1	Transmission sequence number error detected in Level 6
22B2	Incorrect media code from Level 66
22B3	Incorrect record sequence number detected in Level 6
22B4	File buffer overflow

EQUIPMENT REQUIREMENTS

Level 6

- Series 60 Level 6 Central Processor
- Multiline Communication Processor
- Synchronous Communication Line Adapter(s)
- Communication Resource(s)
  Data Set(s) – Bell System 201A, 201B or equivalent. Electrical characteristics: EIA standard RS232-C; Transmission mode: Synchronous, two-way alternate, nonpolled.
  Code Set – Modified ASCII, 7-bit.
  Configuration Load Manager with Communications Extensions
  Communications Subsystem with VIP softward module (ZQPVIP)
- GCOS 6/MDT Operating System with File Manager
- TRAN6 program (requires 2K memory)

Level 66

- o Series 60 Level 66 Central Processor
- o Comprehensive Operating Supervisor (GCOS)
- o TRAN66 program (requires 20K memory including buffers and file I/O)
- o Front-end Network Processor (FNP), applicable to DATANET 305 FNP, the DATANET 355 FNP and the DATANET 6600 FNP
- o Remote Terminal Supervisor(GRTS) – applicable to GRTS/305, GRTS/355 and GRTS/6600

LANGUAGE ELEMENTS

The parameter card is initiated at and received from the TRAN66 program after the logical connect between the two systems is complete. The parameter card arguments define the functions to be performed during the transmission process. The TRAN6 program assumes the responsibility of checking the validity of the arguments and initiates abort action for any illegalities.

The parameter card fields are defined in Table B-1. The Level 66 system interprets the original TRAN statement submitted by the user and the required activity boundaries are defined and transmitted to the originating TRAN program via the program card.

TABLE B-1. TRAN PARAMETER CARD

Columns	Field Name	Contents	Definition
1- 3	Direction	R/C C/R	Remote to Center (Level 6 to 66) Center to remote (Level 66 to 6)
4- 6	Type of Copy	FBS	Fixed-length records with a four-character block serial number to Series 66 Standard Format
9-12	Number of Characters per Record	nnnn	Maximum number of characters per logical record
15-18	Restart count	blank nnnn	Indicates initial file transfer: not a restart Record count for restart file transfer. First record to be transferred is number nnnn.

NOTE: All undefined columns must be blank.

PROGRAMMER PREPARATION INFORMATION

TRAN66

TRAN66 is activated by submitting the following job deck to GCOS, requesting TRAN66 execution and containing the TRAN66 parameter card described above:

```
Column 1      8       16
       $      SNUMB   vvvv
       $      IDENT   requestor, accounting information
       $      USERID  smcn$lop
       $      PROGRAM TRAN66
       $      LIMITS  ,9K
       $      DAC     XX
       $      file    RD....
       $      DATA    CG
          (TRAN parameter card)
       $      ENDJOB
```

vvvvv

Up to five-character job identifier.

smcn$lop

System master catalog name and log-on password.

file

Possible values are FILE and PRMFL.

....

Indicates additional user-supplied information. See the *Series 60 (Level 66/6000) GCOS Control Cards* manual, Order No. DD31, for more details.

The Level 66 must be monitored and the Level 6 operator informed when the job status changes to executing or swapped.

TRAN6

After the status of the TRAN66 job has changed to executing or swapped, TRAN6 is activated by an ECL procedure as described below.

The TRAN6 program is passed pointers to a request block (B4) and a parameter list (B7) at initiation of the transmission activity. The command line to be provided to the ECL processor is as follows:

TRAN LRN, pathname, direction,vvvv,record size

The command statement arguments are defined below:

LRN

Logical resource number for the communications line

pathname

External name of the file to be transmitted, as it is known to the Level 6 file system.

direction

Direction of the transmission activity as viewed from Level 6, i.e., input (I) or output (O).

vvvv

GCOS Job Identifier

record size

Standard record transfer sizes. Default value is set to 160 bytes.

OPERATING PROCEDURES

The TRAN programs are initiated by the TRAN command line as defined earlier in this appendix under "Programmer Preparation Information." Activity initiation continues with the decoding of command line arguments, the physical and logical system connects, and the parameter card transfer from Level 6. These initiating actions may be traced as follows:

- o Direction (open file) to register 5
- o File name to register B4
- o Set record size, default at 160 characters, register B5
- o Set parse parameter list, register B7
- o LRN, physical connect of Level 6 to Level 66
- o Transmit Level 66 logical connect of Level 6 to TRAN66. Logical connect command line is:

$*$ID A$B,22L, vvvvv

ID

GCOS terminal identification

A$B

Default GCOS userid/password entry

22L

GRTS terminal type log-on code

vvvv
GCOS job ID, entered at time of TRAN6 activation (see "Programmer Preparation Information").

- o Parameter card received from Level 66
- o Parameters validated and transmission activity initiated

The system operator must mount the disk volumes which include the files to be transmitted by the TRAN program. If a switched network is used for the file transfer operation, the switched line connection must be completed prior to initiation of the transmission.

Each complete transmission activity consists of the transfer of one entire file from Level 6 to Level 66 or the reverse direction. The logical records are transmitted in sequence until the entire file has been transferred.

DETAIL INTERFACE TO TRAN66

Transmission Unit (TU)

A TU is the physical block transferred on the communication line. It is 320 characters long; it contains one or more logical records or record parts. Each TU is uniquely identified by a TU sequence number (TUSN). The TU sequence number increases sequentially from 040 to 137 and then restarts sequencing at 040.

After the connection is established and the parameter card has been verified, each block transferred has a TUSN attached.

TU Format

TUSN	One character
Data record(s)	n (319) characters

Data Record Format

Media Code	One character
Logical record seq. no.	Four characters (ASCII)
Data	n characters
Record Separator	One character

A data record may span TUs or several data records may be in one TU.

End of File (EOF) Indication

After the EOF record(s) have been transmitted, the Level 66 transmits the termination message.

Level 6 To Level 66

The EOF record follows the last record of valid data. A separate EOF record, contained in one TU, must be sent to complete the transmission.

Level 66 To Level 6

The EOF record follows the last record of valid data.

EOF Record Format
- Media Code (58_{16} – indicates a data record)
- EOF Code ($3F_{16}$)
- Record Separator ($1E_{16}$)

Character Values:

Media Code (data records)	$58_{16} = 130_8$
Media Code (control records)	$5E_{16} = 136_8$
Unit Separator	$1F_{16} = 37_8$
Record Separator	$1E_{16} = 36_8$
EOF Code	$3F_{16} = 77_8$

Logical record sequence numbers are composed of numeric ASCII characters (060 through 071) and increase numerically from 0001 to 9999 in ASCII before wrapping around. The sequence numbers are transmitted from the host in compressed format; e.g., sequence number 0001 is transmitted as 060, 037, 063, 061 (see "Data Compaction" below).

Data Compaction

Compaction is performed to relieve traffic on the communication line. When one character is repeated more than three times, TRAN6 expands data received from the center and compacts the data that it sends.

In TRAN6 operation, when a character is repeated from 3 to 63 (inclusive) times, the following string is substituted for the series:

character to be repeated
unit separator
count (string length, encoded)

The count is encoded in Table B-2.

The first four characters of the data field can be decoded to determine the messages content and, therefore, the termination status as follows:

Successful Transmission 0000	SUCCESSFUL TERMINATION
Parameter card error detected in TRAN66 2280	PARAMETER CARD ERROR
File truncated during restart 2281	FILE TRUNCATED (EOT)
Record size error 2282	RECORD SIZE ERROR
Block size error 2283	BLOCK SIZE ERROR
Block Serial Number error 2284	BLOCK SERIAL NUMBER ERROR
Level 66 unrecoverable I/O error (e.g., disk) 2285	LEVEL 66 UNRECOVERABLE I/O ERROR
Remote Aborts TRAN66 2286	REMOTE ABORT TRAN66
Unexpected abort (e.g., time limit exceeded) 2287	UNEXPECTED ABORT
Block Overflow error 2288	BLOCK OVERFLOW ERROR
Line Disconnected 2289	LINE DISCONNECTS
Transmission sequence number error detected 228A	TRANSMISSION SEQUENCE NUMBER ERROR

TABLE B-2. STANDARD CHARACTER SET

Standard Character Set	Internal Machine Code (Binary)	BCD (Octal)	ASCII (Octal)	Standard Character Set	Internal Machine Code (Binary)	BCD (Octal)	ASCII (Octal)
0	000000	00	060	↑	100000	40	136
1	000001	01	061	J	100001	41	112
2	000010	02	062	K	100010	42	113
3	000011	03	063	L	100011	43	114
4	000100	04	064	M	100100	44	115
5	000101	05	065	N	100101	45	116
6	000110	06	066	O	100110	46	117
7	000111	07	067	P	100111	47	120
8	001000	10	070	Q	101000	50	121
9	001001	11	071	R	101001	51	122
[	001010	12	133	-	101010	52	055
#	001011	13	043	$	101011	53	044
@	001100	14	100	*	101100	54	052
:	001101	15	072	)	101101	55	051
>	001110	16	076	;	101110	56	073
?	001111	17	077	'	101111	57	047
b	010000	20	040	+	110000	60	053
A	010001	21	101	/	110001	61	057
B	010010	22	102	S	110010	62	123
C	010011	23	103	T	110011	63	124
D	010100	24	104	U	110100	64	125
E	010101	25	105	V	110101	65	126
F	010110	26	106	W	110110	66	127
G	010111	27	107	X	110111	67	130
H	011000	30	110	Y	111000	70	131
I	011001	31	111	Z	111001	71	132
&	011010	32	046	←	111010	72	137
.	011011	33	056	,	111011	73	054
]	011100	34	135	%	111100	74	045
(	011101	35	050	=	111101	75	075
<	011110	36	074	"	111110	76	042
\	011111	37	134	!	111111	77	041

Termination Messages

TRAN66

Termination Messages are identified by a media code of 136_8 and have the following format:

Media code
Data
Record Separator

GRTS

GRTS termination messages are listed in the Remote Terminal Supervisor (GRTS) manual.

APPENDIX C

ASCII AND EBCDIC CHARACTER SET

Tables C-1 and C-2 illustrate the ASCII and EBCDIC character sets, respectively. In addition to the ASCII characters, Table C-1 shows the hexadecimal equivalents; Table C-2 shows the binary and hexadecimal equivalents of theEBCDIC character set.

Following are lists of the control characters and special graphic characters that appear in the two tables:

Control Characters

ACK	Acknowledge	EOT	End of Transmission	PF	Punch Off
BEL	Bell	ESC	Escape	PN	Punch On
BS	Backspace	ETB	End of Transmission Block	RES	Restore
BYP	Backspace	ETX	End of Text	RLF	Reverse Line Feed
CAN	Cancel	FF	Form Feed	RS	Reader Stop
CC	Cursor Control	FS	Field Separator	SI	Shift In
CR	Carriage Return	GE	Graphic Escape	SM	Set Mode
CU1	Customer Use 1	GS	Group Separator	SMM	Start of Manual Message
CU2	Customer Use 2	HT	Horizontal Tab	SO	Shift Out
CU3	Customer Use 3	IFS	Interchange File Separator	SOH	Start of Heading
DC1	Device Control 1	IGS	Interchange Group Separator	SOS	Start of Significance
DC2	Device Control 2	IL	Idle	SP	Space
DC3	Device Control 3	IRS	Interchange Record Separator	STX	Start of Text
DC4	Device Control 4	IUS	Interchange Unit Separator	SUB	Substitute
DEL	Delete	LC	Lower Case	SYN	Synchronous Idle
DLE	Data Link Escape	LF	Line Feed	TM	Tape Mark
DS	Digit Select	NAK	Negative Acknowledgement	UC	Upper Case
EM	End of Medium	NL	New Line	US	Unit Separator
ENQ	Enquiry	NUL	Null	VT	Vertical Tab
EO	Eight Ones				

Special Graphic Characters

¢	Cent Sign	?	Question Mark
.	Period, Decimal Point	`	Grave Accent
<	Less-than Sign	:	Colon
(	Left Parenthesis	#	Number Sign
+	Plus Sign	@	At Sign
¦	Logical OR	'	Prime, Apostrophe
&	Ampersand	=	Equal Sign
!	Exclamation Point	"	Quotation Mark
$	Dollar Sign	≈	Tilde
*	Asterisk	{	Opening Brace
)	Right Parenthesis	⌐	Hook
;	Semicolon	⊤	Fork
¬	Logical NOT	}	Closing Brace
-	Minus Sign	\	Reverse Slant
/	Slash	⊣	Chair
|	Vertical Line	∣	Long Vertical Mark
,	Comma	[	Opening Bracket
%	Percent	]	Closing Bracket
—	Underscore	^	Circumflex
>	Greater-than Sign		

TABLE C-1. ASCII/HEXADECIMAL EQUIVALENTS

	H1							
H2	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
0	NUL	DLE	SP	0	@	P	`	p
1	SOH	DC1	!	1	A	Q	a	q
2	STX	DC2	"	2	B	R	b	r
3	ETX	DC3	#	3	C	S	c	s
4	EOT	DC4	$	4	D	T	d	t
5	ENQ	NAK	%	5	E	U	e	u
6	ACK	SYN	&	6	F	V	f	v
7	BEL	ETB	'	7	G	W	g	w
8	BS	CAN	(	8	H	X	h	x
9	HT	EM	)	9	I	Y	i	y
A	LF	SUB	*	:	J	Z	j	z
B	VT	ESC	+	;	K	[	k	{
C	FF	FS	,	<	L	\	l	|
D	CR	GS	-	=	M	]	m	}
E	SO	RS	.	>	N	^	n	~
F	SI	US	/	?	O	_	o	DEL

TABLE C-2. EBCDIC/HEXADECIMAL/BINARY EQUIVALENTS

Bit Positions 4, 5, 6, 7	Second Hexadecimal Digit	00				01				10				11				Bit Positions 0,1
		00	01	10	11	00	01	10	11	00	01	10	11	00	01	10	11	Bit Positions 2,3
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	First Hexadecimal Digit
0000	0	NUL	DLE	DS		SP	&	-						{[a]	}[a]	\[a]	0	
0001	1	SOH	DC1	SOS				/		a	i	~[a]		A	J		1	
0010	2	STX	DC2	FS	SYN					b	k	s		B	K	S	2	
0011	3	ETX	TM							c	l	t		C	L	T	3	
0100	4	PF	RES	BYP	PN					d	m	u		D	M	U	4	
0101	5	HT	NL	LF	RS					e	n	v		E	N	V	5	
0110	6	LC	BS	ETB	UC					f	o	w		F	O	W	6	
0111	7	DEL	IL	ESC	EOT					g	p	x		G	P	X	7	
1000	8	GE[a]	CAN							h	q	y		H	Q	Y	8	
1001	9	RLF[a]	EM						\[a]	i	r	z		I	R	Z	9	
1010	A	SMM	CC	SM		¢	!	;[a]	:								][a]	
1011	B	VT	CU1[a]	CU2[a]	CU3[a]	.	$	,	#									
1100	C	FF	IFS		DC4	<	*	%	@					⌠[a]		⌡[a]		
1101	D	CR	IGS	ENQ	NAK	(	)	_	'									
1110	E	SO	IRS	ACK		+	;	>	=					≠[a]				
1111	F	SI	IUS	BEL	SUB	¦	¬	?	"								EO[a]	

[a]This character is not supported in the 2780 character set.

HONEYWELL INFORMATION SYSTEMS

Technical Publications Remarks Form

TITLE: SERIES 60 (LEVEL 6) GCOS 6/MDT SYSTEM CONTROL

ORDER NO. AX07, REV. 0

DATED MARCH 1977

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 _____

_____

.EASE FOLD AND TAPE –
OTE: 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

Honeywell

Honeywell

Honeywell Information Systems
In the U.S.A.: 200 Smith Street, MS 486, Waltham, Massachusetts 02154
In Canada: 2025 Sheppard Avenue East, Willowdale, Ontario M2J 1W5
In Mexico: Avenida Nuevo Leon 250, Mexico 11, D.F.

17819, 2577, Printed in U.S.A.

AX07, Rev. 0