

# Honeywell



**LEVEL 6**

SOFTWARE

**GCOS6 MOD 400  
OPERATOR'S  
GUIDE**

SERIES 60 (LEVEL 6)  
GCOS 6 MOD 400 OPERATOR'S GUIDE

SUBJECT

Operator's Procedures and Commands for the Series 60 (Level 6) GCOS 6 MOD 400 Operating System

SOFTWARE SUPPORTED

This manual supports Release 0100 of the Series 60 (Level 6) GCOS 6 MOD 400 Operating System. See the Manual Directory of the latest *GCOS 6 MOD 400 System Concepts* manual (Order No. CB20) for information as to later releases supported by this manual.

ORDER NUMBER

CB24, Rev. 0

January 1978

**Honeywell**

# *Preface*

This manual defines the procedures and commands used to operate the GCOS 6 MOD 400 Operating System from the operator terminal.

Unless otherwise stated, the term GCOS refers to the GCOS 6 MOD 400 software. The term Level 6 refers to the Series 60 (Level 6) hardware on which the software executes.

Section 1 is a greatly condensed overview of GCOS software elements and structures that are defined in detail in the *System Concepts* manual. It is included as refresher information; it is not designed to describe all aspects of the system.

Section 2 briefly describes daily routine system startup procedures, assuming that the system has already been configured and initialized, and also discusses activation of the listener component.

Section 3 describes certain aspects of system/operator communication with the operator terminal and through the software Operator Interface Manager.

Section 4 is a detailed description of the formats and functions of operator commands used in system control.

Section 5 discusses the task interrupt (break) function.

Appendix A is a detailed discussion of command line arguments that might be used in situations more complex than those described in earlier sections.

Appendix B describes the systems special listener and login component, which is used to provide system monitoring over other terminals.

Appendix C describes system halts that occur when there is not enough memory for the system, or that might occur at system startup.

Appendix D lists the ASCII character set, with equivalent hexadecimal values.

## MANUAL DIRECTORY

The following publications comprise the GCOS 6 manual set. The Manual Directory in the latest GCOS 6 MOD 400 System Concepts manual lists the current revision number and addenda (if any) for each manual in the set.

<i>Order No.</i>	<i>Manual Title</i>
CB01	<i>GCOS 6 Program Preparation</i>
CB02	<i>GCOS 6 Commands</i>
CB03	<i>GCOS 6 Communications Processing</i>
CB04	<i>GCOS 6 Sort/Merge</i>
CB05	<i>GCOS 6 Data File Organizations and Formats</i>
CB06	<i>GCOS 6 System Messages</i>
CB07	<i>GCOS 6 Assembly Language Reference</i>
CB08	<i>GCOS 6 System Service Macro Calls</i>
CB09	<i>GCOS 6 RPG Reference</i>
CB10	<i>GCOS 6 Intermediate COBOL Reference</i>
CB20	<i>GCOS 6 MOD 400 System Concepts</i>
CB21	<i>GCOS 6 MOD 400 Program Execution and Checkout</i>
CB22	<i>GCOS 6 MOD 400 Programmer's Guide</i>
CB23	<i>GCOS 6 MOD 400 System Building</i>
CB24	<i>GCOS 6 MOD 400 Operator's Guide</i>
CB25	<i>GCOS 6 MOD 400 FORTRAN Reference</i>
CB26	<i>GCOS 6 MOD 400 Entry-Level COBOL Reference</i>
CB30	<i>Remote Batch Facility User's Guide</i>
CB31	<i>Data Entry Facility User's Guide</i>
CB33	<i>Level 6/Level 6 File Transmission</i>
CB34	<i>Level 6/Level 62 File Transmission</i>
CB35	<i>Level 6/Level 64 (Release 0300) File Transmission</i>
CB36	<i>Level 6/Level 66 File Transmission</i>
CB37	<i>Level 6/Series 200/2000 File Transmission</i>
CB38	<i>Level 6/BSC 2780 File Transmission</i>
CB39	<i>Level 6/Level 64 (Release 0220) File Transmission</i>

In addition, the following documents provide general hardware information:

<i>Order No.</i>	<i>Manual Title</i>
AS22	<i>Honeywell Level 6 Minicomputer Handbook</i>
AT04	<i>Level 6 System and Peripherals Operation Manual</i>



.

.



.

.





# Contents

## Section 1. Introduction to GCOS 6 MOD 400 Software System

Task Group .....	1-1
Task Group Identification (Group Id) ...	1-1
User Identification (User_Id) .....	1-1
LFN and LRN Maximum Values .....	1-2
File System Pathnames .....	1-2
Definition of a File .....	1-2
Definition of a Directory .....	1-2
Directory or File Name Construction ...	1-2
Pathname Construction .....	1-2
Device Pathname .....	1-3
Device Files (Other Than Disk and Tape) .....	1-3
Tape Files .....	1-3
Disk Device Files .....	1-3
Device Pathname Examples .....	1-4
Absolute Pathname .....	1-4
Relative Pathname and Working Directory .....	1-4

## Section 2. Routine System Startup and Login Activation

Routine System Startup .....	2-1
System Startup Response .....	2-2
Activating Login Capability through Listener .....	2-2

## Section 3. System Operator Interface with the System

System Operator and Operator Terminal ..	3-1
Operator Terminal .....	3-1
System Operator .....	3-1
Operator Interface Manager (OIM) .....	3-1
Output Message .....	3-2
Output Message Queuing .....	3-2
Output Message Format and Length ..	3-2
Input Message .....	3-2
Input Response Message .....	3-2
Response to Task Group Message Requests .....	3-3
Input Message Format and Length ...	3-3
Input Message Control .....	3-4
Input Directive Message to OIM .....	3-5
Outstanding Output Message List ..	3-5
Change Default Task Group .....	3-5
Change Output Pacing Rate .....	3-5
Task Interrupt (Break) from Operator Terminal .....	3-5
Error Messages Issued by OIM .....	3-5
Sample Dialog Between System Operator and the System .....	3-6

Increasing Response Time for Operator Command Execution .....	3-6
Unit Record Device Timeout .....	3-6

## Section 4. Operator Commands

Operator Commands and Commands .....	4-1
Operator Commands .....	4-1
Operator Commands for Execution Control .....	4-1
Operator Commands for Directory, File, and Device Control .....	4-1
Operator Commands to Monitor the System .....	4-2
Commands .....	4-2
Command Processor Standard Input/Output Files .....	4-2
Command-In File .....	4-2
Operator-Out File .....	4-2
Error-Out File .....	4-2
User-Out File .....	4-2
Concurrency of Standard I/O Files .....	4-3
Input Command Line .....	4-3
Command Line Format .....	4-3
Argument .....	4-3
Positional Argument .....	4-3
Keyword Argument .....	4-3
Control Argument .....	4-3
Spaces in Command Lines .....	4-4
Operator Command Formats and Descriptions .....	4-4
ABORT BATCH .....	4-6
ABORT BATCH REQUEST .....	4-7
ABORT GROUP .....	4-8
ABORT GROUP REQUEST .....	4-9
ACTIVATE BATCH .....	4-10
ACTIVATE GROUP .....	4-11
CHANGE SYSTEM DIRECTORY .....	4-12
CHANGE WORKING DIRECTORY .....	4-13
CREATE BATCH .....	4-14
CREATE GROUP .....	4-15
DELETE BATCH .....	4-17
DELETE GROUP .....	4-18
ENTER BATCH REQUEST .....	4-19
ENTER GROUP REQUEST .....	4-20
EXECUTION COMMAND .....	4-22
FILE OUT .....	4-25
LIST SEARCH RULES .....	4-26
LIST WORKING DIRECTORY .....	4-27
LOAD SHARABLE BOUND UNIT .....	4-28
MODIFY EXTERNAL SWITCHES .....	4-29
MODIFY FILE .....	4-30
REASSIGN .....	4-31

SET DATE .....	4-32
SPAWN GROUP .....	4-33
STATUS GROUP .....	4-35
STATUS SYSTEM .....	4-37
SUSPEND BATCH .....	4-39
SUSPEND GROUP .....	4-40
TIME .....	4-41
UNLOAD SHARABLE BOUND UNIT ..	4-42
WRITABLE CONTROL STORE (WSC)	
LOAD .....	4-43

**Section 5. Task Interrupt (Break) from Operator Terminal**

Break Function Usage .....	5-1
Break Procedures .....	5-1
UW and PI Commands in User Application Programs .....	5-2
Break Command Examples .....	5-2

**Appendix A. Additional Command Line Arguments**

Argument Passing .....	A-1
Input Command Line Parameter Substitution .....	A-1
EC File Execution Command .....	A-2
Group Request Commands .....	A-3

**Appendix B. Listener Component and Login Capability**

Installing a System Login Capability .....	B-1
Memory Pools for Login Tasks .....	B-1
Terminal Login Characteristic File .....	B-1
G-Record in Login File .....	B-1
T-Record in Login File .....	B-2
A-Record in Login File .....	B-2
Listener Activation .....	B-2
Designing the Login Terminal File .....	B-3
Terminal State after Listener is Activated .....	B-3

Noncommunications Terminal State With Listener .....	B-3
Communications Terminal State with Listener .....	B-4
Changing Login Message of the Day ....	B-4

**Appendix C. System Halts**

Insufficient Memory Halts .....	C-1
Startup Halts .....	C-1
Bootstrap Halt Conditions .....	C-1
Initialization Halt Conditions .....	C-1

**Appendix D. ASCII Character Set and Hexadecimal Equivalents**

ASCII Control Characters .....	D-1
ASCII Special Characters .....	D-1

**Figures**

2-1 Stages of System Configuration and Startup .....	2-1
3-1 Sample Operator/System Dialog ...	3-7

**Tables**

3-1 Input Message Format and Use ....	3-3
3-2 Length of Input Message by Device Type .....	3-3
3-3 Input Message Control Format ....	3-4
3-4 Error Messages Issued by Operator Interface Manager (OIM) .....	3-6
4-1 Operator Commands — Function and Command Names .....	4-4
5-1 System Programs Supporting UW (Unwind) Command .....	5-2
D-1 ASCII/Hexadecimal Equivalents ...	D-2

# Section 1

## Introduction to GCOS 6 MOD 400 Software System

This section is a brief introduction to certain aspects of the GCOS MOD 400 Operating System. The manual *System Concepts* describes the system, its components, and its unique features in greater detail, and is necessary for a more complete understanding of system terms, structure, and elements that are only referred to in this Operator's Guide.

### TASK GROUP

A task group is a named set of one or more tasks that share the same set of system resources and is the framework within which all user applications and software service functions operate.

### TASK GROUP IDENTIFICATION (*Group Id*)

Every task group has its own unique identifier or *group id*. For Honeywell-supplied task groups and for the system task group, the group id is \$S; for the batch task group it is \$B. Commands may include the group id to indicate which task group is to be acted on; in other commands the group id is either implied or is the default group id. Operator and user communication with task groups may use the group id in responding to the task group's messages.

An online task group identifier can be any two alphabetic and/or numeric characters, e.g., A5, 2B, AA, 32, etc.

### USER IDENTIFICATION (*User Id*)

A user identification or *user\_id* uniquely identifies the user for whom the task group is currently active, and is included in some operator commands for activating a task group when indicated as a valid argument.

The *user\_id* field comprises three elements or subfields, separated by periods, with no intervening spaces, and terminated by a space, as for example:

person.account.modeΔ

person

Name of a person who may access the system. It has from 1 to 12 characters.

account

Name of an account to which work is charged. It has from 1 to 12 characters.

mode

Identifies the unique operating characteristic (mode) of this task group, i.e., interactive, batch, etc., and helps distinguish a user's output when the same task group is run more than once. It has from one to three characters.

The *user\_id* cannot exceed a total of 29 characters, including the period separators. It can contain only uppercase letters, numbers, the underscore (\_), and the dollar sign (\$).

A *user\_id* remains the same for the duration of the group request.

The *user\_id* may contain all three subfields, only the first two, or only the first, as for example:

WRIT\_JR\$.DOCUM01.ABSΔ

WRIT\_JR\$.DOCUM01Δ

WRIT\_JR\$Δ



## **LFN AND LRN MAXIMUM VALUES**

The logical file number (LFN) is associated with file pathnames within tasks through the ASSOCIATE PATH or GET FILE command (see the *Commands* manual). Its value is from 0 through 255.

The logical resource number (LRN) is used for one task to communicate with another task in the same task group or with a system device driver. Its value is from 0 through 252.

## **FILE SYSTEM PATHNAMES**

The File System is a tree-structured hierarchy through which each volume of storage is identified to the system. The basic element of this structure is the *file*. A special file called the *directory* contains information about other files.

## **DEFINITION OF A FILE**

A file is any unit of storage outside the central processor, which can supply data to or receive data from a task. A file can be a peripheral device such as a printer, card reader, or terminal; or it can be a collection of data stored within a directory structure on a magnetic (tape or disk) 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.

## **DEFINITION OF A DIRECTORY**

A directory is a file that contains information about other "subordinate" storage system entries, which in turn may represent other directories or data files. An entry named in a directory is subordinate to that directory, and is "contained" within it. The information in the containing directory describes physical and logical attributes of the subordinate files.

The directory at the base of a tree structure is the *root directory*. Its name is the same as the name (volume id) of the volume where it resides.

When first created, a volume has only a root directory, within which names and attributes of subordinate directories can be created later.

All references to directories and files begin either explicitly or implicitly with a root directory name.

## **DIRECTORY OR FILE NAME CONSTRUCTION**

A directory or file name can consist of the following ASCII characters:

- Uppercase letters (A through Z)
- Decimal digits (0 through 9)
- Underscore character (\_)
- Period (.)
- Dollar sign (\$).

Any name must begin with a letter or the dollar sign (\$). The underscore is used to join two or more words that the system is to interpret as one name, e.g., DATE\_TIME. The period separates a name from its alphabetic or numeric suffix characters. For example, in the name of a COBOL source file called cobprog.C, cobprog is any user-specified name, and C is the suffix, indicating to the system that this is a COBOL source file.

The length of a *root directory* name or volume identifier can be from one (nonblank) to six characters. A directory (other than root) or file can have from one (nonblank) to twelve characters. A specified file name must provide for any possible suffix that might be appended by the system so that its resultant overall length does not exceed 12 characters.

## **PATHNAME CONSTRUCTION**

A pathname is a string comprising one or more directory names and possibly one file name. All subordinate names of directories or files within a directory must be unique. The pathname describes

the access path to the entity to be acted on. A pathname begins with a root directory name, followed by none or more directory names and possibly a file name, in order of their hierarchy.

The progressive relationship among pathname elements in the hierarchy is indicated by the following symbols:

- Circumflex (^) — Denotes a *root* directory only, and must precede the root directory name, with *no* intervening space (e.g., ^VOL011).
- Greater than symbol (>) — Indicates movement in the hierarchy *away from* the root, and connects two directory names or a directory name and a file name. It can also be the first character in a pathname, in which case it is immediately subordinate to the root directory of the system volume.

Each successive symbol in the string indicates a change of one directory level; the name immediately following the symbol is at the next subordinate level to the name immediately preceding it. Reading a pathname from left to right shows the access through the tree structure, away from the root, to the last element in the pathname. For example, if the root directory VOL011 contains the directory name DIR1, then the pathname for DIR1 is ^VOL011>DIR1. However, if directory DIR1 in turn contains the file FILEA, then the pathname for FILEA is ^VOL011>DIR1>FILEA. The symbol is never followed by a space, nor preceded by a space *except* as the first character in a pathname.

- Less than symbol (<) — Indicates movement in the hierarchy *toward* the root, and a change of one level in that direction. Additional < symbols show successive level changes.

The last element in a pathname is the name of the entity that is to be acted on, and may denote either a directory name or file name, according to the action to be done.

Total length of any pathname, including all hierarchical symbols, cannot exceed 58 characters, except that a working directory pathname cannot exceed 44 characters.

## DEVICE PATHNAMES

Reference to any device is through the Symbolic Peripheral Device (SPD) directory, which is subordinate to the system root.

### DEVICE FILES (Other than Disk and Tape)

The general form of a device file pathname is:

```
>SPD>dev_name
```

where dev\_name is the symbolic name defined for the card reader, punch, printer, or terminal device during system building.

Device files are always reserved for exclusive use (i.e., the reserving task group has read and write access but other users are not allowed to share the file).

### TAPE FILES

The general form of a tape file (device) pathname is:

```
>SPD>dev_name[>volid[>filename]]
```

where dev\_name is the symbolic name defined for the tape device during system building, volid is the name of the tape volume, and filename is the name of the file on the volume.

Tape devices are always reserved for exclusive use (i.e., the reserving task group has read and write access but other users are not allowed to share the file).

### DISK DEVICE FILES

The general form of a disk device-level access pathname is:

```
>SPD>dev_name[>volid]
```

where dev\_name is the symbolic name defined for the disk device during system building and volid is the name of the disk volume.

This pathname format is used only when access to the entire volume is required (such as during a volume copy or a volume dump).

If the volid is not supplied, reservation of the disk is exclusive (i.e., the reserving task group has read and write access but other users are not allowed to share the file). This pathname form is used when creating a new volume.

If the volid is specified, reservation is read/share (i.e., the reserving task group has read access only; other users may read and write). This pathname is used when dumping select portions of a volume without regard to the hierarchical file system tree structure.

The following are examples of device pathnames.

#### DEVICE PATHNAME EXAMPLES

Peripheral Device	Pathname
Line printer	>SPD>LPT01
Exclusive tape volume	>SPD>MT902>VOL3
File on an exclusive tape volume	>SPD>MT902>VOL3>FILEA
Exclusive diskette	>SPD>DSK02
Nonexclusive cartridge disk volume	>SPD>RCD01>V23X

#### ABSOLUTE PATHNAME

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

#### RELATIVE PATHNAME AND WORKING DIRECTORY

A *relative pathname* is one that does *not* begin with the circumflex or greater-than symbol. For a relative pathname that does not begin with a less-than symbol, the first (or only) name in the pathname identifies a directory or file immediately subordinate to a directory known as the *working directory*. The working directory is the user's current working position in the file system hierarchy.

The simplest form of a relative pathname has only one element, the name of the desired entry in the working directory.

The following are examples of relative pathnames and the full pathnames they represent when the working directory pathname is

>UDD>PROJ1>USERA

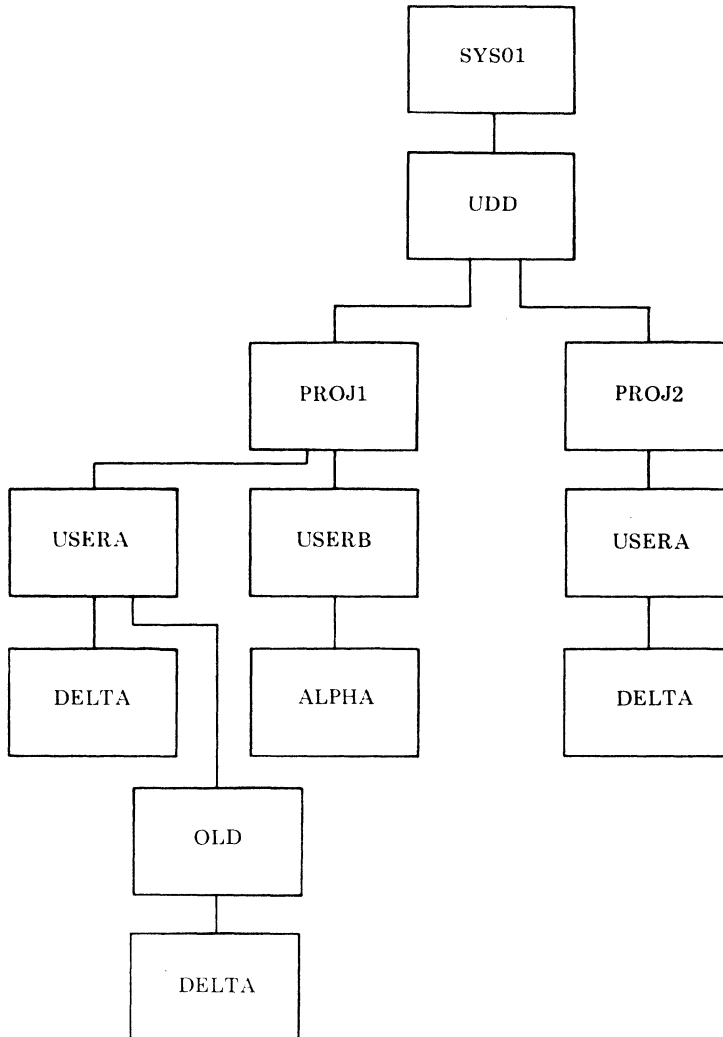
and the system was initialized from the volume SYS01.

*Relative Pathname*

DELTA  
OLD>DELTA  
<USERB>ALPHA  
<<PROJ2>USERA>DELTA  
<

*Full Pathname*

^ SYS01>UDD>PROJ1>USERA>DELTA  
^ SYS01>UDD>PROJ1>USERA>OLD>DELTA  
^ SYS01>UDD>PROJ1>USERB>ALPHA  
^ SYS01>UDD>PROJ2>USERA>DELTA  
^ SYS01>UDD>PROJ1





# Routine System Startup and Login Activation

System configuration and startup includes the four stages shown in Figure 2-1. The first three stages concern configuration and startup at system installation; the *System Building* manual describes them in detail. This section discusses the fourth stage, which is the daily routine system startup in normal operations. It also discusses activation of the system listener component for login commands.

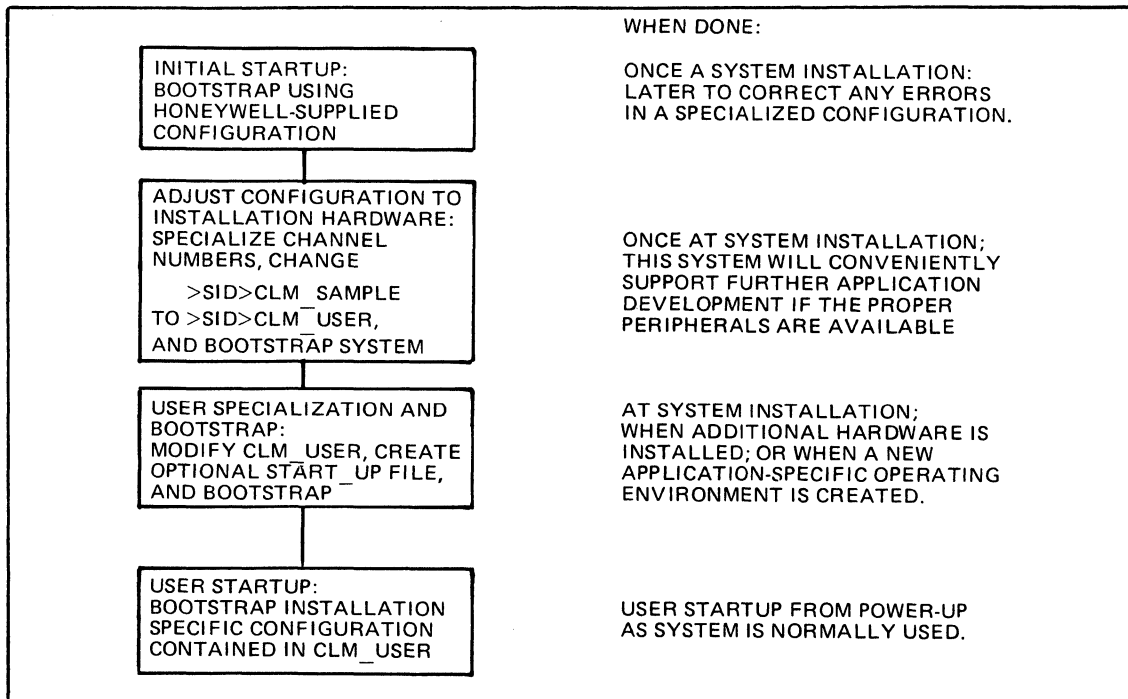


Figure 2-1. Stages of System Configuration and Startup

### ROUTINE SYSTEM STARTUP

Specific operating procedures may differ among various installations. However, some parts of the routine daily system startup are the same for all systems. The following procedures are based on the assumptions that (1) a user-specialized system startup configuration is available, and (2) the central processor and all peripheral devices were previously turned off.

1. Turn the power on for the central processor and all peripheral devices.
2. Mount the disk volume containing the specialized CLM\_USER and START\_UP.EC files, onto the bootstrap disk device.
3. Press the following keys on the central processor control panel; this starts the hardware quality logic test (QLT):
  - Stop
  - CLeaR
  - Load
  - Execute
4. Wait for the TRAFFIC light to go out, then press Execute to execute the bootstrap routine.



## SYSTEM STARTUP RESPONSE

When the system software is ready for use, the operator terminal displays the message:

```
($$) GCOS6 MOD400- { S } rrr-mm/dd/hh/mm  
                   { L }
```

followed by user-specialized configuration messages. The actual value for rrr in the message is the 3-digit software release number (e.g., 100, 110, 210, etc.) for this system. The month, day, hour, and minute that the system was linked are indicated by mm/dd/hh/mm.

## ACTIVATING LOGIN CAPABILITY AND LISTENER

To provide access to the system from user-designated terminals requires that the listener component be activated from the operator terminal as the lead task of a task group. Listener can be activated only after system startup is complete and the system is operational. None of the terminals to be monitored for a LOGIN command can be reserved during listener activation. Once activated, listener cannot be turned off until the system is again started up.

Listener is activated with the CG (CREATE GROUP) and EGR (ENTER GROUP REQUEST) operator commands, or with an SG (SPAWN GROUP) operator command, using the arguments shown below in addition to those described for those commands in Section 4. These command formats are:

```
CG id base_lvl -EFN LISTENER -POOL id  
EGR id user_id -OUT >SPD>CONSOLE -ARG [ { 'pathΔ' }  
                                         { "pathΔ" } ] [x] ["message"]  
or  
SG id user_id base_lvl -EFN LISTENER -POOL id -OUT >SPD>CONSOLE -ARG [ { 'pathΔ' }  
                                         { "pathΔ" } ] [x] ["message"]
```

```
{ 'pathΔ' }  
{ "pathΔ" }
```

Pathname of the terminals file, which lists the terminals on which users may log in, and which contains the terminal characteristics records (see Appendix B). The last character in the pathname must be a blank, and the entire pathname must be enclosed in either single or double quotes. An omitted (default) pathname must be written as a pair of enclosing single or double quotes (' ') or (" "), and results in the default pathname >SID>TERMINALS for use by listener.

x

The first character in the 2-character pool id and group id when default values are used. The second character, from 0 through 9 or A through Z, is appended when a task group is spawned as a result of the LOGIN command. When this argument is omitted, its default value is L.

When a user specifies a group id in a LOGIN command or in a login\_line for a T-record or A-record (see Appendix B), listener uses that as a group id instead of generating a group id.

"message"

The message-of-the-day, enclosed in quotes to provide for embedded blanks, which listener transmits to all login terminals for display.

After listener is activated, the system returns a message containing a message number, which requests a response. The message number must be included in the operator's response, from the operator terminal to listener, that changes the message-of-the-day. The response message cannot exceed 63 characters, and is in the format:

```
Δmsg_noΔmessage-of-the-day
```

Appendix B describes the login and listener characteristics, files, and required resources.

# **Section 3**

## ***System Operator Interface with the System***

### **SYSTEM OPERATOR AND OPERATOR TERMINAL**

The *system operator* and the system task group have two-way communication through the *operator terminal*.

### **OPERATOR TERMINAL**

The *operator terminal* is the single terminal-like keyboard device on the system that is designated as the control terminal for the system, and which must be assigned LRN 0 at system configuration to be identified as such by the system.

It is used to communicate with the:

1. System task group (\$S).
2. User application task group.

Communication with the system task group (\$S) can be only through the operator terminal, using the operator commands described in Section 4. However, the user or operator at the operator terminal can communicate with a user application task group by including the explicit or default task group id in any command or message directed to that task group.

The operator terminal is usually located near other peripheral devices, e.g., card reader, printer, or disk drive, so that the operator can quickly respond to system messages requesting action on those devices.

When in some cases the user may not require an operator terminal, for example, to use the terminal as a work station, none need be designated (i.e., LRN 0 is not assigned to a terminal at system configuration). However, the absence of an operator terminal results in certain conditions that are described in the *Program Execution and Checkout and Commands* manuals.

### **SYSTEM OPERATOR**

The *system operator* is the person who, from the designated operator terminal, controls system operation and communicates with the system.

### **OPERATOR INTERFACE MANAGER (OIM)**

The system communicates with the system operator through a system software component called the Operator Interface Manager (OIM), which controls the operator terminal input and output.

The OIM recognizes and processes these kinds of messages:

1. Output messages — Messages *from* a task group to the operator terminal.
2. Input messages — Messages *from* the system operator to a task group or to the OIM itself.

Message functions and formats are described more fully below.

The OIM provides standardized communication between tasks and the operator by:

- Imposing consistent spacing in operator input commands and in output messages to prevent overprint and ensure correct interpretation.
- Managing multiple input requests from the operator terminal.
- Ensuring prompt printing of critical output messages.
- Permitting the system operator to select which output request he will respond to and when.

## **OUTPUT MESSAGE**

Every output message written from a task group to the operator terminal has as its prefix the task group identifier, or id (see Section 1). When the message requires an operator's response, because the program executed a \$OPRSP macro call, a message number will precede that id. (The operator interface manager (OIM) inserts these task group id and message number prefixes.)

## **OUTPUT MESSAGE QUEUING**

Regardless of the number of task groups active in the system, a maximum of 10 numbered output messages that require operator response can be outstanding at one time. After a task has issued the tenth (message number 9) output message, and nine earlier messages are still outstanding, any task cannot issue any more messages requiring a response (i.e., with a message number prefix), until the operator responds to one or more of the 10 outstanding messages. Any task attempting to issue an eleventh (or successive) message is stalled until the operator responds to an outstanding message. OIM issues the message "OUTPUT STALLED, QUERY ANSWER REQUIRED."

## **OUTPUT MESSAGE FORMAT AND LENGTH**

The operator interface manager (OIM) imposes these standard format features on every output message:

1. A carriage return (C/R) and line-feed follows every message.
2. The first byte in the message is a control byte provided by the user program. This byte specifies either single or triple line spacing (head-of-form).
3. Every message is displayed at a standard pacing rate, or standard roll-up rate on a screen display. The operator can change the pacing rate with an input directive to the OIM (as described later).
4. Maximum message length, resulting from \$OPMSG and \$OPRSP macro calls, is 140 bytes. Any longer length specified by a task input/output request block (IORB) is reset to 140 by the OIM without notification to the task. If an issuing task is to specify a message length greater than the physical line length of the operator terminal, that task must provide enough embedded carriage returns and line feeds so that the message can be displayed on two or more lines.

## **INPUT MESSAGE**

An input message is sent by the operator from the operator terminal to a task group or to the OIM itself.

In the following descriptions of input message formats, the delta ( $\Delta$ ) character represents *exactly one* input space that the operator must include in his message; the characters C/R indicate a carriage return.

Once input is begun, it must be completed by a carriage return before the end of five minutes. Otherwise any input entered after that time is discarded.

## **INPUT RESPONSE MESSAGE**

There are two input response message formats in response to numbered output request messages.

1. The first requires the same message number prefix that was in the request output message, and is expressed as:

$\Delta n \Delta \text{text C/R}$

n — Output request message number

text — Message text

2. The second needs no message number prefix; OIM assumes it is a response to message number 0. It is entered as:

Δtext C/R

Specifying either format when there is no output message with number n or 0 causes an error message.

### RESPONSE TO TASK GROUP MESSAGE REQUESTS

There are two kinds of input responses to task group requests that are not specifically related to a numbered output message.

1. Response message prefix is the group id, so that this response is directed to the issuing task group.
2. Response message has no group id prefix. The message is thus intended for the operator-selected default task group. This format is best for an operator communicating with only one task group, and removes the need for specifying the group id with every input response.

The system task group (id is \$S) is the default task group immediately following system configuration. Operator commands therefore do not require the Δ\$\$Δ id prefix. However, any change in the default id, from \$S, by a directive to OIM (described below) would require that subsequent operator commands to the system task group \$S be preceded by Δ\$\$Δ.

### INPUT MESSAGE FORMAT AND LENGTH

Input message formats are summarized in Table 3-1.

**TABLE 3-1. INPUT MESSAGE FORMAT AND USE**

Input Message Format	Usage
messageC/R	Serial input to the task group defined as default task group to OIM by the operator. Until changed, the system task group \$S is the default task group immediately following configuration.
ΔidΔmessageC/R	Serial input to task group specified by id.
ΔmessageC/R	Input response to output message number 0.
ΔnΔmessageC/R	Input response to message number n (n is 0 through 9).

A carriage return terminates an input message and establishes its physical line length. Whatever the physical line length for a device (see Table 3-2), OIM accepts up to 140 bytes per line. When line length exceeds that limit, OIM considers the line as ended at 140 bytes, and treats the line as though a carriage return had been entered.

**TABLE 3-2. LENGTH OF INPUT MESSAGE BY DEVICE TYPE**

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

<sup>a</sup>The user must embed carriage control characters suitable for the device in the program output message to continue legible output beyond the physical line length limit.

<sup>b</sup>A 4-character message identifier.

## INPUT MESSAGE CONTROL

An input message can be entered even while the operator terminal is producing output, as follows:

1. Press the first input character key.
2. Wait until the output line is completed.
3. The system displays the input character.
4. Complete the input message.

To halt output to the operator terminal, enter one space only, omitting a carriage return. To resume output, type in the at-sign character @ on the same line to delete the space, then enter a carriage return.

To *delete an input message line* not yet accepted by the system (i.e., not followed by a carriage return):

1. Press and hold the CTRL (Control) key and press X.
2. System responds with \*DEL\*.
3. Do a carriage return.

This results in an "empty" line being sent to the system.

To *correct partial input*, i.e., delete erroneous input and enter correct input on the same line:

1. Press and hold the CTRL (Control) key and press X.
2. System responds with \*DEL\*.
3. Enter correct text on the same line.
4. Do a carriage return.

To *delete one or more characters* in a line, when a carriage return was not done, press the @ key immediately following entry of the erroneous character, on the same line. Each successive strike of the @ key deletes a character immediately preceding the last deleted character.

To *enter control characters as data characters*: press the \ (back slash) before pressing any of the following control characters to enter these characters as data characters: @ (at sign), CTRL X, CR (carriage return), and \ (back slash).

Table 3-3 shows input message control formats.

TABLE 3-3. INPUT MESSAGE CONTROL FORMAT

Input Message Control Format	Control Function
Δ (no C/R)	Halt output.
Δ@C/R	Halt, then resume output.
ΔtextCTRL X (System responds with *DEL*) C/R	Delete input designated by "text."
ΔtextCTRL X (System responds with *DEL*) correct text C/R	Delete input designated by "text." Replace on same line with "correct text."
@ (no C/R)	Delete the last entered character.
\ (back slash)	Enter the next entered character as data rather than as a control character.

## INPUT DIRECTIVE MESSAGES TO OIM

The system operator can issue directives for the Operator Interface Manager to:

- List outstanding output messages (those to which the operator has not yet responded).
- Change the default task group.
- Change the output pacing rate on the operator terminal display.
- Simulate the BREAK key action.

Every OIM directive must begin with an uppercase C, preceded and followed by exactly one space (i.e.,  $\Delta C\Delta$ ) and end with a carriage return.

### *Outstanding Output Message List*

To obtain a list of outstanding output messages, enter:

$\Delta C\Delta?C/R$

### *Change Default Task Group*

During operator/task group dialog, a task group can be designated as the default task group so that a group id prefix need not be specified with an input message to that group. (Note that immediately following system configuration the system task group id \$S is the default group id.)

To change the default task group id, enter:

$\Delta C\Delta:id:C/R$

id — The new default task group id.

This directive would be initially used to change the default task group from the system task group \$S to a user application task group id, and used later to change another group to the default task group.

### *Change Output Pacing Rate*

The pacing rate on an output display is the frequency at which each new output line appears (e.g., one line per second, or per five-tenths of a second, etc.). This control over pacing rate is useful with high-speed screen displays.

To change the pacing rate, enter:

$\Delta C\Delta PnnnC/R$

nnn — New pacing rate in tenths of a second. A value of 000 implies a pacing rate that is the fastest physically possible for that device. A value of 999 represents 99.9 seconds, or over 1-1/2 minutes.

Default pacing value is 000.

### *Task Interrupt (Break) From Operator Terminal*

The operator can interrupt (break) a task from the operator terminal. Operator entry is:

$\Delta C\Delta Bid$

Detailed procedures, response commands, and conditions for using the break function are fully described in Section 5.

## **ERROR MESSAGES ISSUED BY OIM**

The operator interface manager issues the messages shown in Table 3-4, at the operator terminal. (Note that these messages do not have any numeric codes.)



**TABLE 3-4. ERROR MESSAGES ISSUED BY OPERATOR INTERFACE MANAGER (OIM)**

<b>Message</b>	<b>Meaning</b>
GROUP id DID NOT ACCEPT INPUT	Task group identified by id did not accept the last-entered input directed to it.
INVALID COMMAND x	The command whose first character is x is invalid.
TERMINAL LINE RECONNECTED	The previously disconnected line between the operator terminal and the MLCP has been re-established.
NO BREAK ORDER FOR id	The ΔCΔBid (break) command is illegal for the task group identified by id.
NO QUERY FOR ANSWER n	Operator's input message includes the message number n; there is no outstanding output message with that number.
OUTPUT STALLED, QUERY ANSWER REQUIRED	Task attempted to issue an output message but no message number available since there are already 10 outstanding messages. Task stalled until operator responds to an outstanding message, and that message number becomes available.

**SAMPLE DIALOG BETWEEN SYSTEM OPERATOR AND THE SYSTEM**

Figure 3-1 represents a sample dialog between the system operator and the system. The boxed entries represent operator input at the operator terminal.

**INCREASING RESPONSE TIME FOR OPERATOR COMMAND EXECUTION**

When entering numerous successive commands and input messages to the system task group \$\$S at the operator terminal, the operator may want to increase command execution speed, especially with a diskette-based system.

To obtain faster response, enter the command

```
EC >SPD>CONSOLE
```

from the operator terminal (or as the last command in the START\_UP.EC file).

This command will cause input to be processed directly by the command processor rather than through the OIM, and will result in faster response at the operator terminal. This direct processing of operator commands requires several hundred more words of main memory from the system memory pool.

Another result of the EC >SPD>CONSOLE command is that the RDF (READY OFF) and RDN (READY ON) commands (described in the *Commands* manual) can be used as operator commands.

To restore the operator terminal to its former response rate, and to return the memory that was required to the memory pool, enter:

```
&QA or Δ$$SΔ&QA
```

as applicable.

**UNIT RECORD DEVICE TIMEOUT**

A unit record device (card reader, card punch, printer) may be offline because it was not turned on, was turned off, it failed, or ran out of cards or paper.

When an input or output order is issued to the offline device, or it goes offline while the order is being processed, the operator is notified and has five minutes to correct the condition, after which the condition is reported as an 0105 (device not ready) error.

A user can program an application program to test for this error condition and to then reissue the order anytime after the device goes online again.

However, a task request for a system program, such as a compiler, terminates abnormally when the device remains offline after five minutes.

(S)GCOS6 MOD400- <sup>{S}</sup> <sub>{L}</sub> 110-11/03/14/05	Output of system task group sign-on message
CG AB 26 -POOL AB EGR AB IW >SPD>CONSOLE -WD ^USER	Input to default task group (\$\$)
#0 (AB) KEY IN THE TIME PLEASE	Task group AB requests response
Δ1021	Operator response to message 0
(AB)END OF SECTION 1	Information message from task group AB
CG YZ 21 -POOL AB EGR YZ 1W >SPD>CONSOLE -WD ^USER	Input to default task group (\$\$)
#0 (AB) KEY IN THE TIME AGAIN	Task group AB requests response
#1 (YZ) MOUNT UNLINED PAPER PRINTER 1	Task group YZ requests action and response
Δ1ΔDONE	Response to preceding message 1
(YZ) ENTER LIST FOLLOWED BY "FINI"	Information message indicating need for input
ΔCΔ:YZ: JONES SMITH LAPIERRE FINI	Operator defines new default task group as YZ
ΔCΔ:\$S	Operator changes default task group to system task group (\$\$)
SSPG AB	Operator command input to default task group \$\$ to suspend task group AB
(YZ) ENTER LIST FOLLOWED BY "FINI"	Information message from group YZ
ΔYZΔLOWELL ΔYZΔHART ΔYZΔFINI	Operator serial input to task group YZ
ΔCΔ?	Operator request for list of outstanding messages
#0 (AB) KEY IN THE TIME AGAIN	Redisplay of message 0
Δ0Δ1100	Operator response to redisplayed message 0

Figure 3-1. Sample Operator/System Dialog



.

.



.

.



# Section 4

## Operator Commands

This section describes operator commands used by the system operator to control the operating system from the operator terminal. The system operator is one who communicates with the system through the terminal device that was assigned logical resource number (LRN) 0 at system configuration; that terminal is the designated operator terminal.

### OPERATOR COMMANDS AND COMMANDS

An order to be processed by the command processor is a command. A "secondary level" order passed through the command processor to a secondary processor such as the Linker or Configuration Load Manager is a *directive*.

In this manual, the terms "operator command" and "command" have different meanings and functions, as described below.

### OPERATOR COMMANDS

An operator command operates at the task group level; and is directed to and executed in the system task group (\$S). An operator command cannot be used to control the execution sequence of tasks in a batch or user online task group.

An operator command can be entered only through the designated operator terminal, or can be read from a command-in file.

The principal uses of an operator command are to:

- Control execution
- Control directories, files, or devices
- Monitor system operation

### OPERATOR COMMANDS FOR EXECUTION CONTROL

Operator commands used for execution control define the initial operating software environment, and then control system operation from the operator terminal; specifically they:

- Create, initiate, abort, or delete a batch or online task group
- Spawn an online task group
- Temporarily suspend and roll out, or reactivate and roll in, the batch task group
- Temporarily suspend or reactivate an online task group
- Load or unload a sharable bound unit from a system memory pool
- Load assembled firmware files into writable control store (WCS)

### OPERATOR COMMANDS FOR DIRECTORY, FILE, AND DEVICE CONTROL

Operator commands effect directory, file, and device control to:

- Change a system library or working directory pathname
- Modify the attributes of a disk file
- Cancel volume mount requests
- "Swap" similar device types

## **OPERATOR COMMANDS TO MONITOR THE SYSTEM**

Operator commands are used in monitoring the system to:

- List devices assigned and/or available
- List task groups and queued requests for batch execution
- List status of tasks or files in a task group
- List working directory pathname
- List the directories that are searched when a bound unit is requested

## **COMMANDS**

*Commands*, as differentiated in this manual from operator commands, enable any user to control any user task group through any device or file that can accept commands and deliver them to the command processor. Such a device or file, e.g., an MDC- or MLCP-connected terminal, is known as a command-in file. The *Commands* manual lists and describes commands in this category.

## **COMMAND PROCESSOR STANDARD INPUT/OUTPUT FILES**

The files associated with the command processor are:

- Command input file, or command-in file
- Operator output file, or operator-out
- Error output file, or error-out
- User output file, or user-out

Their functions and characteristics are described below.

### **COMMAND-IN FILE**

The command-in file is the file from which operator command lines are read. More specifically, it is the device designated as the operator terminal (the LRN 0 device established at system building). 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-in file reverts to the operator terminal.

### **OPERATOR-OUT FILE**

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

The operator-out 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-OUT FILE**

The error-out file is the file to which the command processor and any commands invoked by it write information related to error conditions detected by it. The error-out file is always the operator terminal; it cannot be reassigned by any operator command or command argument.

### **USER-OUT FILE**

The user-out file is the file to which a task group normally writes its output. However, certain system components (compilers, etc.) also write to list files (NAME.L) or to the output file defined in the -COUT argument. The user-out file is initially established by the -OUT argument of the EBR, EGR, or SG command. (Thus, originally, it is the same device as the error-out file device.)

It can be reassigned to another device by use of the **FILE OUT** command or by use of the **NEW USER OUT (\$NUOUT)** monitor call. Such a reassignment remains in effect for the task group until another reassignment occurs. See the *Commands* manual.

### **CONCURRENCY OF STANDARD I/O FILES**

Standard I/O files are reserved when a task group is spawned or requested. All nondisk standard I/O files are reserved for exclusive use. References to these files from within a task group will succeed; attempts to reserve these files from other task groups will fail. Although the operator terminal must be reserved with shared concurrency to allow read and write access by multiple groups, it can be used as a standard I/O file without any concurrency conflicts.

Disk standard I/O input files are reserved to allow multiple readers with no writers. Disk standard I/O output files are reserved for exclusive use.

### **INPUT COMMAND LINE**

Operator commands are read and interpreted by the command 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 command processor, which can then accept another command.

### **COMMAND LINE FORMAT**

A command line to the processor is a string of up to 127 ASCII characters in the form:

```
command-name [arg1 . . . argn]
```

where *command-name* is the pathname of the bound unit that performs the command's function. Each subsequent *arg* entry is an argument whose functions are described below. A command line cannot be continued onto the next line.

### **ARGUMENT**

An argument of a command is an individual item of data passed to the task of the named command. Some commands require no arguments; others accept one or more as indicated in the syntax of each command description. Optional arguments are enclosed in brackets; e.g., [*path*]. There are positional and keyword arguments (see below). Other types of arguments are the additional arguments that follow the **-ARG** keyword, available in some commands, and those following *path* in the **EC** command. They represent data that is to be used in the task group being activated and will be discussed below.

### **POSITIONAL ARGUMENT**

A positional argument is an argument whose position in the line indicates to which variable the item of data is applied. It can occur in a command line immediately after the command name or as the last argument following the control arguments, as in the **LIST NAMES** commands.

### **KEYWORD ARGUMENT**

A keyword argument is a fixed-form character string preceded by a hyphen, thus **-ECL**. It can be alone, as in **-WAIT**, or it can be followed by a value, as in **-FORM xx**.

### **CONTROL ARGUMENT**

A control argument is an additional argument or keyword argument and its value that specifies a command option; e.g., the pathname of an alternate input or output file. In the command syntax descriptions in this manual, control arguments are denoted by the term "ctL\_arg"; the argument descriptions define the specific keywords for that command. Unless otherwise noted, a control argument is optional, as indicated by enclosing brackets, i.e., [ctL\_arg]. A required control argument is so described in the syntax definition, without enclosing brackets.



Except when the last argument of a command line is a positional argument, keywords of control arguments can be entered in any order in the line, following the initial positional arguments.

### SPACES IN COMMAND LINES

Arguments in command lines are separated from each other by spaces. Unless otherwise indicated, a space in a command line syntax represents one or more space characters, or one or more horizontal tab characters, or a 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."

### OPERATOR COMMAND FORMATS AND DESCRIPTIONS

The rest of this section describes the formats, arguments, control arguments, and functions of the operator commands. Some complex cases include examples. Table 4-1 lists these commands by functional category, function name and command name. The command descriptions on subsequent pages are in alphabetic order by *function name*.

**TABLE 4-1. OPERATOR COMMANDS — FUNCTION AND COMMAND NAMES**

Function Name	Command Name
<i>EXECUTION CONTROL COMMANDS</i>	
Abort Group	ABORT_GROUP
Create Group	CG
Delete Group	DG
Enter Group Request	EGR
Execution Command	EC
Modify External Switches	MSW
Spawn Group	SG
Status Group	STG
<i>FILE AND DIRECTORY CONTROL COMMANDS</i>	
Change System Directory	CSD
Change Working Directory	CWD
File Out	FO
List Search Rules	LSR
List Working Directory	LWD
Modify File	MF
<i>INTERACTIVE COMMANDS</i>	
Enter Batch Request	EBR
Time	TIME
<i>OPERATIONS COMMANDS</i>	
Abort Batch Request	ABR
Abort Batch	ABORT_BATCH
Abort Group Request	AGR
Activate Batch	ACTB
Activate Group	ACTG
Create Batch	CB
Delete Batch	DB
Load Sharable Bound Unit	LOAD
Reassign	RAS
Set Date	SD
Status System	STS

**TABLE 4-1 (CONT). OPERATOR COMMANDS — FUNCTION AND  
COMMAND NAMES**

<b>Function Name</b>	<b>Command Name</b>
<i>OPERATIONS COMMANDS</i>	
Suspend Batch	SSPB
Suspend Group	SSPG
Unload Sharable Bound Unit	UNLD
Writable Control Store Load	WCSLD

## **ABORT BATCH**

Command Name: ABORT\_BATCH

Suspend, terminate, and delete the batch task group.

FORMAT:

ABORT\_BATCH

ARGUMENT DESCRIPTION:

No arguments are required for permitted with this command.

FUNCTION DESCRIPTION:

The ABORT BATCH command causes suspension and termination of the batch task group, whether active or dormant. It removes all 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 were 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

ARGUMENT DESCRIPTION:

No arguments are required or permitted with this command.

FUNCTION DESCRIPTION:

The ABORT BATCH REQUEST command stops execution of the current request in the batch task group. It removes all defining and controlling data structures except those associated with the command processor, and reruns 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 command processor honors the next request in the batch request queue, if any. The ABR takes effect as soon as all outstanding input or output orders are complete.

## **ABORT GROUP**

Command Name: ABORT\_GROUP

Suspend, terminate, and delete the indicated online task group.

### **FORMAT:**

ABORT\_GROUP id

### **ARGUMENT 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 suspension and termination of an existing online task group whether active or dormant. It removes all 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 open during the execution of the task group are closed. Any requests pending against the group are cancelled. The action of the ABORT GROUP command is similar to the DELETE GROUP command, except 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

ARGUMENT 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 stops 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. The AGR takes effect as soon as all outstanding input or output orders are complete.

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

Roll in and resume execution of the previously rolled-out and suspend batch task group.

FORMAT:

ACTB

ARGUMENT DESCRIPTION:

No arguments are required or permitted with this command.

FUNCTION DESCRIPTION:

The ACTIVATE BATCH command causes the roll in and resumption of execution of any tasks that were active at the time a SUSPEND BATCH command was issued. All tasks that were active at suspension are requeued on their respective level queues.

Execution resumes when the group is rolled in again; i.e., when all online task groups that extended into batch memory have returned that memory to the batch memory pool. See the SSPB command.

### **Note:**

It is possible that while the batch task group is rolled out as a result of an SSPB command, one or more online task groups running in extendable memory pools have obtained memory from the batch pool and have not returned that memory when the ACTB is issued. In that case the batch task group cannot be rolled in until all the batch memory is returned by the online groups.

## **ACTIVATE GROUP**

**Command Name:** ACTG

**Resume execution of a previously suspended online task group.**

**FORMAT:**

ACTG id

**ARGUMENT DESCRIPTION:**

id

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

**FUNCTION DESCRIPTION:**

The **ACTIVATE GROUP** command resumes 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 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]

ARGUMENT DESCRIPTION:

[path]

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

[ctl\_arg]

Only one control argument is recognized:

—LIBx

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

FUNCTION DESCRIPTION:

The CHANGE SYSTEM DIRECTORY command allows the system operator 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 a 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 was 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 working directory to the specified path.

FORMAT:

CWD path

ARGUMENT DESCRIPTION:

path

The pathname of the new working directory. It may be a relative name or a full pathname; it cannot exceed 44 characters.

FUNCTION DESCRIPTION:

The CHANGE WORKING DIRECTORY allows the system operator to modify the pathname of the system task group's default working directory. After command processor startup, the working directory is system\_volume\_name. However, there can be 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 that immediately contains these files.

For example, the series of functions that the system operator routinely performs at the beginning of a day's operations, after system initialization, 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 argument, the EC command could be given, specifying the simple name of the file containing the functions to be performed.

Example:

A file containing a series of CREATE GROUP operator commands is 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 GC commands.

## **CREATE BATCH**

Command Name: CB

Perform the initialization necessary to initiate the batch task group.

### **FORMAT:**

CB base\_lvl [ctl\_arg]

### **ARGUMENT DESCRIPTION:**

#### **base\_lvl**

A base priority level, *relative to the highest* system physical level, at which tasks in the batch task group will execute. A base (or relative) level of 0, if specified, is the next higher level above the system priority level. The sum of the highest system physical level plus 1, and the base level, and the relative level of a task, must not exceed  $62_{10}$ .

#### **[ctl\_arg]**

One or more control arguments from the following:

#### **-LRN n**

Specifies the highest logical resource number (LRN) to be referred to by any task in the batch task group. The highest LRN used by the system task group is 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 or GET FILE command in the *Commands* manual.

### **FUNCTION DESCRIPTION:**

The CREATE BATCH command causes 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 command processor, and defines it as the lead task of the task group. It does not cause the activation of the command processor; this is done by the use of the ENTER BATCH REQUEST command.

#### **Example:**

CB 20 -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 20, nor refer to a logical file number greater than 6.

## CREATE GROUP

Command Name: CG

Perform the initialization necessary to initiate an online task group.

FORMAT:

CG id base\_lvl ctl\_arg

ARGUMENT DESCRIPTION:

id

The group identification of the new task group, expressed in two characters. A user task group cannot have the \$ as the first id character.

base\_lvl

A base priority level, *relative to the highest* system physical level, at which tasks in the batch task group will execute. A base (or relative) level of 0, if specified, is the next higher level above the system priority level. The sum of the highest system physical level plus 1, and the base level, and the relative level of a task, must not exceed  $62_{10}$ .

ctl\_arg

One or more control arguments from the following:

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 and linked as sharable.

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

-ECL

The root segment of the command 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 highest LRN used by the system task group is 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 of 15. Refer to the ASSOCIATE PATH or GET FILE command in the *Commands* manual.

-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:**

-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 10 -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, in the system task group's working directory, whose execution is to be started at the symbolic address ENTRY1. No task in the group will execute at a base priority level lower than 10, 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

Mark the batch task group as eligible for deletion when it becomes dormant.

FORMAT:

DB

ARGUMENT DESCRIPTION:

No arguments are required or permitted with this command.

FUNCTION DESCRIPTION:

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 is returned to the system memory pool.



## **DELETE GROUP**

**Command Name:** DG

Mark the online task as eligible for deletion when it becomes dormant.

### **FORMAT:**

DG id

### **ARGUMENT 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 structure defining the group and any memory associated with the execution of the group are returned to the appropriate memory pool.

## **ENTER BATCH REQUEST**

Command Name: EBR

Enter a request in the batch request queue for execution of the command processor.

FORMAT:

EBR user\_id in\_path [ctl\_arg]

ARGUMENT DESCRIPTION:

user\_id

A field comprising three subfields in the form person.account.mode $\Delta$ , which identifies this request. See Section 1 for description of user\_id.

in\_path

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

[ctl\_arg]

One or more control arguments from the following:

-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 disk 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 null.

-ARG arg . . . arg

Indicates that additional arguments required by the task group during execution follow. These additional arguments are passed to the command processor to be used as necessary and are substituted for parameters in the command-in file. If used, the -ARG control argument must appear last. Refer to the beginning of this section for an explanation of the use of additional arguments.

FUNCTION DESCRIPTION:

The EBR command initiates execution of the command processor as the lead task in the batch task group previously created by the CB command. When the task group is dormant when the EBR command is issued, execution begins immediately. When the group is not dormant, the request for execution is queued, for execution when the task group becomes dormant, i.e., when the current batch request is terminated.

The command processor will first execute the EC file working-directory >START\_UP.EC (if there is one). The working directory is the one specified in the optional -WD path argument. Whether or not these files exist, the command processor remains active expecting more commands.

Since the command processor obtains its commands from the file named in the in\_path argument, that file must begin with a command, although it may contain other items, such as Editor directives, that the called command function may require for execution.

When displayed by system software, a user identification (user\_id) is shown as person.account.ABS.

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 null, since -WD was not specified.

## **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]

ARGUMENT DESCRIPTION:

id

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

user\_id

A field comprising three subfields in the form person.account.mode $\Delta$ , which identifies this request. See user\_id description in Section 1.

[in\_path]

The name of the file from which 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 from the following:

-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:

in\_path specifies a disk 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

Specifies that path is to be used as the working directory pathname.

-ARG arg arg . . . 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 and are substituted for parameters in the command-in file. If used, the -ARG control argument must appear last. Refer to the beginning of this section for an explanation of the use of additional arguments.

FUNCTION DESCRIPTION:

The EGR command initiates execution of the lead task of a task group previously created (by a CG command). If the task group is dormant when the EGR command is issued, task execution begins immediately. When the group is not dormant, the request for execution of the lead task is queued, for execution when the task group becomes dormant; (an earlier EGR command activated this task group and execution has not yet terminated).

Execution of the lead task begins at the point specified by the -EFN argument.

When the command processor is the lead task, the processor will first execute the EC file working-directory>START\_UP.EC (if there is one). The working directory is the one specified in the -WD path argument. Whether or not these files exist, the command processor remains active expecting more commands. After the START\_UP.EC file is executed, execution begins with reading the file named in the in\_path argument. That named file must begin with a command, although it may contain other items required for execution of the called command function.

**Example:**

```
EGR AX SMITH.SERVICES MPG_DATA -WD >UDD>SERVICES>SMITH -AGR '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 issuer's working directory, and will write its output to a file named MPG\_DATA.AO, in the same working directory. The working directory for group AX will be >UDD>SERVICES>SMITH. The lead task expects one argument, a date and the 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

Call the command processor to read operator commands from a designated file.

### **FORMAT:**

EC path [ct1\_arg]

### **ARGUMENT DESCRIPTION:**

#### **path**

The name of a file containing operator commands and EC directives. If path is a disk file, the system appends .EC to path.

#### **[ct1\_arg]**

The list of additional character string arguments, arg arg . . . arg, that are to be substituted for substitutable parameters in the input lines of the command-in file. The pathname of the EC file is substituted for all occurrences of &0 in the command-in file, the first additional argument for all occurrences of &1, the second additional argument for all occurrences of &2, etc. Refer to the beginning of this section for details about additional arguments.

### **FUNCTION DESCRIPTION:**

The command processor reads from a previously created file a series of operator commands and EC directives. It provides a mechanism to execute a sequence of routinely performed functions without manual entry of commands through the operator terminal.

The file path.EC is a sequentially processed file containing ASCII images of one or more operator commands and EC directives. These images are interpreted by the command processor as indicated below.

When a command is encountered, it is simply passed to the command 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 entered from a terminal device if the function were requested manually. All arguments must be supplied as specified in the individual operator command descriptions.

When a command execution terminates, control is returned to the command processor, which then reads the next line from the file.

The EC file can also contain EC control directives that are not passed to the command processor, but are interpreted and acted upon by the directive routines. 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 command 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 (see *System Messages* manual).

The EC control directives are described in detail below.

#### **&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 current command-in file (EC file) is to be substituted for the user input stream. This means that whenever the executing task refers to its user-in file, the data which would normally be read from this file is obtained instead from the file being read by the command processor.

#### **&DΔ**

This directive restores the user-in file to that which existed when the EC file was invoked.

#### **&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.

## &1FΔ

This directive permits the interrogation by the command processor of an error status code returned by the command executed immediately prior to the &1FΔ directive. For compatibility across all GCOS systems, it has the format:

```
&1F Δ[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 &1FΔ directive enables the command 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 command 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\_GRPES.EC and the following operator commands are contained in it:

```
CB 10 -LRN 10 -LFN 8
CG EC 5 -LRN 12 -POOL A1
CG PR 2 -LRN 15 -LFN 6 -EFN PROD_A -POOL P1
EGR EC GRP.PRIME >SPD>VIP01
EBR DEV.PROG >SPD>KSR02 -OUT>SPD>LPT01 -WD >UDD>PROG>DEV
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 level 10 with the ECL processor as its lead task. Its user\_id, established by the EBR command, is DEV.PROG. Its working directory is >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 command 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 commands are required. Task group EC operates at base level 5 and utilizes memory from the memory pool as A1 at system building. Its input and output are through the terminal VIP01.

The second CG command creates an online task group PR, which uses a bound unit named PROD\_A and its lead task. It operates at the highest level of the three tasks 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 operator commands, an appropriate error message displayed to the system operator and processing of the EC file continues.

The sequence of commands in the above example is intended to show that, once a set of task groups has been created, the order of activation (by the EBR or EGR command) 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]

ARGUMENT 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 argument is omitted, the operator-out file reverts to that established at the conclusion of command 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. The file or device is reserved with exclusive concurrency except that the operator terminal is reserved with shared read/write. When the command 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 argument resets the destination of the operator output to the operator terminal.

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

ARGUMENT DESCRIPTION:

No arguments 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 loader 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 command processor startup, and remains so until modified by one or more CHANGE WORKING DIRECTORY commands. The second is the system directory LIB1. The third is the system directory LIB2. 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:

The system task group's initial working directory is ^SYSVOL, the pathname value for LIB1 and LIB2 is ^SYSVOL>SYSLIB1, and no CWD or CSD commands were issued. The LSR command returns

```
^SYSVOL
^SYSVOL>SYSLIB1
^SYSVOL>SYSLIB2
```

A CSD NEW\_DIR -LIB2 command was executed at some point prior to issuance 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

**ARGUMENT DESCRIPTION:**

No arguments are required or permitted with this command.

**FUNCTION DESCRIPTION:**

The LIST WORKING DIRECTORY command can be used to write to the operator-out file the full pathname of the working directory currently being used by the system task group. It is 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-out file in the form

^ volume\_name[>dir1]...

The ellipsis indicates that one or more subordinate levels may be included in the pathname 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 command 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

## **LOAD SHARABLE BOUND UNIT**

Command Name: LOAD

Load a sharable bound unit into the system memory pool.

FORMAT:

LOAD path

ARGUMENT DESCRIPTION:

path

Pathname of the sharable bound unit to be loaded.

FUNCTION DESCRIPTION:

This command (or a command from the system START\_UP.EC file) loads the named sharable bound unit into the system memory pool. The effect of this command is as though a "dummy" task group had been created with the sharable bound unit as its lead task, but without a group data structure, then not using or terminating the task group.

The bound unit to be loaded must have been linked with both the SHARE and SYS directives otherwise an error message results and the bound unit is not loaded.

Any bound units loaded by this command can be unloaded only with an UNLD operator command (described later in this section), otherwise the named bound unit will remain in the system memory pool.

## **MODIFY EXTERNAL SWITCHES**

Command Name: MSW

Modify selected external switches associated with the indicated task group.

FORMAT:

MSW id ctl\_arg

ARGUMENT DESCRIPTION:

id

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

ctl\_arg

One or more control arguments from the following:

-ONS<sub>i</sub>[<sub>v</sub>S<sub>i</sub>] ...

Set the external switch indicated by S<sub>i</sub> ON. Each S<sub>i</sub> is a hexadecimal digit from 0 through F.

-OFF S<sub>i</sub>[S<sub>i</sub>] ...

Set the external switch indicated by S<sub>i</sub> OFF. Each S<sub>i</sub> 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 of 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 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**

**ARGUMENT DESCRIPTION:**

**path**

The pathname file whose attributes are to be changed.

**ctl\_arg**

One or more control arguments from the following:

{ -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 (see CREATE FILE command in the *Commands* manual) 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.

Read protection can also 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 by online tasks and the read/write protection remains unchanged.

## **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*

ARGUMENT DESCRIPTION:

*ctl\_arg*

One control argument from the following:

**-SWAP** *dev\_name<sub>1</sub>* *dev\_name<sub>2</sub>*

The device controlled by the driver associated with *dev\_name<sub>2</sub>* is to be interchanged with the device under the control of the device driver associated with *dev\_name<sub>1</sub>*. Both devices must be of the same type and must be offline (in standby).

**-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. For disk, if the mount message is "MOUNT ^ vol id", the form of the name argument is ^ vol\_id; if the mount message is "MOUNT>SPD>dev\_name", the form of the name argument is *dev\_name*. For magnetic tape, the form of the name argument is always *dev\_name*.

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<sub>1</sub>* and *dev\_name<sub>2</sub>*, respectively. The device names are those assigned to the devices at system building.

If a task issues a request for a file or volume of 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.

ABORT GROUP REQUEST or ABORT BATCH REQUEST must not be directed to a task group which has a mount request outstanding. Prior to aborting the task group, any mount request issued for it must first be cancelled through the use of the REASSIGN command.

Example 1:

**RAS -SWAP LPT00 LPT01**

The printer LPT00 is to be interchanged with printer LPT01.

Example 2:

**RAS -CANCEL ^ USER04**

A task has requested the mounting of a volume USER04, and the file manager issued a message to the operator directing him to mount the volume. The operator 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]]'

ARGUMENT 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 or quotes because of the embedded space between the dd and hh portions.

The SET\_DATE should be issued immediately after system initialization. The date/time, once specified, should not be respecified to an earlier date/time without reinitializing the system.

## SPAWN GROUP

Command Name: SG

Create, request the execution of, and then delete a task group.

### FORMAT:

SG id user\_id base\_lvl [in\_path] ctl\_arg

### ARGUMENT DESCRIPTION:

id

The group identification of the task group to be spawned, expressed in two characters. A user task group cannot have the \$ as its id first character.

user\_id

A field comprising the subfields person.account.mode $\Delta$ , which identifies this request. See user\_id description in Section 1.

base\_lvl

A base priority level, *relative to the highest system physical level*, at which tasks in the batch task group will execute. A base (or relative) level of 0, if specified, is the next higher level above the system priority level. The sum of the highest system physical level plus 1, and the base level, and the relative level of a task, must not exceed 62<sub>10</sub>.

[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 argument is not specified. in\_path must be specified if the command argument -ECL (see below) is used or implied.

ctl\_arg

One or more control arguments from the following. 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 disk 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.

-EFN root

-EFN root?entry

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

-ECL

Specifies that the lead task of the spawned task group will be the command processor.

-LRN n

Specifies the highest LRN that will be referred to by any task in this task group. The highest LRN used by the system task group is 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 2-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 arg arg ... arg**

Indicates that additional arguments required by the spawned task group during execution follow. These additional arguments are passed to the lead task of the spawned group to be used as necessary and are substituted for parameters in the command-in file. If used, the -ARG control argument must appear last. Refer to the beginning of this section for an explanation of the use of additional arguments.

**Note:**

If any SG command, -EFN or -ECL, but not both, can be specified. If neither is specified, -ECL is assumed and the in\_path argument 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 existence 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 command processor as its lead task is an example of such a group; it is initiated by the SG command, and it is terminated by a BYE command entered by the user on whose behalf the task group is running. The duration of any task group may vary, the significant difference between a spawned task group and a created task group is that, in the former case, the operator need not know 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 UPMAS -OUT MSTR LST -POOL M1 -WD UDD>REMOTE>SMITH
```

The data structures defining and controlling task group UM are allocated and initialized. A request identified as SMITH.REMOTE as entered against the task group. The group's working directory is >UDD>REMOTE>SMITH. A bound unit, UPMAS, 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, MSTR LST, also contained in the working directory. The base 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 UPMAS program as a signal to issue a system service call to the Monitor to terminate 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 MSTR LST 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]

ARGUMENT 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 from the following.

**-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, and whether they are open or closed.

FUNCTION DESCRIPTION:

The STATUS GROUP command writes to the operator-out 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
  - X = Task is being terminated
  - D = Dormant
  - A = Active
- Memory pool identification, if the task group is not batch
- Current user\_id. User\_id for the batch task group is person.account.ABS.
- Full pathname of the error-out file
- Full pathname of the user-out 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
- First six characters of the task's bound unit name
- Full pathname of the command-in file
- Full pathname of the user-in file.

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

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

<b>For Group id</b>	<b>For Other Groups</b>
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

- File type. The rightmost six bits of the status word form a hexadecimal value for the file type; i.e., the left hexadecimal digit of the hexadecimal value can only represent 0 through 3. See the COMMAND IN macro call in the *System Service Macro Calls* manual for the file type descriptions.
- Logical file number, if one is associated with the file, otherwise spaces.
- Open/closed status of the file. O for open, C for closed.

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.

When the system hardware includes the memory management unit (MMU), use the following command to display the status of the batch task group \$B:

`Δ$$ΔSTGΔ$B`

## **STATUS SYSTEM**

Command Name: STS

Display general system status.

FORMAT:

STS [ctl\_arg]

ARGUMENT DESCRIPTION:

[ctl\_arg]

One or more control arguments from the following.

-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.

-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 allows the system operator 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
  - X = Task is being terminated
  - D = Dormant
  - A = Active
- Memory pool identification, if the task group is not batch
- 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 is returned in the form:

b dev\_name vol\_id { D  
nn }

Contents of the display are defined as:

b

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

dev\_name

Device unit name or file name of a device specified in a CLM DEVICE directive

vol\_id

Name of the volume mounted on dev\_name

D

Device is currently disabled

nn

00 if device contains no active files (i.e., device is available); or 01-99<sub>10</sub>, 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, and roll it out of memory.

FORMAT

SSPB

ARGUMENT DESCRIPTION:

No arguments are required or permitted with this command.

FUNCTION DESCRIPTION:

The SUSPEND BATCH command stops execution of any tasks that may be active in the batch task group, after completion of any outstanding input/output requests. Then, provided that at least one memory pool was configured (with the x argument of the CLM directive MEMPOOL), the batch group is then rolled out. The task group remains suspended and rolled out until it is reactivated by an ACTIVE BATCH command. All controlling structures remain intact and memory used by the group is returned to the memory pool during the suspended state.

The batch task group is rolled out (if memory was configured as indicated above) following the suspend action. It cannot be rolled in until the operator issues an ACTB (ACTIVATE BATCH) command.

If another task group that forced the batch task group to be rolled out (with a \$SUSPG (Suspend Group) macro call) is aborted or terminated without explicitly enabling roll-in of the batch task group, the operator must issue an ACTB command for the batch task group to be rolled in.

## **SUSPEND GROUP**

Command Name: SSPG

Temporarily terminate the execution of the specified online task group.

FORMAT:

SSPG id

ARGUMENT DESCRIPTION

id

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

FUNCTION DESCRIPTION

The suspend group command stops 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 memory used by the task group is not to the group's memory pool during the suspended state.

If another task group that forced this task group to be suspended, with a **\$SUSPG** (Suspend Group) macro call, is aborted or terminated, the operator must issue an **ACTG** command to activate the task group.

## **TIME**

Command Name: TIME

Display the current date and time in ASCII format.

FORMAT:

TIME

ARGUMENT DESCRIPTION:

No arguments 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

The information returned by the TIME command depends upon the accuracy of the data entered in the SET DATE command.



## **UNLOAD SHARABLE BOUND UNIT**

Command Name: UNLD

Unload from the system memory pool the sharable bound unit previously loaded with the LOAD operator command.

FORMAT:

UNLD path

ARGUMENT DESCRIPTION:

path

Pathname of the bound unit to be unloaded.

FUNCTION DESCRIPTION:

The UNLD command removes from the system memory pool the sharable bound unit, with the same pathname, that was previously loaded by the LOAD operator command (described earlier in this section). Memory space used by the bound unit is returned to the system memory pool.

The UNLD command does not take effect until there are no more users for the sharable bound unit.

An attempt to unload any bound unit *not* previously loaded by the LOAD operator command results in an error message; the bound unit is not unloaded.

## WRITABLE CONTROL STORE (WCS) LOAD

Command Name: WCSLD

Load one or more firmware object text files, stored on disk, into the writable control store (WCS).

FORMAT:

WCSLD [path<sub>1</sub>] [path<sub>2</sub>] . . . [path<sub>n</sub>] [ { -DUMPF } [ (xxx,yyy) ] ]  
[ { -FILL } (xxxx,xxxx,xxxx,xxxx) ] [-OFF] [ { -CPU } (x) ]

### ARGUMENT DESCRIPTIONS:

path ... path

Full or relative pathname(s) of firmware object text file(s) to be loaded

{ -DUMP } (xxx,yyy)  
-D

Dump locations of the WCS, from locations xxx through yyy inclusive, to the user-out file after the WCS is loaded. xxx is the beginning (start) address of the area to be dumped, yyy is the inclusive end address. When both are omitted, the default is that the entire contents of the WCS random-access memory (RAM) are dumped. When xxx is omitted the default is the low address for any RAM in the WCS; when yyy is omitted, the default is the RAM's highest address. For the largest possible RAM the low address is 800<sub>16</sub>, the highest is FFF<sub>16</sub>.

-FILL { -FILL } (xxxx,xxxx,xxxx,xxxx)  
-FL -FL

Fill all remaining locations in the WCS, that were not written to when the firmware files were loaded, with the user-specified firmware word that comprises four 4-character hexadecimal words, separated by commas, all enclosed by a pair of parentheses.

-CPU(x)

Apply all arguments in this command to the WCS associated with the central processor on the channel indicated by x.

-OFF

Disable the WCS (i.e., set it offline) after the WCS loader has completed all actions indicated by the other WCSLD arguments.

Although all arguments for this command are optional, *at least one argument* must be specified. Numerical values in arguments are hexadecimal.

### FUNCTION DESCRIPTION:

The WCSLD command causes the WCS loader to be executed under Monitor control in the system task group. The loader loads the WCS with user firmware in the form of object text files previously created by a WCS assembler and stored on disk. The writable control store (WCS) is that hardware storage in microinstruction memory, located immediately above Honeywell-supplied firmware, into which the user can load his own firmware. (See the hardware manual *Writable Control Store User's Guide*, Order No. FQ41.)

According to the specified arguments, the WCS loader functions are:

1. Load one or more firmware files into WCS.
2. Fill unused locations in the WCS with an operator-specified firmware word.
3. Dump contents of the WCS.
4. Disable the WCS.

The operator can request any function in any order separately or concurrently with another. However, the loader will execute each function, if requested, in the order shown above. Multiple firmware files (e.g., path<sub>1</sub>, path<sub>2</sub>, path<sub>n</sub>, etc.) are loaded in the order in which specified. An

attempt to load more than one file in the same location in one execution of the loader causes a warning error message.

The WCS loader assumes that all firmware object text files were generated by the WCS assembler, and that their pathname(s) end with the suffix .WO.

When the loader has completed its execution, it reports to the operator the internal name of each loaded file, its revision number, the date that it was assembled, and 20 characters of additional identification information. It does not identify the loaded files by their external directory names.

When the WCS does *not* include any RAM's (random-access memory), the only arguments that can be entered with the WCSLD command are -OFF and -DUMP.

The -OFF argument can be used to prevent user-coded generic instructions from being executed in the WCS PROMs.

# Task Interrupt (Break) from Operator Terminal

The user at the operator terminal can interrupt or “break” a running task in order to reenter commands, temporarily halt the task, or to terminate it. The break sequence begins with the command  $\Delta C\Delta Bid$  (id is the task group identifier). Subsequent steps and other conditions are described later in this section.

At a user terminal that is *not* the operator terminal, the break can be activated by pressing the appropriate BRK (Break) or INTERRUPT key. Conversely, pressing the BRK or INTERRUPT key at the operator terminal has no break effect.

### BREAK FUNCTION USAGE

Typically a break from the interactive command-in terminal can be used to interrupt:

- Any program running in a task group whose lead task is the command processor.
- Any program invoked through a \$CMDLIN (process command line) Monitor macro call from the lead task.

The break cannot be used with a program that is designated as the lead task in a create group or spawn group command. It can be used only under the following conditions:

- When entered from an interactive command-in terminal (see the *Commands* manual).
- When used to interrupt a program invoked from the lead task and by a command to the command processor.
- When the task to be interrupted is not running under the system task group (\$S). The break cannot be used to interrupt execution of any operator commands.

### BREAK PROCEDURES

A break is effective only with an active running task. If the command processor is inactive, waiting for input, a break typein from the operator terminal will have no effect, and will result in the message:

NO BREAK ORDER FOR id

To effect a break (task interrupt) in a running task from the operator terminal:

1. Type  $\Delta C\Delta Bid$  (id is the task group identifier)
2. The system then:
  - a. Might truncate the current output line
  - b. Temporarily suspends the active task
  - c. Puts the lead task into “break mode”
  - d. Issues the break prompter message **\*\*BREAK\*\***
3. Enter a response according to one or more of the following shown in a, b, c, or d below.
  - a. Enter any command (see the *Commands* manual), that is not an operator command (see Section 4). This may be followed by another command or by one of the response commands described later in this section. When the entered command that is *not* SR, BYE, NEW \_ PROC, UW, or PI (described later), completes execution, the lead task again enters break mode and issues another **\*BREAK\*\*** prompter message requesting another response.

- b. Enter one of the following break mode responses to the **\*\*BREAK\*\*** message:
- (1) **SR (Start)** This resumes execution of the suspended task i.e., as though the break had not been made.
  - (2) **BYE (Bye)** This aborts and deletes the current task group request.
  - (3) **NEW \_ PROC (New Process)** This aborts all task requests in the task group except for the lead task; then restarts the task group, using the same arguments as specified in the initial task group request.
- Any of these commands terminates the current break, i.e., there will be no other **\*\*BREAK\*\*** message after they are executed.
- c. Enter **UW (Unwind)** If the current task is a Honeywell program listed in Table 5-1, it terminates itself and returns all its resources. The break responses indicated in 3a and 3b above are also usable with these programs. These system programs must be running in a task group whose lead task is the command processor.
- If the terminated task was invoked following a break, the lead task reenters break mode, issues another **\*\*BREAK\*\*** prompter message, and awaits a response.
- If the terminated task did *not* follow a break, processing continues as though the task terminated normally.
- A **UW** command to any system program other than shown in Table 5-1 results in a 0343 or 0344 error return, followed by another **\*\*BREAK\*\*** prompter message.
- d. Enter **PI (Program Interrupt)** For Linker and Editor, suppress output and return to directive input level. The **PI** command suppresses output resulting only from the Linker **MAP** directive and from the Editor **P-type** directives.
- The **PI** command is meaningful only to the Linker and Editor running in a task group whose lead task is the command processor. The commands described in 3a, 3b, and 3c above are also usable with Linker and Editor.

**TABLE 5-1. SYSTEM PROGRAMS SUPPORTING UW (UNWIND) COMMAND**

Command Name	Function	Command Name	Function
ASSEM	Assembler	LCD	List Creation Date
COBOL	COBOL Compiler	LINKER	Linker <sup>a</sup>
CP	Copy	LS	List Names
CPA	Compare	MACROP	Macro Preprocessor
CV	Create Volume	MERGE	Merge
DP	Dump Edit	PR	Print
ED	Editor <sup>a</sup>	SORT	Sort
FC	File Change	STG	Status Group
FD	File Dump		

<sup>a</sup>Both Editor and Linker also support the **PI (Program Interrupt)** command.

### **UW AND PI COMMANDS IN USER APPLICATION PROGRAMS**

**PI** and **UW** commands are effective in user application programs only when the task to be interrupted has been previously enabled for the necessary trap. The user program must include the **\$TRPHD** and **\$ENTRP** Monitor service macro calls for the simulated trap.

### **BREAK COMMAND EXAMPLES**

#### **Example 1:**

The Editor is executing a print directive, and during output, the user enters the break message **ΔCΔBid**, thereby stopping further output. After the **\*\*BREAK\*\*** message appears, the user responds with **PI**, which returns the program to directive input level. A response of **UW** instead of **PI**, would have terminated the Editor.

**Example 2:**

An LS (LIST NAMES) command is executing with output going to the operator terminal. The user wants to change the output path to the line printer. One possible method is:

- |   |  |
|---|--|
| 1. Enter ΔCΔBid   |  |
| 2. System responds with <b>**BREAK**</b>  | Lead task enters break mode                |
| 3. Enter FO >SPD>LPT01  | File out command specifying a line printer |
| 4. FO execution terminates; the system issues another <b>**BREAK**</b> message. |  |
| 5. Enter SR (start) command.  | Resume execution of the LS command.        |

Another possible method is:

- |  |   |
|--|---|
| 1. Enter ΔCΔBid                          |   |
| 2. System responds with <b>**BREAK**</b> | Lead task enters break mode.                          |
| 3. Enter the UW command                  | The current LS task terminates itself.                |
| 4. Enter FO >SPD>LPT01                   | File out command specifying a line printer            |
| 5. Enter LS                              | Start the list names (LS) program from the beginning. |

**Example 3:**

This example shows successive nested break functions. Though representing a continuous procedure, the example is shown in numbered sequences for clarity.

1. The first sequence includes a command to the command processor to invoke the Editor, then to read and print the file PATH1. A break command is entered to interrupt the output, which was found to be from the wrong file.
2. Following issuance of the **\*\*BREAK\*\*** message, the user enters LS (list names) to obtain a display of PATH2 file names. He then enters another break command to interrupt that LS command in order to change the pathname from PATH2 to PATH3.
3. A new LS command is entered to list the files in PATH3; however, the preceding LS command (for PATH2) is not terminated, but remains suspended. The required file is found at the beginning of the listing, the rest of the PATH3 list is not needed so the user enters a break command to interrupt listing of PATH3.

The following command sequences are keyed to preceding numbered descriptions.

- |                                |   |
|--------------------------------|---|
| 1. Enter RDN                   | The system will print RDY: as each command completes execution. |
| Enter ED                       | Activates the Editor  |
| Enter R PATH1                  | Read the file PATH1   |
| Enter 1,&P                     | Print the file PATH1  |
| Editor issuing print lines     |   |
| Enter ΔCΔBid                   | Causes a break in printing                                      |
| System issues <b>**BREAK**</b> | Command processor is in break mode                              |
| 2. Enter LS -PN PATH2          | List the PATH2 directory  |
| System printing the list       | User determines list is for wrong directory                     |
| Enter ΔCΔBid                   | Causes a break in the LS command for PATH2                      |
| System issues <b>**BREAK**</b> | Command processor is in break mode                              |

3. Enter LS -PN PATH3 -FILE

System issuing the list

Enter ΔCΔBid

System issues \*\*BREAK\*\*

List files in PATH3 directory

User finds desired file, no more output needed

Causes break in LS command for PATH3

Command processor is in break mode

Subsequent actions are described as separate alternatives in Example 4 below.

Example 4:

This example consisting of five discrete actions, continues from Example 3, and in particular shows the use of the UW command to terminate successively activated tasks (i.e., unwind stacked tasks). Each part of the example is a separate procedure, independent from the others, and shows an alternative method of continuing with Example 3.

1. Start again at command level.

Enter NEW\_PROC

Aborts all prior tasks; the command processor is ready for input.

2. Return to the Editor directive input level.

Enter UW

LS command for PATH3 terminates itself.

System issues \*\*BREAK\*\*

Since the LS for PATH3 followed a break, the command processor reenters command mode.

Enter UW

LS command for PATH2 terminates itself.

System issues \*\*BREAK\*\*

Since the LS for PATH2 followed a break, the command processor reenters command mode.

Enter PI

Editor is ready for the next Editor input directive.

Enter next Editor directive

3. Return to command level by terminating in turn each previously activated task.

Enter UW

LS command for PATH3 terminates itself.

System issues \*\*BREAK\*\*

Command processor enters break mode.

Enter UW

LS command for PATH2 terminates itself.

System issues \*\*BREAK\*\*

Command processor enters break mode.

Enter UW

The Editor terminates itself.

System issues RDY:

Prompter message at command level.

Enter next command

4. Complete the printout of PATH1 file.

Enter UW

LS command for PATH3 terminates itself.

System issues \*\*BREAK\*\*

Command processor enters break mode.

Enter UW

LS command for PATH2 terminates itself.

System issues \*\*BREAK\*\*

Command processor enters break mode.

Enter SR

Restarts printing out of PATH1 from point of interrupt.

Editor issues print lines.

5. Delete current task group request.

Enter BYE

Deletes all task group request structures except the lead task. Another task group request is required to activate the lead task.





# Appendix A

## Additional Command Line Arguments (ARG)

This appendix shows in detail how to use additional arguments that appear in command lines of operator commands, and applies when a user contemplates sophisticated use of the system.

For command lines dealing with task or group activation there is a mechanism for handling additional arguments entered in the command line. If an activated task is a user application, the arguments are passed to the task for processing. On any command-in file that is not an interactive file or on a user-in file that is the same as a noninteractive command-in file, parameter substitution of command line arguments also occurs.

### ARGUMENT PASSING

The arguments following the keyword `-ARG` in the `EGR`, `EBR`, and `SG` commands are passed to the activated task. If the task is the command processor, the argument list is used for parameter substitution. When the activated task is software used in program preparation or a utility, it uses the values in the argument list for its own required arguments.

Examples:

```
EGR AX SMITH.SERVICES -OUT>SPD>LPT00 -WD ^ VOLA>JR -ARG -IN >SPD>CRD00 -LL 80
```

A previous `CG` command has identified `ED`, for Editor as the name of the bound unit root segment to be loaded as the lead task. The arguments supplied to the Editor, namely, the input file pathname and maximum line length are included in the argument list for the `EGR` command.

```
EGR AX SMITH.SERVICES -OUT >SPD>LPT00 -WD ^ VOLA>JR -ARG FILEA >UDD>BOOKS>FILEA -PR 20
```

A previous `CG` command has identified `CPA`, for the compare utility, as the name of the bound unit root segment to be loaded as the lead task. The first and second arguments in the argument list are used for the first and second positional arguments in the compare command line. The third argument in the list is an optional keyword argument passed to the utility.

If the task being activated is a user application that is to be passed arguments, it must contain an assembly language routine that looks at the argument list in its parameter block. The parameter block is a variable size augment to the task request block. These structures are described in the *System Service Macro Calls* manual.

### INPUT COMMAND LINE PARAMETER SUBSTITUTION

A substitutable parameter in the command-in file is an ASCII character string whose first character is an ampersand followed by one or more digits, e.g., `&0`, `&1`, . . . The digit indicates the position in the argument list of the data element to be substituted, namely, the first argument is substituted for `&0`, the second for `&1`, etc. Depending on the case, the first argument can be path or the first additional argument.

If the argument list is smaller than the number of substitutable parameters present in the command-in file, the null parameter is substituted for all parameters not supplied in the argument list; e.g., `XY&1Z` is the substitutable line; after substitution with the null parameter for `&1`, it is `XYZ`.

Parameter substitution enables a user to change parameters in a noninteractive `EC` file. For example, this can be applied when an `EZ` command line is an abbreviation for a set of parameterized functions. Parameters are substituted for all lines read from the command-in

file. When &A is present in the EC file, parameters are substituted for all lines read from the user-in file.

Nesting of argument lists is supported when a command line with additional arguments specifies a command file that, in turn, contains a command line with additional arguments that specifies a command file etc. At each level of nesting, the argument list to be used for the parameterized command file is taken from the arguments in the command line that specified the command file.

### **EC FILE EXECUTION COMMAND**

The EC command has the format:

```
EC path [arg 1 arg 2. . .]
```

In the parameterized command-in file path is substituted for all occurrences of &0 arg 1 for all occurrences of &1, etc. To parameterize directives to system software, e.g., Linker, in the EC file, an &A directive must precede the command line activating the system software. This changes the user-in file to be identical to the command input file which is the EC file.

Example:

A task group AX has previously been created with the command processor as its lead task. To execute the EC file, enter the following command lines:

```
Δ$$ΔEGR AX IW >SPD>CONSOLE -WD ^ VOLA>JR  
ΔAXΔEC ASM_LNK TEST>SPD>LPT01 -NL TEST.L START_AD DIR>SEC
```

The contents of the EC file ASM\_LNK.EC are:

```
ASSEM &1 -COUT &2 &3  
&A  
LINKER &1 -COUT &4  
parameterized linker directives
```

After substitution the command lines contain:

```
ASSEM TEST -COUT >SPD>LPT01 -NL  
&A  
LINKER TEST -COUT TEST.L  
parameterized linker directives
```

After EGR is executed, both user-in and command-in files are the operator terminal. After the EC is executed, user-in is still the terminal whereas command-in is ASM\_LNK.EC. The &A directive in the EC file is required to change the user-in file to be the same as the command-in file, namely, ASM\_LNK.EC. If &A is not included, Linker directives are read from >SPD>CONSOLE. TEST is substituted for all occurrences of &1, >SPD>LPT01 for &2 -NL for &3 and TEST.L for &4. The first parameter &0 is not used. The arguments START\_AD and DIR>SEC are substituted for parameters in Linker directives.

Example:

The command line is:

```
EC ASMBL TEST -COUT TEST.L -SAF
```

The contents of the EC file, ASMBL.EC are:

```
&PΔSTART ASSEMBLY of &1  
ASSEM &1 &2 &3 &4 &5 &6 &7 &8  
&PΔEND ASSEMBLY  
&QΔ
```

After substitution the command lines contain:

```
&PΔSTART ASSEMBLY OF TEST  
ASSEM TEST -COUT TEST.L -SAF  
&PΔEND ASSEMBLY  
&QΔ
```

ASMBL is not substituted for any parameter. Since the first parameter, &1, is a positional parameter, the first argument must always refer to path, TEST in the example. Entries for the keyword parameters can appear in any order in the argument list or not appear at all.

**Example:**

If in the above example the command line was mistakenly entered as:

```
EC ASMBL TEST -COUT -SAF
```

After substitution the ASSEMB command line contains:

```
ASSEMB TEST -COUT -SAF
```

The positional argument following -COUT is missing and the next argument, -SAF, is substituted in its place resulting in an incorrect command line.

### GROUP REQUEST COMMANDS

The following commands have parameter substitution performed on the command-in files read by the lead task. The operator command formats are shown; corresponding commands do not have a user\_id argument.

```
EBR user_id in_path [ctl_arg -ARG arg_1 arg_2 ...]
EGR id user_id [in_path][ctl_arg -ARG arg_1 arg_2...]
SG id user_id base_lvl [in_path] [ctl_arg -ARG arg_1 arg_2...]
```

The lead task can be the command processor or an applications task each with its own rules for parameter substitution.

1. If the command processor is the lead task, in\_path is substituted for all occurrences of &0, the first argument following -ARG for &1, the second for &2, etc...
2. If an applications task is the lead task, the first argument following -ARG is substituted for all occurrences of &0, the second for &1, etc...

**Example:**

The command line is:

```
EBR IW TRYEDT -OUT >SPD>LPT01 -WD ^ VOLA>JR -ARG TRYCOM 1 100
```

The contents of the noninteractive file TRYEDT are:

```
&PAEDIT COMMANDS ARE IN &0
ED
R &1
&2, &3P
QT
BYE
```

After substitution, the command lines contain:

```
&PAEDIT COMMANDS ARE IN TRYEDT
ED
R TRYCOM
1,100P
QT
BYE
```

The command processor is the lead task. TRYEDT is substituted for &0, TRYCOM for &1, 1 for &2 and 100 for &3.

**Example:**

This example illustrates the nesting of arguments in successive command lines. The task group activation command line is:

```
EGR AX SMITH.SERVICES MPG DATA -WD ^ VOLA>JG -ARG TEST -NL TEST.L
```

The first line of the noninteractive command-in file MPG\_DATA is:

```
EC ASM_LNK &1 >SPD>LPT01 &2 &3 START_AD DIR>SEC
```

The contents of the EC file ASM\_LNK.EC are:

```
ASSEM &1 -COUT &2 &3
&A
LINKER &1 -COUT &4
parameterized linker directives
```

After the first substitution, the EC command line is:

```
EC ASM_LNK TEST >SPD>LPT01 -NL TEST.L START_AD DIR>SEC
```

After substitution using the argument list in the EC command, the command lines in ASM\_LNK.EC are:

```
ASSEM TEST -COUT >SPD>LPT01 -NL
&A
LINKER TEST -COUT TEST.L
parameterized linker directives
```

A previous CG has created the AX task group whose lead task is the command processor. The additional arguments in the EGR command are substituted for the parameters in the command input file, MPG\_DATA.

After EGR is processed the command-in and user-in file is MPG\_DATA. After the EC line is processed command-in is ASM\_LNK.EC and user-in is still MPG\_DATA. Linker reads its directives from user-in. If these directive are to be parameterized they must be in the command-in file. The &A directive changes the input file read by Linker from MPG\_DATA to ASM\_LNK.EC and the arguments in the EC line apply to parameterized lines in the EC file ASM\_LNK.EC.

After the lines in ASM\_LNK.EC are processed, the command and user-in files revert back to MPG\_DATA. For parameterized command lines that follow the first EC line in MPG\_DATA the argument list is that of the EGR command.

## Listener Component and Login Capability

### INSTALLING A SYSTEM LOGIN CAPABILITY

System software includes a "listener" component that permits a user, with the LOGIN command, to gain access to the system from a user-designated noncommunications terminal (MDC-connected) or communications terminal (MLCP-connected). See the *Commands* manual for details.

To provide a system with login capability:

1. At configuration, provide enough memory pools, to be used as default pools, in the same number as the maximum number of concurrently logged-in users.
2. Before activating the listener component, create a terminals file that describes the login characteristics of each terminal to be used for login purposes. A terminal can (a) require a LOGIN command typein, (b) allow a user to type an abbreviation for the LOGIN command line, (c) immediately log in the terminal, without any typein, when it is ready or connected.
3. Activate the listener component as the lead task of a task group.

### MEMORY POOLS FOR LOGIN TASKS

At login, a user may either specify a memory pool or use a default pool. If an installation provides default pools, they must be defined at configuration in the same number as the maximum number of users who can concurrently gain system access through the login procedure. The pools can be defined as completely overlapping by specifying each pool in a separate MEMPOOL directive. If no memory pools are configured specifically for login users, a pool\_id must be specified in the LOGIN command line for each terminal.

The default pool\_id consists of two characters: the first is an alphabetic character specified by the user when listener is activated; the second is a system-appended character from 0 through 9 and A through Z in that order, i.e., the first character must be determined at configuration even though not used until listener activation. This first character must be unique, i.e., *not* used for any memory pool other than for login.

### TERMINAL LOGIN CHARACTERISTIC FILE

Each installation must create and name a terminals file that describes the login characteristics of each terminal to be monitored for system access requests. The file is created with the Editor and consists of variable-length records, whose arguments within a record are separated by one or more blank characters. The file consists of G-, T-, and A-type records and has the following layout:

G-Record (only one per file) [A-Records — one or more for all terminals]
T-Record — for a specified terminal [A-Records — one or more for the above terminal]
T-Record — for another specified terminal [A-Records — one or more for the above terminal]

### G-RECORD IN LOGIN FILE

There is one G-record in the login terminals file, in the format:

G base\_lvl max\_user

#### base\_lvl

Level, relative to the lowest numeric (highest priority) level not used by the system group, on which the lead task of groups spawned by listener for terminals, are to execute.

#### max\_user

Maximum number of concurrent logged-in users allowed on the system. This value does not include task groups created or spawned by commands other than LOGIN. Additional logins exceeding this limit are terminated with a 3915 error status return.

#### T-RECORD IN LOGIN FILE

There is one T-record in the terminal login file for each terminal on which a user may log in, in the format:

```
T dev_name [login_line]
```

#### dev\_name

Symbolic device name of the terminal, as specified at configuration.

#### login\_line

The login command line image (including the LOGIN or L characters) used instead of a user typein when a terminal is to be used for direct login.

#### A-RECORD IN LOGIN FILE

An A-record contains a character abbreviation and the associated LOGIN command line image that the listener will use when a user types in the abbreviation. A variable number of A-records may follow the G-record and/or any T-record. When a user enters an abbreviation, listener scans the A-records following the T-record for that terminal and if a match is found, uses that login line for logging in. If the abbreviation is not found, listener scans the A-records following the G-record for a match, and if a match is found, uses that login line for logging in. If no match is found, an error is displayed at the terminal. The format for the A-record is:

```
A abbrev login_line
```

#### abbrev

A 1-character abbreviation that a user can optionally type in when logging in on this terminal.

#### login\_line

The LOGIN command line image associated with the abbreviation.

#### LISTENER ACTIVATION

To provide terminal monitoring, the listener component must be activated as the lead task of a task group from the operator terminal any time after startup is complete and the configured system is operational. During activation of listener, none of the terminals to be monitored for a LOGIN command may be reserved. Once activated, listener cannot be turned off until the system is again started up.

Listener is activated with the CG (CREATE GROUP) and EGR (ENTER GROUP REQUEST) operator commands, or with an SG (SPAWN GROUP) operator command, using the arguments shown below in addition to those described for those commands in Section 4. These command formats are:

```
CG id base_lvl -EFN LISTENER -POOL id
```

```
EGR id user_id -OUT >SPD>CONSOLE -ARG [ { 'pathΔ' } ] [x] ["message"]
```

```
SG id user_id base_lvl -EFN LISTENER -POOL id -OUT >SPD>CONSOLE -ARG [ { 'pathΔ' } ] [x] ["message"]
```

{ 'pathΔ' }  
{ "pathΔ" }

Pathname of the terminals file, which lists the terminals on which users may log in, and which contains the terminal characteristics records.

The last character in the pathname must be a blank, and the entire pathname must be enclosed in either single or double quotes. An omitted (default) pathname must be written as a pair of enclosing single or double quotes ( ' ' ) or ( " " ), and results in the default pathname >SID>TERMINALS for use by listener.

x

The first character in the 2-character pool id and group id when default values are used. The second character, from 0 through 9 or A through Z, is appended when a task group is spawned as a result of the LOGIN command. When this argument is omitted, its default value is L.

When a user specifies a group id in LOGIN command or in a login\_line for a T-record or A-record, listener uses that as a group id instead of generating a group id.

"message"

The message-of-the-day, enclosed in quotes to provide for embedded blanks, which listener transmits to all login terminals for display.

### **DESIGNING THE LOGIN TERMINAL FILE**

For a terminal to have the direct login characteristic, the LOGIN command line image must be in the T-record for that terminal.

For a terminal to optionally accept abbreviations for LOGIN commands required A-records with the desired command line image and the absence of a login line in the T-record for that terminal. One or more abbreviations can be specified. The A-records following a T-record are associated only with that terminal. The A-records following the G-records allow all terminals to use those abbreviations for command lines. When the same abbreviation is used in an A-record following G-record, and in an A-record following a T-record, the command line image in the A-record following the T-record is used for the terminal.

### **TERMINAL STATE AFTER LISTENER IS ACTIVATED**

When first activated, listener performs specific operations affecting the state of a terminal, and again when the session terminates. The output on the terminal that a user sees and the state of the terminal depend on whether it is a noncommunications or a communications terminal.

### **NONCOMMUNICATIONS TERMINAL STATE WITH LISTENER**

If a terminal is not ready when listener is activated, no initial output messages from listener or the login component are displayed when the terminal comes on line.

When listener is activated:

1. If there are terminals online ready for direct login, they display the message-of-the-day. A task group is spawned for each such terminal, using the login\_line image contained in that terminal's T-record in the terminal login file. The lead task defined in the login line is executed. The application should display a prompter message to the terminal indicating that it is ready to accept input. When the lead task terminates, the message-of-the-day is displayed and a task group is immediately spawned again.
2. Terminals that require a user login display the message-of-the-day and the user login prompter message identifying the system and giving the date and time:

LOGIN system\_id yyyy/mm/dd hhmm:ss.t

The user can then type in the LOGIN command. When the lead task terminates, the message-of-the-day is displayed followed by the login prompter message.



## COMMUNICATIONS TERMINAL STATE WITH LISTENER

Although a communications terminal may not be ready when the listener is activated, listener displays a message when the terminal comes online. Otherwise, when listener is activated, the same operations are done for communications terminals as for noncommunications terminals described above.

When the lead task terminates:

1. A terminal connected by phone and with the hangup option, is disconnected. The user must dial in again to use the terminal.
2. A terminal connected through a modem by-pass or by phone without the hangup option, displays the message-of-the-day; either the login prompter message is displayed or, for a direct login, a login task group is spawned.

## CHANGING THE LOGIN MESSAGE OF THE DAY

After listener is activated, it places an operator response request to the operator terminal. The request number must be used in the response to listener from the operator terminal that changes the message-of-the-day. The message to the listener cannot exceed 63 characters and is in the form:

$\Delta$ msg\_no $\Delta$ message-of-the-day

# Appendix C

## System Halts

This appendix describes conditions causing system halts during system execution and startup.

### INSUFFICIENT MEMORY HALTS

During execution, the system halts in certain cases when it is out of memory resources, with the registers containing special information as indicated below.

- If the number of trap save areas (TSA) in the system is exhausted, the system halts on level 2 (status register = 4002). The contents of location 10 equal zero. Increase the TSAs using CLM SYS directive.
- If the number of indirect request blocks (IRBs) in the system is exhausted, the system halts on level 2 with ASCII IR in R1 (status register = 4002, D1 = 4952). Increase the IRBs using CLM SYS directive.

### STARTUP HALTS

The following conditions cause a halt during startup bootstrap and initialization.

#### BOOTSTRAP HALT CONDITIONS

During bootstrap, if the Halt option (xx04 or xx06) is requested

R1 will contain the bootstrap channel after halting

R2 will contain address mode flag 0 — SAF, 1 — LAF

R3 will contain disk type flag 0 — cartridge or storage module, 1 — diskette

Other halts during bootstrap routine are:

R1 = 1611

The next sector of the file (Z3EXECUTIVE (S/L)) being read is beyond the range of the last sector. No retry is possible.

R1 = 1612

The bound unit (Z3EXECUTIVE (S/L)) to be loaded was not found on the volume major directory. Check to see if proper disk is mounted. No retry is possible.

R1 = 1616

An I/O error has occurred. R6 contains the error bits of the hardware status word. Press Execute to retry the I/O function.

#### INITIALIZATION HALT CONDITIONS

The following conditions will result in a halt during initialization. The halt condition code is displayed in register R1. Any further breakdown of the error is displayed in the register indicated in the code explanations below, (which are *not* printed out).

\*

9902	ERROR ASSIGNING USER INPUT, R2 = ERROR
9903	ERROR SPAWNING CLM, R2 = ERROR
9905	ERROR REASSIGNING USER INTO CONSOLE, R2 = ERROR
9906	ERROR CHANGING SYSTEM DIRECTORY, R2 = ERROR
9907	ERROR CHANGING WORKING DIRECTORY, R2 = ERROR
9908	NO OP CONSOLE, EITHER LOCAL OR REMOTE
9909	AVAILABLE

9910 FILE NOT FOUND, B2 →FILE NAME  
9911 IO ERROR, R2 = ERROR, B5 →WHERE CALL WAS FROM  
9912 NO MEMORY FOR OPENING ROLLOUT FILE  
9913 ERROR OPENING ROLLOUT FILE, R2 = ERROR  
9914 ERROR DOING COMMAND IN, R2 = ERROR  
9915 ERROR DOING USER OUT, R2 = ERROR  
9916 ERROR CLOSING USER OUT, ERROR OUT, USER IN, COMMAND IN  
9917 FILE CONTAINS ILLEGAL REMOTE EXTENT RECORD  
9918 LRN 2 NOT LEFT AVAILABLE FOR MLCP OP CONSOLE'S ALTERNATE  
9921 NO MEMORY FOR INPUT BUFFER OVER 140 BYTES  
9922 PROBLEM SPAWNING TASK WHICH DOES PCL FOR EC START\_UP  
9923 PROBLEM IN OPENING THE ERROR MESSAGE LIBRARY  
9940 OUTPUT BY OIM, MSG TO GO OUT, BUT NO OP CONSOLE YET

## Appendix D

# ASCII Character Set and Hexadecimal Equivalents

This appendix contains lists of ASCII control characters and special graphic characters. Table D-1 shows the ASCII/hexadecimal equivalent values.

### ASCII CONTROL CHARACTERS

ACK	Acknowledge	FF	Form Feed
BEL	Bell	FS	Field Separator
BS	Backspace	GS	Group Separator
CAN	Cancel	HT	Horizontal Tab
CR	Carriage Return	LF	Line Feed
DC1	Device Control 1	NAK	Negative Acknowledgment
DC2	Device Control 2	NUL	Null
DC3	Device Control 3	RS	Reader Stop
DC4	Device Control 4	SI	Shift In
DEL	Delete	SO	Shift Out
DLE	Data Link Escape	SOH	Start of Heading
EM	End of Medium	SP	Space
ENQ	Enquiry	STX	Start of Text
EOT	End of Transmission	SUB	Substitute
ESC	Escape	SYN	Synchronous Idle
ETB	End of Transmission Block	US	Unit Separator
ETX	End of Text	VT	Vertical Tab

### ASCII SPECIAL GRAPHIC CHARACTERS

>	Greater-than 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	}	Closing Brace
;	Semicolon	\	Reverse Slant
/	Slash	[	Opening Bracket
	Vertical Line	]	Closing Bracket
,	Comma	%	Percent
-	Underscore	^	Circumflex

**TABLE D-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

## INDEX

- ABORT
  - ABORT BATCH REQUEST (ABR) COMMAND, 4-7
  - ABORT BATCH (ABORT\_BATCH) COMMAND, 4-6
  - ABORT GROUP REQUEST (AGR) COMMAND, 4-9
  - ABORT GROUP (ABORT\_GROUP) COMMAND, 4-8
- ABSOLUTE PATHNAME
  - ABSOLUTE PATHNAME, 1-4
- ACCESS
  - FILE ACCESS PROTECTION, 4-30
  - LOGIN COMMAND TO ACCESS SYSTEM, B-1
- ACTIVATE
  - ACTIVATE BATCH (ACTB) COMMAND, 4-10
  - ACTIVATE GROUP (ACTG) COMMAND, 4-11
- A-RECORD
  - LOGIN FILE A-RECORD, B-2
- AMPERSAND
  - AMPERSAND (&) PARAMETER SUBSTITUTION, A-1
- ARG ARGUMENTS
  - ARG ARGUMENTS PASSED TO TASK, A-1
  - COMMAND LINE (ARG) ARGUMENTS, A-1
- ARGUMENT
  - ARGUMENT NESTING, A-2
  - CONTROL ARGUMENT, 4-3
  - INPUT-COMMAND ARGUMENT, 4-3
  - KEYWORD ARGUMENT, 4-3
  - POSITIONAL ARGUMENT, 4-3
- ASCII
  - ASCII AND HEXADECIMAL EQUIVALENT CHARACTERS, D-2
  - ASCII CHARACTERS, D-1
- BATCH COMMANDS
  - ABORT BATCH REQUEST (ABR) COMMAND, 4-7
  - ABORT BATCH (ABORT\_BATCH) COMMAND, 4-6
  - ACTIVATE BATCH (ACTB) COMMAND, 4-10
  - CREATE BATCH (CB) COMMAND, 4-14
  - DELETE BATCH (DB) COMMAND, 4-17
  - ENTER BATCH REQUEST (EBR) COMMAND, 4-19
  - SUSPEND BATCH (SSPB) COMMAND, 4-39
- BOUND UNIT COMMANDS
  - LOAD SHARABLE BOUND UNIT (LOAD) COMMAND, 4-28
  - UNLOAD SHARABLE BOUND UNIT (UNLD) COMMAND, 4-42
- BREAK
  - BREAK COMMAND EXAMPLES, 5-2
- BREAK (CONT'D)
  - BREAK (TASK INTERRUPT) PROCEDURES, 5-1
  - BYE RESPONSE COMMAND TO BREAK, 5-2
  - OIM DIRECTIVE FOR BREAK (TASK INTERRUPT), 3-5
  - PROGRAM INTERRUPT (PI) RESPONSE COMMAND TO BREAK, 5-2
  - START (SR) RESPONSE COMMAND TO BREAK, 5-2
  - UNWIND (UW) RESPONSE COMMAND TO BREAK, 5-2
- CHARACTER
  - CHARACTER DELETE IN INPUT MESSAGE, 3-4
  - CONTROL CHARACTER AS DATA CHARACTER, 3-4
- CHARACTERS
  - ASCII AND HEXADECIMAL EQUIVALENT CHARACTERS, D-2
  - ASCII CHARACTERS, D-1
- CODES, HALT
  - INITIALIZATION HALT R1-REGISTER CODES, C-1
- COMMAND
  - BREAK COMMAND EXAMPLES, 5-2
  - COMMAND LINE PARAMETER SUBSTITUTION, A-1
  - COMMAND LINE (ARG) ARGUMENTS, A-1
  - COMMAND PROCESSOR EXECUTION, 4-19
  - EC FILE EXECUTION COMMAND, A-2
  - EXECUTION COMMAND (EC) CONTROL DIRECTIVES, 4-22
  - LISTENER ACTIVATION COMMAND, 2-2
  - LOGIN COMMAND TO ACCESS SYSTEM, B-1
  - OPERATOR COMMAND FUNCTIONS AND NAMES (TBL), 4-4
- COMMAND-IN FILE
  - COMMAND PROCESSOR COMMAND-IN FILE, 4-2
- COMMAND NAMES
  - ABORT\_BATCH, 4-6
  - ABORT\_GROUP, 4-8
  - ABR, 4-7
  - ACTB, 4-10
  - ACTG, 4-11
  - AGR, 4-9
  - CB, 4-14
  - CG, 4-15
  - CSD, 4-12
  - CWD, 4-13
  - DB, 4-17
  - DG, 4-18
  - EBR, 4-19
  - EC, 4-22
  - EGR, 4-20
  - FD, 4-25
  - LOAD, 4-28

INDEX

COMMAND NAMES (CONT'D)

LSR, 4-26  
LWD, 4-27  
MF, 4-30  
MSW, 4-29  
RAS, 4-31  
SD, 4-32  
SG, 4-33  
SSPB, 4-39  
SSPG, 4-40  
STG, 4-35  
STS, 4-37  
TIME, 4-41  
UNLD, 4-42  
WCSLD, 4-43

COMMAND PROCESSOR FILES

COMMAND-IN FILE, 4-2  
ERROR-OUT FILE, 4-2  
INPUT/OUTPUT FILE, 4-2  
OPERATOR-OUT FILE, 4-2  
USER-OUT FILE, 4-2

COMMANDS, FUNCTIONS AND NAMES

ABORT BATCH (ABORT BATCH), 4-6  
ABORT BATCH REQUEST (ABR), 4-7  
ABORT GROUP (ABORT GROUP), 4-8  
ABORT GROUP REQUEST (AGR), 4-9  
ACTIVATE BATCH (ACTB), 4-10  
ACTIVATE GROUP (ACTG), 4-11  
CHANGE SYSTEM DIRECTORY (CSD), 4-12  
CHANGE WORKING DIRECTORY (CWD), 4-13  
CREATE BATCH (CB), 4-14  
CREATE GROUP (CG), 4-15  
DELETE BATCH (DB), 4-17  
DELETE GROUP (DG), 4-18  
ENTER BATCH REQUEST (EBR), 4-19  
ENTER GROUP REQUEST (EGR), 4-20  
EXECUTION COMMAND (EC), 4-22  
FILE OUT (FO), 4-25  
LIST SEARCH RULES (LSR), 4-26  
LIST WORKING DIRECTORY (LWD), 4-27  
LOAD SHARABLE BOUND UNIT (LOAD),  
4-28  
MODIFY EXTERNAL SWITCHES (MSW), 4-29  
MODIFY FILE (MF), 4-30  
REASSIGN (RAS), 4-31  
SET DATE (SD), 4-32  
SPAWN GROUP (SG), 4-33  
STATUS GROUP (STG), 4-35  
STATUS SYSTEM (STS), 4-37  
SUSPEND BATCH (SSPB), 4-39  
SUSPEND GROUP (SSPG), 4-40  
TIME (TIME), 4-41  
UNLOAD SHARABLE BOUND UNIT (UNLD),  
4-42  
WRITABLE CONTROL STORE LOAD (WCSLD),  
4-43

COMMUNICATIONS TERMINAL

COMMUNICATIONS TERMINAL WITH  
LISTENER, B-4

CONCURRENCY, FILE

FILE CONCURRENCY, 4-3

CONCURRENCY, FILE (CONT'D)

FILE CONCURRENCY INFORMATION, 4-36

CONTROL

ASCII CONTROL CHARACTERS, D-1  
CONTROL ARGUMENT, 4-3  
CONTROL CHARACTER AS DATA  
CHARACTER, 3-4  
EXECUTION COMMAND (EC) CONTROL  
DIRECTIVES, 4-22  
INPUT MESSAGE CONTROL FUNCTIONS/  
FORMAT (TBL), 3-4  
OPERATOR COMMAND FOR EXECUTION  
CONTROL, 4-1  
OPERATOR COMMAND FOR FILE  
DIRECTORY DEVICE CONTROL, 4-1

CORRECTION, MESSAGE

CORRECTION TO INPUT MESSAGE, 3-4

CREATE

CREATE BATCH (CB) COMMAND, 4-14  
CREATE GROUP (CG) COMMAND, 4-15

DATE, SET

SET DATE (SD) COMMAND, 4-32

DEFINITION

DEFINITION OF A FILE, 1-2  
DEFINITION OF DIRECTORY, 1-2  
OPERATOR COMMAND DEFINITION, 4-1  
TASK GROUP DEFINITION, 1-1

DELETE

CHARACTER DELETE IN INPUT MESSAGE,  
3-4  
DELETE BATCH (DB) COMMAND, 4-17  
DELETE GROUP COMMAND (DG), 4-18  
INPUT MESSAGE DELETE, 3-4

DEVICE

DEVICE FILE PATHNAME, 1-3  
DEVICE PATHNAME, 1-3  
DISK DEVICE PATHNAME, 1-3  
OFFLINE UNIT RECORD DEVICE TIMEOUT,  
3-6  
OPERATOR COMMAND FOR FILE DIRECTORY  
DEVICE CONTROL, 4-1  
PERIPHERAL DEVICE EXCHANGE, 4-31  
PERIPHERAL DEVICE STATUS  
INFORMATION, 4-38  
TAPE DEVICE PATHNAME, 1-3

DIALOG, OPERATOR/SYSTEM

OPERATOR AND SYSTEM DIALOG EXAMPLE  
(FIG), 3-7  
SAMPLE OPERATOR AND SYSTEM DIALOG,  
3-6

DISK PATHNAME

DISK DEVICE PATHNAME, 1-3

EC

EC FILE EXECUTION COMMAND, A-2  
EXECUTION COMMAND (EC) COMMAND, 4-22

INDEX

- ENTER
  - ENTER BATCH REQUEST (EBR) COMMAND, 4-19
  - ENTER GROUP REQUEST (EGR) COMMAND, 4-20
- ERROR MESSAGES
  - OPERATOR INTERFACE MANAGER (OIM) ERROR MESSAGES, 3-5
- ERROR-OUT
  - COMMAND PROCESSOR ERROR-OUR FILE, 4-2
- EXCHANGE, DEVICE
  - PERIPHERAL DEVICE EXCHANGE, 4-31
- EXECUTION
  - EC FILE EXECUTION COMMAND, A-2
  - EXECUTION COMMAND (EC) COMMAND, 4-22
  - LEAD TASK EXECUTION, 4-20
  - OPERATOR COMMAND FOR EXECUTION CONTROL, 4-1
- FILE
  - COMMAND PROCESSOR COMMAND-IN FILE, 4-2
  - COMMAND PROCESSOR ERROR-OUT FILE, 4-2
  - COMMAND PROCESSOR INPUT/OUTPUT FILE, 4-2
  - COMMAND PROCESSOR OPERATOR-OUT FILE, 4-2
  - COMMAND PROCESSOR USER-OUT FILE, 4-2
  - DEFINITION OF A FILE, 1-2
  - DEVICE FILE PATHNAME, 1-3
  - EC FILE EXECUTION COMMAND, A-2
  - FILE ACCESS PROTECTION, 4-30
  - FILE CONCURRENCY, 4-3
  - FILE CONCURRENCY INFORMATION, 4-36
  - FILE OUT (FO) COMMAND, 4-25
  - FILE STATUS INFORMATION, 4-36
  - MODIFY FILE (MF) COMMAND, 4-30
  - OPERATOR COMMAND FOR FILE DIRECTORY DEVICE CONTROL, 4-1
- FILE NAME
  - FILE NAME CONSTRUCTION, 1-2
- FILE SYSTEM
  - FILE SYSTEM, 1-2
  - FILE SYSTEM DIRECTORY, 1-2
  - FILE SYSTEM PATHNAME, 1-2
- FIRMWARE
  - FIRMWARE OBJECT FILES, 4-43
- FORMAT
  - INPUT MESSAGE FORMAT, 3-3
  - INPUT COMMAND LINE FORMAT, 4-3
- GET
  - OIM DIRECTIVE GET MESSAGE LIST, 3-5
- G-RECORD IN LOGIN
  - LOGIN FILE G-RECORD, B-1
- GROUP
  - ABORT GROUP REQUEST (AGR) COMMAND, 4-9
  - ABORT GROUP (ABORT\_GROUP) COMMAND, 4-8
  - ACTIVATE GROUP (ACTG) COMMAND, 4-11
  - CREATE GROUP (CG) COMMAND, 4-15
  - DELETE GROUP COMMAND, 4-18
  - ENTER GROUP REQUEST (EGR) COMMAND, 4-20
  - GROUP REQUEST PARAMETER SUBSTITUTION, A-3
  - OIM DIRECTIVE CHANGE DEFAULT TASK GROUP, 3-5
  - SPAWN GROUP (SG) COMMAND, 4-33
  - STATUS GROUP (STG) COMMAND, 4-35
  - SUSPEND GROUP (SSPG) COMMAND, 4-40
  - TASK GROUP DEFINITION, 1-1
  - TASK GROUP IDENTIFICATION (GROUP ID), 1-1
  - TASK GROUP STATUS INFORMATION, 4-35, 4-37
  - TASK GROUP USER IDENTIFICATION (USER-ID), 1-1
- HALTS
  - BOOTSTRAP HALTS, C-1
  - INITIALIZATION HALTS, C-1
  - STARTUP HALTS, C-1
  - SYSTEM HALTS, C-1
- HEXADECIMAL
  - ASCII AND HEXADECIMAL EQUIVALENT CHARACTERS, D-2
- IDENTIFICATION
  - TASK GROUP IDENTIFICATION (GROUP ID), 1-1
  - TASK GROUP USER IDENTIFICATION (USER-ID), 1-1
- INITIALIZATION
  - INITIALIZATION HALT R1-REGISTER CODES, C-1
  - INITIALIZATION HALTS, C-1
- INPUT MESSAGE
  - CHARACTER DELETE IN INPUT MESSAGE, 3-4
  - CORRECTION TO INPUT MESSAGE, 3-4
  - INPUT MESSAGE CONTROL FUNCTIONS/FORMAT (TBL), 3-4
  - INPUT MESSAGE DELETE, 3-4
  - INPUT MESSAGE FORMAT, 3-3
  - INPUT MESSAGE LENGTH, 3-3
  - INPUT MESSAGE OIM, 3-1
  - INPUT (OPERATOR RESPONSE) MESSAGE, 3-3
- INPUT COMMAND
  - INPUT COMMAND ARGUMENT, 4-3
  - INPUT COMMAND LINE, 4-3



INDEX

INPUT COMMAND (CONT'D)  
 INPUT COMMAND LINE FORMAT, 4-3  
 INPUT COMMAND SPACES IN COMMAND  
 LINE, 4-3

INPUT/OUTPUT FILE  
 COMMAND PROCESSOR INPUT/OUTPUT  
 FILE, 4-2

INTERRUPT  
 BREAK (TASK INTERRUPT) PROCEDURES,  
 5-1  
 OIM DIRECTIVE FOR BREAK (TASK  
 INTERRUPT), 3-5  
 PROGRAM INTERRUPT (PI) RESPONSE  
 COMMAND TO BREAK, 5-2  
 TASK INTERRUPT (BREAK), 5-1

KEYWORD ARGUMENT  
 KEYWORD ARGUMENT, 4-3

LFN  
 LOGICAL FILE NUMBER (LFN), 1-2

LINE, COMMAND  
 COMMAND LINE PARAMETER  
 SUBSTITUTION, A-1  
 COMMAND LINE (ARG) ARGUMENTS, A-1  
 INPUT COMMAND LINE, 4-3  
 INPUT COMMAND LINE FORMAT, 4-3  
 INPUT COMMAND SPACES IN COMMAND  
 LINE, 4-3

LIST  
 LIST SEARCH RULES (LSR) COMMAND,  
 4-26  
 LIST WORKING DIRECTORY (LWD)  
 COMMAND, 4-27  
 OIM DIRECTIVE, GET MESSAGE LIST,  
 3-5

LISTENER  
 ACTIVATING LISTENER, 2-2  
 CHANGE LISTENER MESSAGE-OF-THE-DAY,  
 B-4  
 COMMUNICATIONS TERMINAL WITH  
 LISTENER, B-4  
 LISTENER ACTIVATION, B-2  
 LISTENER COMPONENT, B-1  
 LISTENER MESSAGE-OF-THE-DAY, B-3  
 NONCOMMUNICATIONS TERMINAL WITH  
 LISTENER, B-3  
 TERMINAL STATE WITH LISTENER, B-3

LOAD  
 LOAD SHARABLE BOUND UNIT (LOAD)  
 COMMAND, 4-28  
 WRITABLE CONTROL STORE LOAD (WCSLD)  
 COMMAND, 4-43

LOADER  
 WRITABLE CONTROL STORE (WCS) LOADER,  
 4-43

LOGICAL  
 LOGICAL FILE NUMBER (LFN), 1-2  
 LOGICAL RESOURCE NUMBER (LRN), 1-2

LOGIN  
 ACTIVATING LOGIN CAPABILITY, 2-2  
 LOGIN COMMAND TO ACCESS SYSTEM,  
 B-1  
 LOGIN FILE A-RECORD, B-2  
 LOGIN FILE G-RECORD, B-1  
 LOGIN FILE RECORD, B-2  
 LOGIN FILE T-RECORD, B-2  
 LOGIN TERMINAL FILE ATTRIBUTES,  
 B-1  
 LOGIN TERMINAL FILE DESIGN, B-3  
 MEMORY-POOLS FOR LOGIN TASKS, B-1

LRN  
 LOGICAL RESOURCE NUMBER (LRN), 1-2

MANAGER, OPERATOR INTERFACE  
 OPERATOR INTERFACE MANAGER (OIM),  
 3-1  
 OPERATOR INTERFACE MANAGER (OIM)  
 ERROR MESSAGES, 3-5

MEMORY POOLS  
 MEMORY POOLS FOR LOGIN TASKS, B-1

MESSAGE  
 CHARACTER DELETE IN INPUT MESSAGE,  
 3-4  
 CORRECTION TO INPUT MESSAGE, 3-4  
 INPUT MESSAGE CONTROL FUNCTIONS/  
 FORMAT (TBL), 3-4  
 INPUT MESSAGE DELETE, 3-4  
 INPUT MESSAGE FORMAT, 3-3  
 INPUT MESSAGE LENGTH, 3-3  
 INPUT MESSAGE OIM, 3-1  
 INPUT (OPERATOR RESPONSE) MESSAGE,  
 3-3  
 MESSAGE ENTRY DURING OUTPUT, 3-4  
 MESSAGE FUNCTIONS, 3-1  
 MESSAGE TO HALT OUTPUT, 3-4  
 OIM DIRECTIVE GET MESSAGE LIST,  
 3-5  
 OUTPUT MESSAGE NUMBER, 3-2  
 OUTPUT MESSAGE OIM, 3-1  
 OUTPUT MESSAGE PREFIX, 3-2  
 OUTPUT MESSAGE QUEUE, 3-2

MESSAGE-OF-THE-DAY, LISTENER  
 CHANGE LISTENER MESSAGE-OF-THE-  
 DAY, B-4  
 LISTENER MESSAGE-OF-THE-DAY, B-3

MESSAGES  
 OPERATOR INTERFACE MANAGER (OIM)  
 ERROR MESSAGES, 3-5

MODIFY  
 MODIFY EXTERNAL SWITCHES (MSW)  
 COMMAND, 4-29  
 MODIFY FILE (MF) COMMAND, 4-30

INDEX

NESTING, ARGUMENT  
 ARGUMENT NESTING, A-2

NUMBER  
 LOGICAL FILE NUMBER (LFN), 1-2  
 LOGICAL RESOURCE NUMBER (LRN), 1-2  
 OUTPUT MESSAGE NUMBER, 3-2

OIM (OPERATOR INTERFACE MANAGER)  
 INPUT MESSAGE OIM, 3-1  
 OIM DIRECTIVE CHANGE DEFAULT TASK GROUP, 3-5  
 OIM DIRECTIVE CHANGE OUTPUT PACING RATE, 3-5  
 OIM DIRECTIVE FOR BREAK (TASK INTERRUPT), 3-5  
 OIM DIRECTIVE GET MESSAGE LIST, 3-5  
 OPERATOR INTERFACE MANAGER (OIM), 3-1  
 OPERATOR INTERFACE MANAGER (OIM) ERROR MESSAGES, 3-5  
 OUTPUT MESSAGE OIM, 3-1

OPERATOR  
 INCREASE OPERATOR TERMINAL RESPONSE SPEED, 3-6  
 INPUT (OPERATOR RESPONSE) MESSAGE, 3-3  
 OPERATOR AND SYSTEM DIALOG EXAMPLE (FIG), 3-7  
 OPERATOR COMMAND FUNCTIONS AND NAMES (TBL), 4-4  
 OPERATOR COMMANDS, 4-1  
 OPERATOR INTERFACE MANAGER (OIM), 3-1  
 OPERATOR INTERFACE MANAGER (OIM) ERROR MESSAGES, 3-5  
 OPERATOR OIM DIRECTIVE MESSAGES, 3-5  
 OPERATOR TERMINAL, 3-1  
 SAMPLE OPERATOR AND SYSTEM DIALOG, 3-6

OPERATOR COMMAND  
 OPERATOR COMMAND DEFINITION, 4-1  
 OPERATOR COMMAND FOR EXECUTION CONTROL, 4-1  
 OPERATOR COMMAND FOR FILE DIRECTORY DEVICE CONTROL, 4-1  
 OPERATOR COMMAND TO MONITOR SYSTEM, 4-2

OPERATOR-OUT FILE  
 COMMAND PROCESSOR OPERATOR-OUT FILE, 4-2

OUTPUT  
 MESSAGE ENTRY DURING OUTPUT, 3-4  
 MESSAGE TO HALT OUTPUT, 3-4  
 OIM DIRECTIVE CHANGE OUTPUT PACING RATE, 3-5  
 OUTPUT MESSAGE NUMBER, 3-2  
 OUTPUT MESSAGE OIM, 3-1  
 OUTPUT MESSAGE PREFIX, 3-2  
 OUTPUT MESSAGE QUEUE, 3-2

PACING RATE  
 OIM DIRECTIVE CHANGE OUTPUT PACING RATE, 3-5

PARAMETER  
 AMPERSAND (&) PARAMETER SUBSTITUTION, A-1  
 COMMAND LINE PARAMETER SUBSTITUTION, A-1  
 GROUP REQUEST PARAMETER SUBSTITUTION, A-3

PATHNAME  
 ABSOLUTE PATHNAME, 1-4  
 DEVICE FILE PATHNAME, 1-3  
 DEVICE PATHNAME, 1-3  
 DISK DEVICE PATHNAME, 1-3  
 FILE SYSTEM PATHNAME, 1-2  
 PATHNAME CONSTRUCTION, 1-2  
 PATHNAME SYMBOLS, 1-3  
 RELATIVE PATHNAME, 1-4  
 TAPE DEVICE PATHNAME, 1-3

PERIPHERAL DEVICE  
 PERIPHERAL DEVICE EXCHANGE, 4-31  
 PERIPHERAL DEVICE STATUS INFORMATION, 4-38

PI  
 PROGRAM INTERRUPT (PI) RESPONSE COMMAND TO BREAK, 5-2

POSITIONAL ARGUMENT  
 POSITIONAL ARGUMENT, 4-3

PREFIX, MESSAGE  
 OUTPUT MESSAGE PREFIX, 3-2

PROCESSOR, COMMAND  
 COMMAND PROCESSOR EXECUTION, 4-19

PROGRAM  
 PROGRAM INTERRUPT (PI) RESPONSE COMMAND TO BREAK, 5-2

PROTECTION, FILE ACCESS  
 FILE ACCESS PROTECTION, 4-30

QUEUE, MESSAGE  
 OUTPUT MESSAGE QUEUE, 3-2

REASSIGN  
 REASSIGN (RAS) COMMAND, 4-31

RECORD  
 LOGIN FILE RECORD, B-2  
 OFFLINE UNIT RECORD DEVICE TIMEOUT, 3-6

RELATIVE PATHNAME  
 RELATIVE PATHNAME, 1-4

REQUEST  
 ABORT BATCH REQUEST (ABR) COMMAND, 4-7

INDEX

REQUEST (CONT'D)  
 ABORT GROUP REQUEST (AGR) COMMAND, 4-9  
 ENTER BATCH REQUEST (EBR) COMMAND, 4-19  
 ENTER GROUP REQUEST (EGR) COMMAND, 4-20  
 GROUP REQUEST PARAMETER SUBSTITUTION, A-3

RESOURCE, LOGICAL NUMBER  
 LOGICAL RESOURCE NUMBER (LRN), 1-2

RESPONSE  
 INCREASE OPERATOR TERMINAL RESPONSE SPEED, 3-6  
 INPUT (OPERATOR RESPONSE) MESSAGE, 3-3  
 SYSTEM STARTUP RESPONSE, 2-2

RESPONSE COMMANDS TO BREAK  
 BYE RESPONSE, 5-2  
 PROGRAM INTERRUPT (PI) RESPONSE, 5-2  
 START (SR) RESPONSE, 5-2  
 UNWIND (UW) RESPONSE, 5-2

ROOT DIRECTORY  
 ROOT DIRECTORY, 1-2

RULES, SEARCH  
 LIST SEARCH RULES (LSR) COMMAND, 4-26

SEARCH, LIST  
 LIST SEARCH RULES (LSR) COMMAND, 4-26

SET DATE  
 SET DATE (SD) COMMAND, 4-32

SHARABLE BOUND UNIT COMMANDS  
 LOAD SHARABLE BOUND UNIT (LOAD) COMMAND, 4-28  
 UNLOAD SHARABLE BOUND UNIT (UNLD) COMMAND, 4-42

SPACES IN COMMAND LINE  
 INPUT COMMAND SPACES IN COMMAND LINE, 4-3

SPAWN  
 SPAWN GROUP (SG) COMMAND, 4-33

SPEED, OPERATOR TERMINAL  
 INCREASE OPERATOR TERMINAL RESPONSE SPEED, 3-6

STARTUP  
 STARTUP HALTS, C-1  
 SYSTEM STARTUP, 2-1  
 SYSTEM STARTUP RESPONSE, 2-2

STATUS  
 FILE STATUS INFORMATION, 4-36  
 PERIPHERAL DEVICE STATUS INFORMATION, 4-38

STATUS (CONT'D)  
 STATUS GROUP (STG) COMMAND, 4-35  
 STATUS SYSTEM (STS) COMMAND, 4-37  
 TASK GROUP STATUS INFORMATION, 4-35, 4-37  
 TASK STATUS INFORMATION, 4-35

STORE, WRITABLE CONTROL (WCS)  
 WRITABLE CONTROL STORE LOAD (WCSLD) COMMAND, 4-43  
 WRITABLE CONTROL STORE (WCS) LOADER, 4-43

SUBSTITUTION, PARAMETER  
 AMPERSAND (&) PARAMETER SUBSTITUTION, A-1  
 COMMAND LINE PARAMETER SUBSTITUTION, A-1  
 GROUP REQUEST PARAMETER SUBSTITUTION, A-3

SUSPEND  
 SUSPEND BATCH (SSPB) COMMAND, 4-39  
 SUSPEND GROUP (SSPG) COMMAND, 4-40

SWITCHES, MODIFY  
 MODIFY EXTERNAL SWITCHES (MSW) COMMAND, 4-29

SYMBOLS, PATHNAME  
 PATHNAME SYMBOLS, 1-3

SYSTEM  
 CHANGE SYSTEM DIRECTORY (CSD) COMMAND, 4-12  
 STATUS SYSTEM (STS) COMMAND, 4-37  
 SYSTEM HALTS, C-1  
 SYSTEM STARTUP, 2-1  
 SYSTEM STARTUP RESPONSE, 2-2

TAPE DEVICE PATHNAME  
 TAPE DEVICE PATHNAME, 1-3

TASK  
 ARG ARGUMENTS PASSED TO TASK, A-1  
 BREAK (TASK INTERRUPT) PROCEDURES, 5-1  
 LEAD TASK EXECUTION, 4-20  
 OIM DIRECTIVE CHANGE DEFAULT TASK GROUP, 3-5  
 OIM DIRECTIVE FOR BREAK (TASK INTERRUPT), 3-5  
 TASK GROUP DEFINITION, 1-1  
 TASK GROUP IDENTIFICATION (GROUP ID), 1-1  
 TASK GROUP STATUS INFORMATION, 4-35, 4-37  
 TASK GROUP USER IDENTIFICATION (USER-ID), 1-1  
 TASK INTERRUPT (BREAK), 5-1  
 TASK STATUS INFORMATION, 4-35

TERMINAL  
 COMMUNICATIONS TERMINAL WITH LISTENER, B-4

INDEX

TERMINAL (CONT'D)  
INCREASE OPERATOR TERMINAL RESPONSE  
SPEED, 3-6  
LOGIN TERMINAL FILE ATTRIBUTES, B-1  
NONCOMMUNICATIONS TERMINAL WITH  
LISTENER, B-3  
OPERATOR TERMINAL, 3-1  
TERMINAL STATE WITH LISTENER, B-3

TIME  
TIME (TIME) COMMAND, 4-41

TIMEOUT  
OFFLINE UNIT RECORD DEVICE  
TIMEOUT, 3-6

T-RECORD  
LOGIN FILE T-RECORD, B-2

UNLOAD  
UNLOAD SHARABLE BOUND UNIT (UNLD)  
COMMAND, 4-42

UNWIND  
UNWIND (UW) RESPONSE COMMAND TO  
BREAK, 5-2

USER-ID  
TASK GROUP USER IDENTIFICATION  
(USER-ID), 1-1

USER-OUT FILE  
COMMAND PROCESSOR USER-OUT FILE,  
4-2

WCS  
WRITABLE CONTROL STORE (WCS) LOADER,  
4-43

WORKING DIRECTORY  
CHANGE WORKING DIRECTORY (CWD)  
COMMAND, 4-13  
LIST WORKING DIRECTORY (LWD)  
COMMAND, 4-27  
WORKING DIRECTORY, 1-4

WRITABLE CONTROL STORE (WCS)  
WRITABLE CONTROL STORE LOAD (WCSLD)  
COMMAND, 4-43  
WRITABLE CONTROL STORE (WCS) LOADER,  
4-43



C

.

.

C

.

.

C



.

.



.

.



**HONEYWELL INFORMATION SYSTEMS**  
Technical Publications Remarks Form

CUT ALONG LINE

TITLE **SERIES 60 (LEVEL 6) GCOS  
MOD 400 OPERATOR'S GUIDE**

ORDER NO. **CB24, REV. 0**


DATED **JANUARY 1978**

**ERRORS IN PUBLICATION**

[Empty box for errors in publication]

**SUGGESTIONS FOR IMPROVEMENT TO PUBLICATION**

[Empty box for 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 \_\_\_\_\_  
TITLE \_\_\_\_\_  
COMPANY \_\_\_\_\_  
ADDRESS \_\_\_\_\_  
\_\_\_\_\_

DATE \_\_\_\_\_



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

FIRST CLASS  
PERMIT NO. 39531  
WALTHAM, MA  
02154

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

Postage Will Be Paid By:

HONEYWELL INFORMATION SYSTEMS  
200 SMITH STREET  
WALTHAM, MA 02154

ATTENTION: PUBLICATIONS, MS 486

**Honeywell**

CUT ALONG LINE

FOLD ALONG LINE

FOLD ALONG LINE



.

.



.

.



# 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.

21149, 3678, Printed in U.S.A.

CB24, Rev. 0