SERIES 60 (LEVEL 6)

GCOS 6 MOD 400 OPERATOR'S GUIDE ADDENDUM A

SUBJECT

Changes and Additions to the Manual

SPECIAL INSTRUCTIONS

Insert attached pages into Revision 0 of the manual dated January 1978 (see Collating Instructions). Change bars indicate new and changed information; asterisks denote deletions.

Note:

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

SOFTWARE SUPPORTED

This update supports Release 0110 of the Series 60 (Level 6) GCOS 6 MOD 400 software system. For any later release of MOD 400 software, see the Manual Directory of the latest *System Concepts* manual to ascertain whether this update supports that release.

ORDER NUMBER

CB24A, Rev. 0

June 1978

21005
3678
Printed in U.S.A.

Honeywell

COLLATING INSTRUCTIONS

Remove	Insert
	Manual directory/blank
v/vi	v/vi
1-3/1-4	1-3/1-4
2-1/2-2	2-1/2-2
3-5/3-6	3-5/3-6
4-3/4-4	4-3/4-4
4-5/4-6	4-5/4-6
——	4-16.1/blank
——	4-17.1/blank
4-21/4-22	4-21/4-22
4-23/4-24	4-23/4-24
4-25/4-26	4-25/4-25.1
	4-25.2/4-25.3
	blank/4-26
	4-27/4-28
4-31/4-32	4-31/4-31.1
	4-31.2/4-31.7
	blank/4-32
4-43/4-44	4-43/4-44
B-1/B-2	B-1/B-2
B-3/B-4	B-3/B-4

MANUAL DIRECTORY

The following publications comprise the GCOS 6 manual set. The Manual Directory in the latest GCOS 6 MOD 400 Systems Concepts manual (Order No. CB20) lists the current revision number and addenda (if any) for each manual in the set.

Order No.	*Manual Title*
CB01	*GCOS 6 Program Preparation*
CB02	*GCOS 6 Commands*
CB03	*GCOS 6 Communications Processing*
CB04	*GCOS 6 Sort/Merge*
CB05	*GCOS 6 Data File Organizations and Formats*
CB06	*GCOS 6 System Messages*
CB07	*GCOS 6 Assembly Language Reference*
CB08	*GCOS 6 System Service Macro Calls*
CB09	*GCOS 6 RPG Reference*
CB10	*GCOS 6 Intermediate COBOL Reference*
CB20	*GCOS 6 MOD 400 System Concepts*
CB21	*GCOS 6 MOD 400 Program Execution and Checkout*
CB22	*GCOS 6 MOD 400 Programmer's Guide*
CB23	*GCOS 6 MOD 400 System Building*
CB24	*GCOS 6 MOD 400 Operator's Guide*
CB25	*GCOS 6 MOD 400 FORTRAN Reference*
CB26	*GCOS 6 MOD 400 Entry-Level COBOL Reference*
CB27	*GCOS 6 MOD 400 Programmer's Pocket Guide*
CB28	*GCOS 6 MOD 400 Master Index*
CB30	*Remote Batch Facility User's Guide*
CB31	*Data Entry Facility User's Guide*
CB32	*Data Entry Facility Operator's Quick Reference Guide*
CB33	*Level 6/Level 6 File Transmission Facility User's Guide*
CB34	*Level 6/Level 62 File Transmission Facility User's Guide*
CB35	*Level 6/Level 64 (Native) File Transmission Facility User's Guide*
CB36	*Level 6/Level 66 File Transmission*
CB37	*Level 6/Series 200/2000 File Transmission*
CB38	*Level 6/BSC 2780/3780 File Transmission Facility User's Guide*
CB39	*Level 6/Level 64 (Emulator) File Transmission Facility User's Guide*
CB40	*IBM 2780/3780 Workstation Facility User's Guide*
CB41	*HASP Workstation Facility User's Guide*
CB42	*Level 66 Host Resident Facility User's Guide*
CB43	*Terminal Concentration Facility User's Guide*

In addition, the following documents provide general hardware information:

Order No.	*Manual Title*
AS22	*Honeywell Level 6 Minicomputer Handbook*
AT04	*Level 6 System and Peripherals Operation Manual*
AT97	*MLCP Programmer's Reference Manual*
FQ41	*Writable Control Store User's Guide*

Contents

Figures

Tables

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 (>). With a circumflex, this pathname is a full pathname, with a greater-than 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.

Section 2
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.

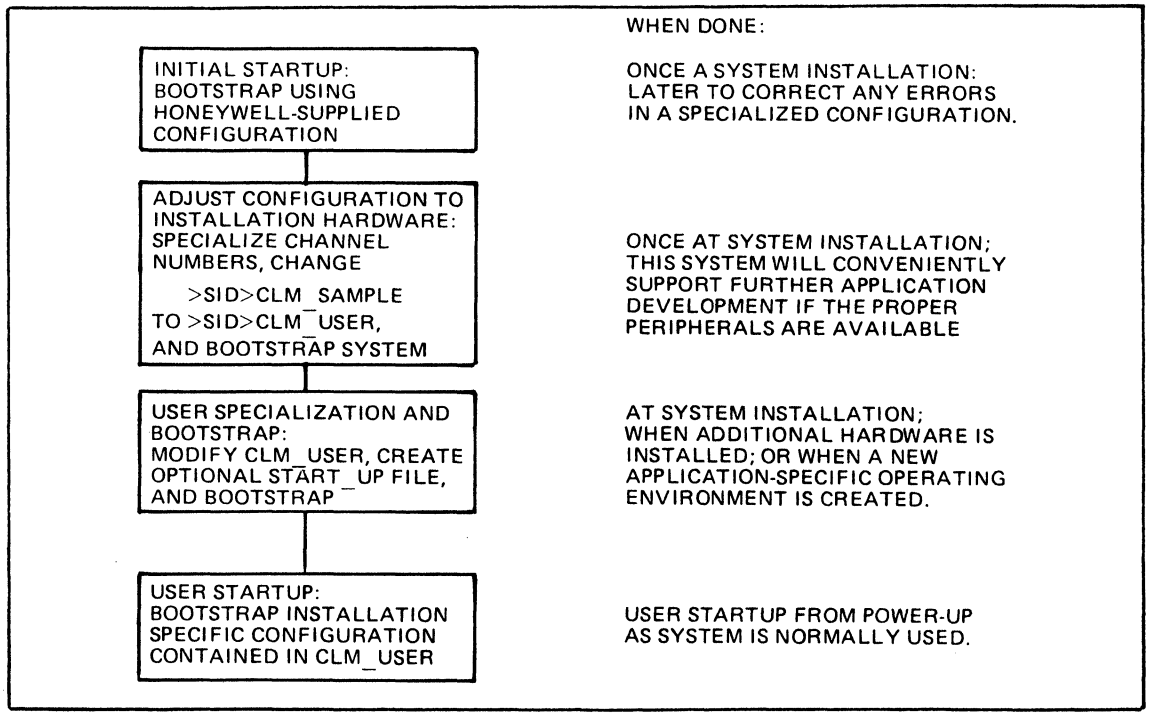


Figure 2-1. Stages of System Configuration and Startup

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:

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

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 task group $L. 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. These commands including their control arguments are described in Section 4.

```
CG $L base_lv1 -EFN LISTENER -POOL id
EGR $L user_id -OUT >SPD>CONSOLE -ARG  [ {'pathΔ'} ]  [x]  ["message"]
                                       [ {"pathΔ"} ]
                or
SG $L user_id base_lv1 -EFN LISTENER -POOL id -OUT >SPD>CONSOLE -ARG  [ {'pathΔ'} ]  [x]  ["message"]
                                                                     [ {"pathΔ"} ]
```

{'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.

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., ΔCΔ) and end with a carriage return.

Outstanding Output Message List

To obtain a list of outstanding output messages, enter:

ΔCΔ?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:

ΔCΔ: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:

ΔCΔPnnnC/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:

ΔCΔBid

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)

Message	Meaning
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.

DECREASING 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:

&QΔ or Δ$SΔ&QΔ

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.

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 [arg_1 . . . arg_n]; [command-name [arg_1 . . . arg_n]] . . .

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 can span two or more physical lines. A line is concatenated with the next line by ending it with an ampersand (&). A command line consisting of two or more concatenated physical lines can be cancelled by entering a single ampersand on the next physical line. Two or more commands may be entered in a command line by ending each command (except the last one) with a semicolon.

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
EXECUTION CONTROL COMMANDS	
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
FILE AND DIRECTORY CONTROL COMMANDS	
Change System Directory	CSD
Change Working Directory	CWD
File Out	FO
List Search Rules	LSR
List Working Directory	LWD
Modify File	MF
INTERACTIVE COMMANDS	
Enter Batch Request	EBR
Time	TIME
OPERATIONS COMMANDS	
Abort Batch Request	ABR
Abort Batch	ABORT_BATCH
Abort Group Request	AGR
Activate Batch	ACTB
Activate Group	ACTG
Create Batch	CB

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

Function Name	Command Name
OPERATIONS COMMANDS	
Delete Batch	DB
Load Sharable Bound Unit	LOAD
Reassign	RAS
Set Date	SD
Status System	STS
Suspend Batch	SSPB
Suspend Group	SSPG
Unload Sharable Bound Unit	UNLD
Writable Control Store Load	WCSLD
FILE AND DIRECTORY ACCESS COMMANDS	
Delete Access Control List	DA
Delete Common Access Control List	DCA
List Access Control List	LA
List Common Access Control List	LCA
Set Access Control List	SA
Set Common Access Control List	SCA

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.

DELETE ACCESS CONTROL LIST

Command Name: DA or DELETE_ACL

Remove entries from access control list (ACL) of a file or directory.

FORMAT:

```
{DA          }
{DELETE_ACL  } |path [user_id]| |ctl_arg|
```

ARGUMENT DESCRIPTION:

[path]

Specifies the pathname of a file or directory. If this argument is omitted or if -WD is entered, the working directory is specified. If it is omitted, user_id cannot be specified.

[user_id]

Specifies an access control name that must be of the form person.account.mode. All ACL entries with matching names are deleted. (For a description of the matching strategy, refer to the SET_ACL command.) If path is specified, user_id should also be specified. If user_id is omitted, the system id of operator.system.* is used.

[ctl_arg]

One or more control arguments from the following list.

```
{-A   }
{-ALL }
```

Causes all ACL entries to be deleted. This argument overrides user_id, if both are specified.

```
{-BF    }
{-BRIEF }
```

Suppresses the message "USER NAME NOT ON ACL"

FUNCTION DESCRIPTION:

This command removes entries from the access control list (ACL) of a file or directory. The user must have modify access to the containing directory in order to delete entries.

If the command is invoked with no arguments, it deletes the entry for the user's person.account.* on the ACL of the working directory.

DELETE COMMON ACCESS CONTROL LIST

Command Name: DCA or DELETE_CACL

Remove entries from common access control list (CACL) of a directory.

FORMAT:

{DCA / DELETE_CACL} [path[user_id]][ctl_arg]

ARGUMENT DESCRIPTION:

[path]
Specifies the pathname of a directory. If this argument is omitted or if _WD is entered, the working directory is specified. If it is omitted, user_id cannot be specified.

[user_id]
Specifies an access control name that must be of the form person.account.mode. All CACL entries with matching names are deleted. (For a description of the matching strategy, refer to the SET_ACL command.) If path is specified, user_id should also be specified. If user_id is omitted, the system id of operator.system.* is used.

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

{-A / -ALL}

Causes all CACL entries to be deleted. This argument overrides user_id, if both are specified.

-DIR
-DIRECTORY
Causes directory CACL entries to be deleted

-FILE
Causes file CACL entries to be deleted

{-BF / -BRIEF}

Suppresses the message "USER NAME NOT ON CACL"

FUNCTION DESCRIPTION:

This command removes entries from the common access control list (CACL) of a directory. The user must have modify access to the containing directory in order to delete entries.

If the command is invoked with no arguments, it deletes the entry for the user's person.account.* on the file CACL of the working directory. If -DIR and -FILE are both specified, both directory and file CACL entries are deleted. If neither is specified, only file CACL entries are deleted.

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 [ctl_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.

[ctl_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 intertretation 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 causes the file specified by path to be attached as the user_in file. If path is omitted, the current command_in file is assumed to be the user_in file.

&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.

&G label1

This directive, in conjunction with the &L directive provides a "go to" capability, and in conjunction with the IF-THEN-ELSE directive provides conditional execution of commands within the EC file. The next command to be processed is the command immediately after the first &L directive that defines the label.

&L label1

This directive defines a label that may be the object of a &G (or a conditional goto) statement. The label begins with the first non-blank (or tab) character after the &L and its length is restricted only by the inputn line length.

$$\&IF\Delta\left[EQUAL\Delta[RETCODE]\Delta hhhh\right]\ \Delta\&THEN\ \left[\left\{\begin{matrix}\Delta\&Q\\ \Delta\&G\ label1\end{matrix}\right\}\right]\ \left[\Delta\&ELSE\ \left[\left\{\begin{matrix}\Delta\&Q\\ \Delta\&G\ label2\end{matrix}\right\}\right]\right]$$

This directive causes the command processor to interrogate the status code (RETCODE) returned by the command executed immediately before the &IF directive. The subsequent processing depends on whether or not the status code enteree by the user matches that returned by the previous command.

[EQUALΔ[RETCODE]Δhhhh]

CAUTION

This argument including the double set of brackets must be entered when this directive is specified. This argument is an active function and the brackets are a part of it. Elsewhere, brackets denote optionality.

The hexadecimal number hhhh designates a status code. One to four digits may be entered. If less than four digits are entered, the field is left filled with zeros.

$$\&THEN\left\{\begin{matrix}\Delta\&Q\\ \Delta\&G\ label1\end{matrix}\right\}$$

If the status codes match, processing ceases or continues on the line following the label1 statement, depending on the option selected. If the options are omitted, processing continues on the next line.

$$\&ELSE\left\{\begin{matrix}\Delta\&Q\\ \Delta\&G\ label2\end{matrix}\right\}$$

If the status codes do not match, processing ceases or continues on the line folliwng the label2 statement, depending on the option selected. If the options are omitted, processing continues on the next line.

&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_GRPS.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 WD  VOL528>USER
EBR  DEV.PROGΔ>SPD>KSR01  -OUTΔ>SPD>LPT01Δ-W  >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 command 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 defined 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 operator output is 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 ACCESS CONTROL LIST

Command Name: LA or LIST_ACL

List entries on access control list of a file directory.

FORMAT:

{LA / LIST_ACL} [path[user_id]][ctl_arg]

ARGUMENT DESCRIPTION:

[path]

Specifies the pathname of a file or directory. If this argument is omitted or if -WD is entered, the working directory is specified. If it is omitted, user_id cannot be specified.

[user_id]

Specifies an access control name that must be of the form person.account.mode. All ACL entries with matching names are listed. (For a description of the matching strategy, refer to the SET_ACL command.) If user_id is omitted, the user's person.account.* is used.

[ctl_arg]

One or more control arguments from the following list.

-A
-ALL

Causes the entire ACL to be listed. If user_id is omitted, -A or -ALL need not be specified. If user_id is specified and -A or -ALL is also specified, the entire ACL is listed.

-BF
-BRIEF

Suppresses the message Δ"USER NAME NOT ON ACL"

Function Description:

The LIST_ACL command causes the entries on the access control list (ACL) of a file or directory to be listed. To use the command, the user must have list access to the directory containing the ACL.

If the command is invoked with no arguments, the entire ACL on the working directory is listed.

If user_id is specified (and -A or -ALL is not), the ACL entries that match the access control name are listed.

Example:

LA^ SYS01>UDD>PROJ2>FILEAA

The entire ACL contained on FILEAA is listed.

LIST COMMON ACCESS CONTROL LIST

Command Name: LCA or LIST_CACL

Lists all entries on requested Common Access Control List (CACL).

FORMAT:

LCA
LIST_CACL[path[user_id]][ctl_arg]

ARGUMENT DESCRIPTION:

[path]

Pathname of the directory containing the CACL entries. If this argument is -WD, or if this argument is omitted, the working directory is used. If the argument is omitted, user_id cannot be specified.

[user_id]

An access control name having the following format:

person.account.mode

All CACL entries with matching names are listed. If user_id is omitted, all CACL entries are listed.

[ctl_arg]

The following optional control arguments can be chosen:

{-A
-All}

Causes all entries on the requested CACL to be listed. If user_id is omitted, this argument is redundant; if both are specified, this argument overrides user_id.

{-DIR
-DIRECTORY}

Specifies that directory CCACL entries are to be listed. If -FILE (see below) is also specified, both file and directory CACL entries are listed. If neither DIR nor -FILE is specified, all CACL entries are listed.

-FILE

specifies that file CACL entries are to be listed. If -DIR is also specified, both file and directory CACL entries are listed. If neither -DIR nor -FILE is specified, all CACL entries are listed.

{-BF
-BRIEF}

Suppress the following message:

USER NAME NOT ON CACL

FUNCTION DESCRIPTION:

The LIST COMMON ACCESS CONTROL LIST command causes the entries on the common access control list (CACL) in the specified directory to be listed. Directory CACL entries, file CACL entries, or both, can be listed. The user must have list access to the directory referred to by path.

If this command is invoked with no arguments, all CACL entries on the working directory are listed.

If user_id is specified (and -A or -ALL is not), the CACL entries that match the access control name are listed. An explanation of the way in which names are matched is given in the description of the SET_ACL command.

Example:

LIST_CACL -WD JONES.INTFIN. -DIR

The directory CACL entries in the working directory that have JONES as the person, INTFIN as the account, and any value as the mode are listed.

LIST MOUNT REQUESTS

Command Name: LMR

Display all outstanding mount requests

FORMAT:

LMR

ARGUMENT DESCRIPTION:

No arguments are required or permitted with this command.

FUNCTION DESCRIPTION:

The List Mount Requests command lists (on user_out) the unsatisfied volume mount requests and, where applicable, the corresponding device names.

The entries in the list may be in either of two formats:

MOUNT VOLUME vvvvvv
MOUNT VOLUME vvvvvv ON DEVICE dddddd

The name of a volume in an entry may be blank, if a volume is to be formatted on a specific device.

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.

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_1 dev_name_2

The device controlled by the driver associated with dev_name_2 is to be interchanged with the device under the control of the device driver associated with dev_name_1. 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_1 and dev_name_2, 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 ACCESS CONTROL LIST

Command Name: SET_ACL or SA

Manipulate the access control list (ACL) of a file or directory by adding new entries or changing the access mode of existing entries.

FORMAT:

{SA / SET_ACL} path access_mode user_id [ctl_arg]

ARGUMENT DESCRIPTION:

path
Pathname of the file or directory containing the ACL. If this argument is -WD, the working directory is used.

access_mode
Specifies an access mode for the directories or files. (path specifies whether a directory or file is being operated upon.)

Any or all of the following values may be specified for files:

R — Read access
E — Execute access
W — Write access

Any or all of the following values may be specified for directories:

L — List access
M — Modify access
C — Create access

One of the following values may be specified for files or directories; if used, it must be the only entry:

N — Null access
Δ — Null access

user_id
An access control name having the following format:

person.account.mode

Existing ACL entries that have matching access control names receive the access mode specified by the access_mode argument.

If no matching entry is found, the entry is added to the ACL, provided that each component of the access control name resolves to a literal value.

If the user_id argument is not specified, the current user id is employed, with the following format:

person.account.*

[ctl_arg]
The following optional control arguments can be chosen:

{-DIR / -DIRECTORY}

Specifies that only directory values are to be allowed for the access_mode argument; if this argument is not specified, the first access_mode value specified sets the default.

-FILE
Specifies that only file values are to be allowed for the access_mode argument; if this argument is not specified, the first access_mode value sets the default.

FUNCTION DESCRIPTION:

Access control lists (ACL's) and common access control lists (CACL's) are an optional system feature that afford variable types of protection for mass storage directories and files.

ACL's can be set for any or all directories and files on mass storage volumes. An ACL for a *directory* indicates which users can create, modify, and/or list directories or files immediately subordinate to this directory. An ACL for a *file* indicates which users can read, write, and/or execute (if it is an executable entity) the file.

CACL's can be set at any or all directories on mass storage volumes. They are classified in two types: directory CACL's and file CACL's. A directory CACL indicates access rights for *all directories* immediately subordinate to the directory at which the directory CACL is set; a directory CACL is thus equivalent to an identical ACL being set for *each* of the immediately subordinate directories. A file CACL indicates access rights for *all files* immediately subordinate to the directory at which the file CACL is set; a file CACL is thus equivalent to an identical ACL being set for *each* of the immediately subordinate files.

The access rights defined by a directory CACL entry or by a file CACL entry can be overridden, on an individual basis, by ACL entries set on specific, immediately subordinate directories and files.

ACL's are set for directories and files by means of SET_ACL commands. ACL's are listed by LIST_ACL commands and are deleted by DELETE_ACL commands. CACL's are set at directories by SET_CACL commands. CACL's are listed by LIST_CACL commands and are deleted by DELETE_CACL commands.

It is important to realize that although ACL's and CACL's provide a *static* description of which users may gain access to specific directories and files and what type(s) of access they have, ACL's and CACL's do *not*, of themselves, *guarantee* a user access to a directory or file *at all times*. *Concurrency* constraints may *at some times* prevent a user from gaining access to a directory or file to which he is otherwise entitled to access by virtue of an ACL or CACL. (For instance, if user A has obtained access to a given file with *exclusive* concurrency control, other users cannot gain concurrent access to this file even though existing ACL or file CACL otherwise would permit them access to this file.)

Once established by a SET_ACL command, an ACL for a given directory or file consists of one or more entries, each of which contains a user_id and an access privilege. The same is true of a directory CACL and a file CACL established at a given directory by a SET_CACL command.

The user_id in an ACL or CACL entry consists of the three elements shown below:

person.account.mode

Note the correspondence between the elements of this user_id and the identity established for a user as he logs in to the system. The significance of this correspondence is that, in an environment that employs access control, a user's access to directories and files is directly related to his account.person.mode identity as established as he logs in to the system.

An important feature of the user_id in an ACL or CACL entry is that any or all elements of the user_id may be expressed by an asterisk. The asterisk is equivalent to "any" and thus permits varying degrees of generality or comprehensiveness for the access privilege specified in the same ACL or CACL entry. The following examples illustrate this point.

1. A user_id expressed as SMITH.ACCT1.* applies the accompanying access privilege to SMITH under ACCT1 in ANY mode.
2. A user_id expressed as *.ACCT2.* applies the accompanying access privilege to any user under ACCT2 in ANY mode.
3. A user_id expressed as HIRSCH.*.* applies the accompanying access privilege to HIRSCH under ANY account and in ANY mode.
4. A user_id expressed as *.*.* applies the accompanying access privilege to ANY user under ANY account and in ANY mode.
5. All other combinations are also legal.

As described later, a hierarchical scheme — relative to how "explicitly" a user_id is expressed — governs whether the access privilege specified in an ACL entry will be overridden by the access privilege specified in a related directory CACL or file CACL entry.

The access mode in an ACL entry for a file, and in a file CACL, can be set to one, two, or three of the elements shown below.[1] Permissible combinations are R, E, RE, RW, and RWE.

R — The user or users designated in this entry can *read* the file or files affected.

W — The user or users designated in this entry can *write* the file of files affected. (Write access to a file requires read access as well.)

E — The user or users designated in this entry can *execute* the file or files affected (provided the file is an executable entity).

The access privilege in an ACL entry for a directory, and in a directory CACL, can be set to one, two, or three of the elements shown below.[1] Permissible combinations are C, L, LM, LC, and LMC.

C — The user or users designated in this entry can *create* files and directories immediately subordinate to the directory or directories affected.

M — The user or users designated in this entry can *modify* files and directories immediately subordinate to the directory or directories affected. (Modify access to a directory requires list access as well.)

L — The user or users designated in this entry can *list* files and directories immediately subordinate to the directory or directories affected.

When checking access rights, the system first compares the user's login identity (person_id.account.mode) against the entries in the ACL (if any) of the target directory or file.

- If a direct match is found (possible only if all elements of an ACL entry's user_id are explicitly stated — i.e., no asterisks), the user's access privilege is established by that ACL entry.
- If a direct match is *not* found, the system searches the ACL entries to find the one of the highest priority that *includes* the user's login identity. The priorities are shown below in decreasing order of priority.

  highest 1. person.accound.mode
  2. person.account.*
  3. person.*.mode
  4. person.*.*
  5. *.account.mode
  6. *.account.*
  7. *.*mode
  lowest 8. *.*.*

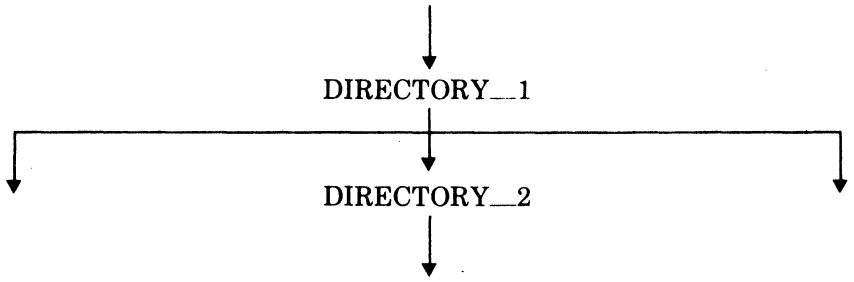
  When the system finds the highest priority ACL entry that *includes* the user's login identity (e.g., the user's login identity is ROTH.ACCT6.INT and the system finds an ACL entry whose user_id is *.ACCT6.* the system searches the related directory CACL or file CACL (if any) trying to find a CACL entry of *higher* priority that includes the user's login identity. If the system *does find* such an entry in the related directory CACL or file CACL, the user's access privilege to the target directory or file is as established therein. If no directory CACL or file CACL exists or if one exists but it does not contain an entry of higher priority that includes the user's login identity, the user's access privilege is as established in the target directory's or file's highest priority ACL entry that includes the user's login identity.
- If the target directory or file contains *no* ACL entry that includes the user's login identity, the directory CACL or file CACL (if any) is searched for the highest priority entry that includes the user's login identity. The user's access privilege to the target directory or file will be as established therein. (If the user's login identity is not included in *either* an ACL entry for the target directory or file *or* in an entry in a related directory CACL or file CACL, he has no access privilege in a protected access environment.)

---

[1] Alternatively, the access privilege can be set to N (no access). If N is specified, it must be the *only* access privilege element.

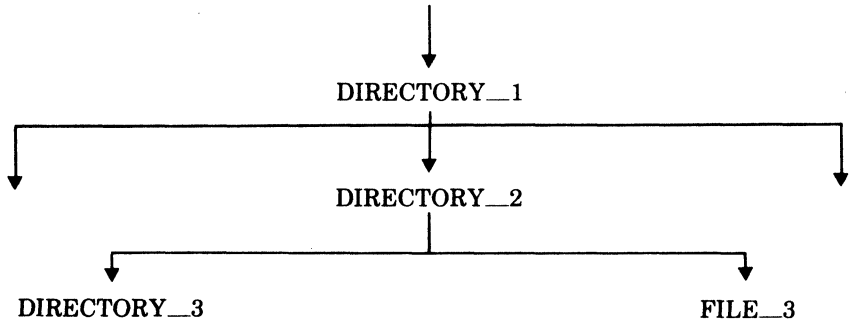
The following general guidelines and examples should prove helpful to your understanding of some of the basic characteristics of ACL's and CACL's.

For the purpose of the general guidelines first consider a directory structure as depicted below. Assume that this is a protected access environment.



To Perform This Action at DIRECTORY_2	User Requires This Type of Access Privilege
create an immediately subordinate directory or file	C (create) access in an ACL for DIRECTORY_2 or in a directory CACL at DIRECTORY_1
list any immediately subordinate directory or file	L (list) access in an ACL for DIRECTORY_2 or in a directory CACL at DIRECTORY_1
release an immediately subordinate directory or file	M (modify) access in an ACL for DIRECTORY_2 or in a directory CACL at DIRECTORY_1

Now assume that a DIRECTORY_3 and FILE_3 have been created immediately subordinate to DIRECTORY_2 as shown below.



To Perform This Action at DIRECTORY_3 or FILE_3	User[a] Requires This Type of Access Privilege
set-ACL	M (modify) access in an ACL for DIRECTORY_2 or in a directory CACL at DIRECTORY_1
list-ACL	L (list) access in an ACL for DIRECTORY_2 or in a directory CACL at DIRECTORY_1
delete-ACL	M (modify) access in an ACL for DIRECTORY_2 or in a directory CACL at DIRECTORY_1
set-common-ACL[b]	M (modify) access in an ACL for DIRECTORY_3 or in a directory CACL at DIRECTORY_2
List-common-ACL[b]	L (list) access in an ACL for DIRECTORY_3 or in a directory CACL at DIRECTORY_2
delete-common-ACL[b]	M (modify) access in an ACL for DIRECTORY_3 or in a directory CACL at DIRECTORY_2

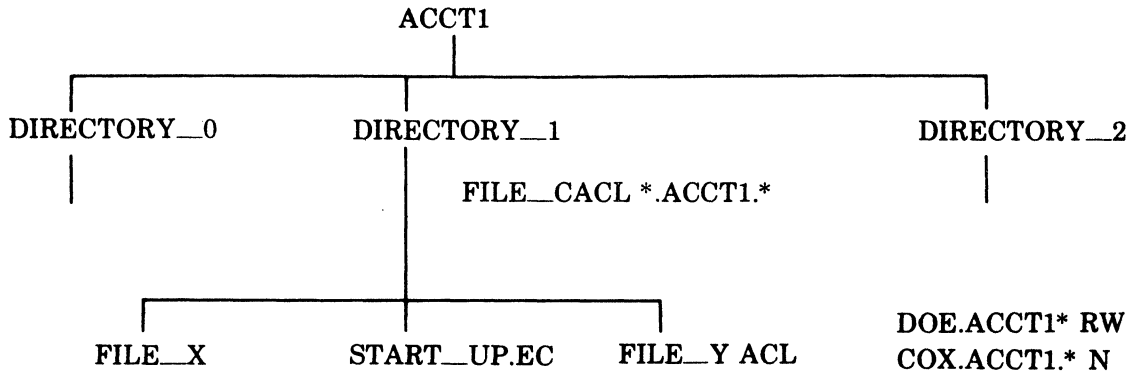
[a] This user need not necessarily be the one who created DIRECTORY_3 and/or FILE_3.

[b] Only at DIRECTORY_3.

The following examples illustrate a hypothetical case in which an account administrator has created several files and directories and then proceeds to set access privilege for account users. The examples illustrate a number of features of ACL's and CACL's but they are by no means comprehensive.

Any other approaches to setting access privilege might be used.

EXAMPLE 1:



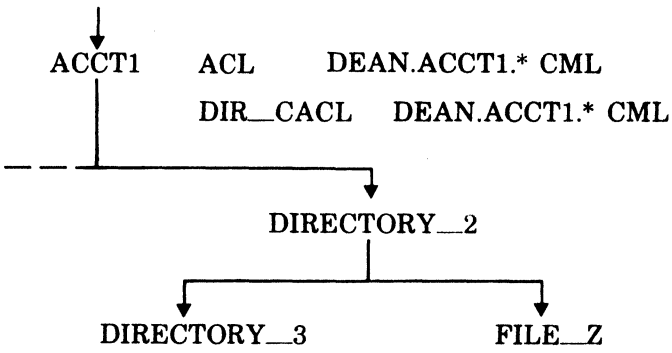
This example shows that the account administrator of ACCT1 has done the following:

1. He has created a file CACL entry at DIRECTORY_1 allowing all users of ACCT1 (in any mode) to read all files immediately subordinate to DIRECTORY_1. (Exceptions are noted below.)
2. He has set an ACL entry at FILE_Y allowing DOE to *read and write* the file. Thus DOE's access privilege to FILE_Y *exceeds* that of other users of ACCT1.
3. He has set an ACL entry at FILE_Y denying COX any access to the file. Thus COX's access privilege to FILE_Y is *less than* that of other users of ACCT1.

Both DOE and COX, like other users of ACCT1, have read access privilege to FILE_X and to START_UP.EC.

Note that although there is no ACL set directly at FILE_X or at START_UP.EC, no user except those in ACCT1 has access privilege to them because of the file CACL entry set at DIRECTORY_1.

EXAMPLE 2:



This example shows that although DEAN under ACCT1 (in any mode) is the only user who can create DIRECTORY_3 and FILE_Z (by virtue of the directory CACL entry at ACCT1), once they have been created, any user could create directories and files subordinate to DIRECTORY_3 and any user could set directory CACL or file CACL at DIRECTORY_3. These capabilities are available to all users because no directory CACL has been set at DIRECTORY_2. (This may be perfectly acceptable.)

If DEAN were to set a directory CACL entry for himself at DIRECTORY_2 *before* creating DIRECTORY_3 or FILE_Z, he could deny the above mentioned capabilities to all other users.

In any case, because of his access privilege at a higher level, DEAN could, by setting appropriate access privilege for himself, "undo" anything undesirable that a user had done at or below DIRECTORY_3.

SET COMMON ACCESS CONTROL LIST

Command Name: SET_CACL or SCA

- Manipulate the file or directory common access control list (CACL) of a directory by adding a new entry or changing the access mode of existing entries.

FORMAT:

{SCA / SET_CACL} path access_mode user_id [ctl-arg]

ARGUMENT DESCRIPTION:

path
Pathname of the directory containing the CACL. If this argument is -WD, the working directory is used.

access_mode
Specifies an access mode valid for directories or files (depending on whether a file or directory CACL is being added or changed).

Any or all of the following values may be specified for files:

R — Read access

E — Execute access

W — Write access

Any or all of the following values may be specified for directories:

L — List access

M — Modify access

C — Create access

One of the following values may be specified for files or directories; when the value is used, it must be the only entry:

N — Null access

Δ — Null access

user_id
An access control name having the following format:

person.account.mode

Existing CACL entries that have matching access control names receive the access mode specified by the <access_mode> argument.

If no matching entry is found, the entry is added to the CACL, provided that each component of the access control name resolves to a literal value.

If the user_id argument is not specified, the current user id is employed, with the following format:

person.account.*

ctl_arg
The following optional control arguments can be chosen:

{-DIR / -DIRECTORY}

Specifies that only directory values are to be allowed for the access_mode argument. If this argument is unspecified, the first access_mode value specified sets the default.

-FILE
Specifies that only file values are to be allowed for the access_mode argument. If this argument is unspecified, the first access_mode value specified sets the default.

FUNCTION DESCRIPTION:

The SET COMMON ACCESS CONTROL LIST command is used to modify the file or directory common access control list (CACL) of a directory by adding a new entry or changing the access mode of an existing entry. A file CACL contains the access control entries to be applied to all files described in the specified directory. A directory CACL contains the access control entries to be applied to all directories described in the specified directory. See "Function Description" for the Set Access Control List (SA) command for further information.

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.

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 $\left[[\text{path}_1]\ [\text{path}_2] \ldots [\text{path}_n]\right] \left[\begin{Bmatrix}\text{-DUMP}\\\text{-D}\end{Bmatrix}\ [(\text{xxx,yyy})]\right]$

$\left[\begin{Bmatrix}\text{-FILL}\\\text{-FL}\end{Bmatrix}\ (\text{xxxx,xxxx,xxxx,xxxx})\right]$ [-OFF] $\left[\begin{Bmatrix}\text{-CPU}\\\text{-CP}\end{Bmatrix}(\text{x})\right]$

ARGUMENT DESCRIPTIONS:

path ... path

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

$\begin{Bmatrix}\text{-DUMP}\\\text{-D}\end{Bmatrix}$ (xxx,yyy)

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_{16}, the highest is FFF_{16}.

-FILL
-FL $\quad\begin{Bmatrix}\text{-FILL}\\\text{-FL}\end{Bmatrix}$ (xxxx,xxxx,xxxx,xxxx)

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

Do not enable the WCS (i.e., leave it set to offline) after the WCS loader has completed all actions indicated by the other WCSLD arguments. The loader automatically disables the WCS at the beginning of its execution in the course of ascertaining the memory limits of the WCS. When loading is completed, the WCS is automatically enabled, unless -OFF is specified.

If no arguments are specified, the WCS is enabled. 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., $path_1$, $path_2$, $path_n$, 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 hardware.

Appendix B

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.

TERMINALS 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. These commands and their arguments are described in Section 4.

CG **$L** base_lvl -EFN LISTENER -POOL id

EBR **$L** user_id -OUT >SPD>CONSOLE -ARG [{'pathΔ' / "pathΔ"}] [x] ["message"]

SG **$L** 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.

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 have the option of accepting abbreviations for LOGIN commands requires 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 and again when the session terminates, listener performs specific operations affecting the state of a terminal. 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:

Δmsg_noΔmessage-of-the-day

CUT ALONG LINE

HONEYWELL INFORMATION SYSTEMS

Technical Publications Remarks Form

TITLE: SERIES 60 (LEVEL 6)
GCOS 6 MOD 400
OPERATOR'S GUIDE ADDENDUM A

ORDER NO. CB24A, REV. 0

DATED JUNE 1978

ERRORS IN PUBLICATION

SUGGESTIONS FOR IMPROVEMENT TO PUBLICATION

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

FROM: NAME _____ DATE _____

TITLE _____

COMPANY _____

ADDRESS _____

_____

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

FIRST CLASS
PERMIT NO. 39531
WALTHAM, MA
02154

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

Postage Will Be Paid By:

HONEYWELL INFORMATION SYSTEMS
200 SMITH STREET
WALTHAM, MA 02154

ATTENTION: PUBLICATIONS, MS 486

CUT ALONG LINE

FOLD ALONG LINE

FOLD ALONG LINE

Honeywell