MAX-PLANCK-INSTITUT FÜR PLASMAPHYSIK

GARCHING BEI MÜNCHEN

GALE Terminal Users Guide

Robert Lathe
Klaus Engelhardt
Klaus Kottmann
Erich Müller

IPP R/23 October 1977

Die nachstehende Arbeit wurde im Rahmen des Vertrages zwischen dem Max-Planck-Institut für Plasmaphysik und der Europäischen Atomgemeinschaft über die Zusammenarbeit auf dem Gebiete der Plasmaphysik durchgeführt.

GALE Terminal Users Guide

R. Lathe
K. Engelhardt
K. Kottmann
E. Mueller

Abstract

This manual is designed to be used by general GALE terminal users. As such, it contains information describing the usage of the four main tasks which at present comprise the GALE system. Furthermore, certain RSX-11M operator commands which have been modified or are necessary in order to activate the GALE system are described.

Table of Contents

Preface

0.1 Manual Objectives

This manual is designed to be used by general GALE terminal users. As such, it contains information describing the usage of the four main tasks which at present comprise the GALE system. Furthermore, certain RSX-11M operator commands which have been modified or are necessary in order to activate the GALE system are described.

More information concerning RSX-11M commands is available in the RSX-11M Operator's Procedures Manual. Standard system utility programs for creating and editing files are described in the RSX-11M Utilities Procedures Manual. Information concerning programming may be found in the appropriate RSX-11M language manuals.

0.2 Structure of the Manual

This manual is organized into chapters. The first three chapters provide information of a general nature and describe in detail the conventions applicable to terminals and files. Chapter 4 describes the usage of operator commands. The remaining four chapters give a detailed description of the operating procedures for the major GALE tasks. Finally, the appendix gives a summary of error messages given by the GALE system.

CHAPTER 1

Introduction

GALE (General Acquisition system for Laboratory Experiments) is a functional and logical extension of the RSX-11M operating system. As such, it supports multi-tasking real-time operations. Its fundamental function is to provide control for the sharing and utilization of hardware (and software) resources among several system and user supplied tasks *.

Tasks stored on a file-structured volume may be installed into the system and subsequently run by issuing appropriate commands to the Monitor Console Routine (MCR). The terms "install" and "run" have definite meanings which are discussed below. MCR provides the interface between the user and the system.

Tasks compete for system resources depending upon their priority and upon resource availability. The highest priority task which has all the resources required to run, and which can make use of the resources it needs, will be in control of the CPU. As tasks request system services, they become blocked, usually waiting for an I/O transfer to complete. During this blockage, the system looks for another task to run. Running a task in the blocked intervals of other tasks is termed multi-tasking.

In addition to competition between memory resident tasks, another level of competition is introduced by the checkpointing facility. A task currently running may, if declared checkpointable or if waiting for terminal input, be pre-empted from memory, moved to mass storage, and replaced by a higher priority task installed to run. Later, when the

* A more complete description of the GALE philosophy and general components is provided in the manual "Introduction to the GALE System".

higher priority task has finished, the previously swapped-out task is returned to memory and continues operation from the point of interruption. This roll-in, roll-out process is termed checkpointing.

It was stated above, that the terms:

install, and

run

have definte meanings. These meanings will be defined while sketching the creation of a task. Assume that an operational system exists.

A task comes into existence through four basic steps:

1. A program is written in a supported source language (MACRO-11 or FORTRAN) and stored on a file-structured volume via a source language processor (e.g., EDI).

2. The source code is submitted to the applicable translator and an object file is built on a file-structured volume.

3. The object file is submitted to the Task Builder, and the output, known as the task-image file, is also stored on a file-structured volume.

4. Finally, via an MCR command, the task is installed into the operational system.

Installation involves recording a number of task parameters in a system-resident table called the system task directory (STD). The recorded task parameters include its name, length, and volume address. An installed task is one which has an entry in the STD. It is simply known to the system and is considered dormant (not running) until activated.

A dormant task may be activated (set running) via the MCR Run command, or it may be activated internally by another active task which issues a request to the system. When such a request is received, a sequence of events is initiated that allocates resources, brings the task into memory, and places it in active competition with other tasks. Thus, an installed task is one that is known to the system, which is dormant, since it is not eligible to compete for resources. On the other hand, an active task is also known to the system, but it is eligible to compete for system resources.

The concept of dormant versus active is much more important in a realtime system than in a batch-oriented system. A dormant task uses very little memory; yet, when the task is needed to service a realtime event, it can quickly and efficiently be introduced into active competition for system resources, since most of the parameters describing the task are already resident in main memory.

CHAPTER 2

Procedures and Conventions

2.1 Terminal Operations

Effective terminal interaction requires complete understanding of the following:

Terminal operations

Peripheral device naming and assignment

File naming and file specifiers.

This and the following chapter of this manual provide complete information for normal users on these points. Information on advanced techniques useful for system programmers and system operators may be found in the RSX-11M Operator's Procedures Manual. *

2.1.1 Special and Control Characters

Special characters are used to control an operator terminal. Control characters are typed while the control key (CTRL) and one other key are simultaneously depressed. The two control characters CTRL/Z and CTRL/U are echoed as ^Z and ^U. Other CTRL characters, although echoed, are non-printing characters and, therefore, do not appear on the display medium. The characters used in terminal operations are given below.

* Portions of this manual are excerpted from the RSX-11M Operator's Procedures Manual.

CTRL/C Typing CTRL/C causes the MCR recognition sequence disucssed below to be initiated. ^C is not echoed.

CTRL/I Horizontal tab causes spacing to the next tab stop. These stops are set by the software following every eighth character position. ^I is not echoed.

CTRL/K Typing CTRL/K causes a vertical tab (appears on the terminal as four line feeds). ^K is not echoed.

CTRL/L Typing CTRL/L causes a form feed. Paging is not done; eight line feeds are placed on the display medium. ^L is not echoed.

CTRL/O CTRL/O is used to suppress the display of unwanted output on a terminal. The subsequent effect of entering CTRL/O depends upon the state of the terminal when the character is entered. ^O is not echoed.

CTRL/O is controlled internally by a single bit called the Disable-Output bit. When this bit is set, output to the terminal is disabled; when cleared, output can proceed.

Initially the bit is cleared. When CTRL/O is entered, the bit is complemented. Thus, a typical response of entering successive CTRL/O characters is to stop and start output to the terminal. Characters that would have normally been displayed during the stopped interval are discarded.

If more than one task is sending output to a terminal, it is essential that CTRL/O affect only the task delivering output at the time CTRL/O is entered; control is implemented as follows:

The disable bit is cleared for attached terminals at:

The issuance of an Attach QIO directive;

The issuance of a Detach QIO directive;

The solicitation of input, and

The arrival of unsolicited input.

For unattached terminals, the bit is also cleared each time an output request directed to the terminal is initiated.

Given these conditions for clearing the disable bit, and the fact that entering CTRL/O always complements the bit, the following applies at the entering terminal:

For an attached terminal, output can be stopped by entering CTRL/O. The interrupted output stream will be discarded until the next:

Detach;

Solicited input;

Unsolicited input, or

Another CTRL/O.

For unattached terminals, entering CTRL/O stops output, but remains in effect for only the current buffer or output, since, at the next I/O initiation, the system always clears the Disable-Output bit for unattached terminals.

CTRL/Q Typing CTRL/Q after typing CTRL/S will resume output suspended by the previous CTRL/S. ^Q is not echoed.

CTRL/R Typing CTRL/R before typing a line terminator causes the system to retype the current line on a new line, omitting any deleted characters. If the current line is empty, CTRL/R performs a carriage return and line feed.

CTRL/S Typing CTRL/S during output suspends additional output until CTRL/Q is typed. The functions of CTRL/S and CTRL/Q are convenient when using a display terminal which transmits at rates above 600 baud. ^S is not echoed.

CTRL/U Typing CTRL/U prior to typing a line terminator causes the previously typed characters to be deleted back to the default prompting symbol ">". The system responds with a carriage return and line-feed so that the line can be retyped. Echoed as ^U.

CTRL/Z A break character indicating end-of-file. It is used as a signal to system tasks, such as PIP and DLG, indicating that the user is finished and the subject task may exit. Echoed as ^Z.

ESC The ESC character represents the ESCape or ALTMODE key. Entering this line terminator leaves the carriage (or cursor) of the terminal intact. It has a specialized meaning to MCR when terminating a command. It will suppress the default prompt from MCR commands when they complete. However, an ESCape character terminating a Run command produces special results (see the description of the Run command for a detailed explanation). ESC is not echoed.

CR — Typing the RETURN key will end line input and cause the carriage (or cursor) to return to carriage position 1.

\ — The back slash symbol is used to delimit characters deleted by typing the RUBOUT or DELETE key. Typing a RUBOUT deletes the last character typed. Several contiguous characters can be deleted by typing successive RUBOUT's.

PAGE — The Page or RESETPAGE key is available only on page oriented terminals (e.g., TEK4012). Depressing this key starts a sequence in the terminal which is not carried out under system control. The screen is erased and the cursor is set to the upper left corner of the screen. This process requires approximately 750 milliseconds, during which time the terminal input and output is disabled. Each time the terminal page is filled, indicated by a red light on the keyboard, the PAGE key must be depressed before terminal operations can continue.

2.1.2 Terminal Characteristics

There are two types of MCR commands: privileged and non-privileged. Non-privileged commands may be invoked at any terminal, whereas privileged commands may be invoked only at privileged terminals. The privileged characteristic is associated with a given terminal via the sign-on (HELlo) procedure and is thus directly associated with the User Identification Code (UIC) of the terminal user.

Caution

Privileged commands can destructively interfere with system operation and with themselves. Caution is advised when more than one privileged terminal is present on the system.

Non-privileged commmands are availble to any unattached * terminal in the system. A terminal may be attached to a task other than MCR, in which case, all input or output is directed to or from that task. This capability is restricted only by the memory available for the various tasks.

* Attached terminals are those dedicated to a single task. Only the task to which it is dedicated can perform I/O to such a terminal.

Commands are serviced by either:

1. Independent tasks (such as DLG), or

2. The MCR Dispatcher and a set of overlays.

Whenever a line is entered from an unattached terminal, the MCR Dispatcher reads the commmand line. It then takes the first three characters of the line (these are unique and form a part of the command) and searches a table of commands processed by overlays. If the MCR Dispatcher finds the command in this table, it loads the overlay to process the command. If it does not find the name, it prefixes

... (three periods)

to the command and serches the system task directory (STD) for a task by that name (e.g., ...DLG), as described in the following paragraphs.

If the MCR Dispatcher finds the task and it is not active, the Dispatcher will request the Executive to run the task. If the task is active or not in the system, the Dispatcher will search for the task, as follows:

<command-name>Tn

where n is the terminal unit number from which the command was issued (e.g., DLGT2). If the Dispatcher finds the task and it is not already active, the Dispatcher will request that the task be run. If the Dispatcher does not find task <command-name>Tn, but task ...<command-name> is installed in a system-controlled partition and is not checkpointable, the Dispatcher will create a task by the name of <command-name>Tn and request it to be run. This task will be identical (uses the same task-image file) to the parent task ...<command-name>. This feature is useful in multi-terminal systems.

At the time the Dispatcher makes the request, it also sends the complete command to the task it just requested. This scheme makes it possible for users to add (or delete) services in the requested task.

For example, an installation could write a routine which displays the day of the year using the internally stored current date. If the task is called DAY, the user would build and install it as:

...DAY

Then the user-issued command:

MCR>DAY

will result in task initiation.

2.2 Terminal Input Conventions

An unattached terminal waiting for input is in one of three states:

1. Ready to accept unsolicited input;
2. Ready to accept solicited input to MCR; or
3. Ready to accept solicited input to a task other than MCR.

A terminal ready to accept unsolicited input displays the character:

```
>
```

on the terminal. This single-character display on a terminal is referred to as the default prompt. Input entered following a default prompt is always implicitly directed to MCR, and the MCR Dispatcher identifies, analyzes, and responds to subsequent input.

Whenever CTRL/C is entered as the first character on a terminal awaiting input, MCR is explicitly initiated, and the prompt:

```
MCR>
```

appears on the display medium. An input entered following the

```
MCR>
```

prompt is explicitly directed to MCR. In the case of the default prompt, however, another task could internally solicit input; if the soliciting task did not itself issue a prompt, then input (though thought to be directed to MCR) would in fact be sent to the task most recently soliciting input. The default prompt merely indicates that MCR will field unsolicited input.

To avoid erroneous input, any task intending to solicit input should:

1. Attach to the terminal;
2. Prompt with a properly formatted identifier (i.e., tsk>), and then
3. Solicit input.

Once a task is initiated by MCR, it will generally prompt with:

```
tsk>
```

where tsk is a 3-character task name. All system tasks are identified with a 3-character prompt; it is advisable that user programs follow this convention. When a task prompt has been issued and immediately followed by a read, the next line of input is directed to the soliciting task.

When a task is sending characters to a terminal without attaching to it, the output can be interrupted by entering any character. At the completion of the current I/O operation, the character will be echoed, and the operator can issue a command to MCR.

For attached terminals sending data, a similar interruption can be achieved by entering CTRL/C. Then, as with unattached terminals, at the completion of the current I/O operation:

```
MCR>
```

will be displayed. These conventions enable tasks delivering output to be interrupted conveniently.

MCR commands are usually initiated as unsolicited input and are terminated by typing a carriage return; the OPEn command is an exception, because it is terminated with an ESCape key. If any other command to MCR is terminated with an ESCape character, MCR will not respond with the default prompt. Typing CTRL/C as the first character in a line, or anytime when a terminal is sending data, will always obtain MCR's attention, even for attached terminals. Unsolicited input, except CTRL/C, will be rejected if the terminal is attached.

If a task prompt is awaiting input, the sequence:

```
CTRL/C <CR>
```

will obtain MCR's attention. Following a single input line to MCR, the original task whose prompt was overridden is again given control of the terminal.

2.3 Error Detection and Handling

Error detection and reporting are provided for the various MCR commands. When an error is detected, an appropriate message (prefixed by the name of the command) is displayed at the entering terminal. Messages which are unique to a given command are listed with the discussion of the command. All command error messages are listed in Appendix A. Error messages which result from incorrect I/O operations are displayed without the corresponding task name prefix.

2.4 Command Strings

When typing MCR commmand strings, the following conventions apply:

1. Command strings are terminated either by a carriage return (<CR>) or ESCape (<ESC>) key depression. If the command is terminated by a <CR>, a default prompt will be displayed on completion of the command. If the command is terminated by an <ESC>, the default prompt will not be displayed when the command completes. Default prompt suppression is used in conjunction with a special feature to the Run command. Thus if the following sequence is entered:

   ```
   >RUN TEST<ESC>
   ```

   no default prompt will be displayed when the Run command completes the request to initiate the task TEST. Rather, as a feature of the Run command terminated by an ESCape action, the default prompt will be displayed when the task TEST exits. This feature enables the operator to distinguish when a task, which produces no output on the terminal, has completed.

2. A command must be separated from its parameters by at least one space or one tab.

3. If an error is discovered while entering a command string prior to typing a terminator, the line may be deleted back to the prompting character (>) by typing CTRL/U (formed by simultaneously pressing the CTRL and U characters). RUBOUT, which is echoed as a backslash (\), may be used to delete the last character typed. The first RUBOUT typed will cause \ and the character deleted to be echoed. Further RUBOUTs cause only the deleted character to be echoed. The next non-RUBOUT char-

acter entered will cause a \ to be echoed, along with the character entered. Subsequent non-RUBOUT characters (with no intervening RUBOUTs) are echoed. Thus:

MISTKAE\EAK\AKE

produces:

MISTAKE

4. Any number of alphanumeric characters may be concatenated to the first three letters of the command and before its arguments or command string terminator (<CR> or <ESC>). One or more spaces or tabs must separate the command name from its operands. Spaces or tabs are useful in improving the readability of printed copy. For example, the Mount command could be invoked to mount the pseudo-device MO: by typing:

MOU MO:<CR>

or

MOUNT MO:<CR>

2.5 Initiating Tasks

There are four methods for intiating tasks. These methods are described below. The first two methods apply when the task is installed and ready to be executed. The remaing two cause the task to be installed, executed, and then removed on exit.

2.5.1 Initiation of Installed Tasks

Method 1. >utilityname command-string <CR>

Loads, executes the specified command(s) and exits.

Method 2. >utilityname <CR>

Responds with the following prompt:

utilityname>

At this point, the user enters the utility command string to execute the desired function. When the utility has completed processing a command string, it again issues a prompt. The user can either enter another command string or enter a CTRL/Z to terminate the utility.

2.5.2 Initiation of Uninstalled Tasks

Method 3. >RUN $taskname <CR>

Causes the task to be installed and loaded, and to issue the following prompt:

taskname>

At this point, the user enters the task command string to execute the desired function. When the task has completed processing a command string, it again issues a prompt. The user either enters another command string to execute the desired function or enters CTRL/Z to cause the task to exit. After exiting, the utility is removed from the system.

Method 4. >RUN $taskname/UIC=[g,m] <CR>

The UIC under which the utility executes is explicitly specified for this run only. Normally, tasks execute with the default UIC associated with the initiating terminal. When the task is installed and loaded, it issues the following prompt:

taskname>

At this point, the user enters the task command string to execute the desired function. When the task has completed processing a command string, it again issues a prompt. The user either enters another command string to execute the desired function or enters CTRL/Z to cause the task to exit. After exiting, the task is removed from the system task directory.

2.6 Peripheral Device Naming and Assignment

During system generation, all peripheral devices attached to the system are explicitly described. To simplify the naming of these devices, each device is given a unique name which is used in all commands referring to the device.

All peripheral devices are referenced by using a 2-character ASCII device name and an optional 1- or 2-digit octal unit number (for example, DK0 and TT2). The combination of device name and unit identifier is referred to as the device-unit pair or simply the device-unit. If the unit number is omitted, the system uses unit 0 by default; thus, LP: indicates the line printer 0.

Programs communicate with peripheral devices through logical unit numbers (LUNs). By relating a LUN to a device-unit, input/output issued to that LUN will be directed to the device represented by the device-unit designation. LUNs are associated with device-units at one of three stages in the task creation process:

1. During task build;
2. Via an MCR REAssign command after the task has been installed, and
3. At task run time via the Assign LUN Executive directive.

2.6.1 Device Assignment

The device assignment process effects a name connection. It relates a program name (LUN) to a physical device-unit; nothing more is implied. If the device-unit has been generated into the system, the name connection will succeed. Success, however, does not necessarily mean the device can immediately be accessed. The device may, for example, be attached to another task, or it may be physically inoperable; neither of these conditions will prevent a valid assignment from occurring. Successfull access can subsequently be made when the device-unit becomes unattached or the malfunction is corrected.

2.6.2 Device Redirection

The operator, in practice, does not often reassign devices. A much more common operator function is that of device redirection. Redirection causes I/O directed to one device-unit to bypass the original name connection and instead be directed to a different physical device-unit. Physical device redirection is a system wide phenomenon. Thus, any task referring to a redirected device will have its I/O redirected to the target device.

Suppose, for example, that a task has LUN 1 assigned to TT:. Assume during the task's execution that TT: becomes inoperable, making it impossible for the task to continue. The operator can redirect I/O destined for TT: to TT1. This redirection remains in effect until another Redirect command changes it. Redirection can occur to any level. Thus, if TT1: in the above example becomes inoperable, I/O can be redirected to TT5: or any other appropriate device.

Note that redirection does not in any way alter the original LUN-to-physical-device-unit assignment. If the task is removed from memory and then subsequently re-activated, and the internal redirection is still in effect, the redirect path will automatically hold for the task.

2.6.3 Pseudo Devices

The redirect process usually involves seven pseudo devices:

CO:	Used for console output
CL:	Used for console listing
DA:	Data Acquisition pseudo-device
MO:	Message Output pseudo-device
NT:	Used for network communications
TI:	Used for terminal input
SY:	Used as the system device

Virtually every task in the system, including system tasks, has a need to communicate with one or more of these pseudo devices. Unlike the real devices in the system, pseudo devices are so named because they do not correspond to real devices until they are redirected.

In a given task, LUN 1 may be assigned to CL:. This task, in delivering output to LUN 1 wants the data sent to console listing device; the task itself is not concerned with the particular device in the system actually serving as the console listing device.

The operator can change the device to be used as the console listing device by redirecting CL: to another device-unit (e.g., from a line printer to a terminal). The effect of this redirection is to cause all I/O directed to CL: to go to the terminal rather than to the line printer. Note that the tasks directing I/O to CL: are unaware of this change and are not altered in any way.

When a pseudo device-to-real device redirect connection is made, the pseudo device assumes the privileged or non-privileged attribute of the real device. Thus if CO: is redirected to TT0: (which is privileged), and then subsequently redirected to TT1: (which is not privileged), CO: loses its privileged status.

Normally, CO: is redirected to the main operator's terminal, and CL: is normally redirected to the line printer. The pseudo device TI: (terminal input/output) is the pseudo device most frequently used. When a task is initiated via an MCR Run command, the terminal from which it is started is established as the task's TI: terminal. The initiated task can communicate with the user via the TI: pseudo device. If a task is initiated internally via a Run Executive directive, TI: is defaulted to CO:.

The MO: pseudo device is used for system output to the user terminal and to the CL: pseudo device. Formatted messages may be specified from user or system tasks for output to either or both units, where the terminal device TI: is always that from which the task was started. Thus MO: essentially interprets a message request and redirects the output to the specified device(s). MO: may not be redirected via the REDirect command.

The DA: pseudo device provides the means for implementing the dynamic I/O System. DA: accepts requests to load and unload device handlers and passes requests from user or system tasks on to these handlers. The DA: pseudo device may not be redirected via MCR.

2.6.4 Logical Devices

Logical device names may be assigned to physical devices by means of the MCR ASsigN command. A logical device name consits of a 2-character ASCII name and an optional 1- or 2-digit octal unit number. Any reference to a logical device will be mapped by the system into the physical device assignments: local and global. Local logical device assignments are in effect only for tasks intiated from the entering terminal. That is, local logical device assignments are associated with a particular terminal. Global logical device assignments affect all tasks running in the system.

The logical device table is always searched by the system before the physical device table. This precedence allows logical device assignments to override physical device names. Local device assignments override any global assignments, and global assignments override physical device names. Similarly, identical local logical device names may be used at different terminals without interference.

This feature, which is a system generation option, allows a programmer to write a program without concern for the device actually used at run time. He may reference a logical device name and assign it to the desired device prior to run time.

In the GALE system, two logical devices, DS: and DD:, deserve particular attention. The DS: logical device name must be assigned to the physical device which contains the system configuration file DASCNF. This file contains a description of the devices to be used in acquiring data. The logical device DD: is used to designate the physical device onto which the files containing the data taken are to be written. In systems supporting only one experiment, these assignments may be made globally; in multi-experiment systems, the assignments must be made locally.

2.7 Multiuser Protection Functions

Multiuser protection, a system generation option, allows an RSX-11M installation to monitor and control individual users of the system. In particular, the system is protected against undesired users, and user files are protected against inadvertant changes by other users. Users of smaller systems which do not support multiuser protection may ignore the following comments.

2.7.1 Logging On and Off a Terminal

Each user must log onto a terminal before the system allows the user to issue further MCR or task commands, other than the MCR command HELP. Depending on the user's UIC, the terminal logged onto becomes privileged or nonprivileged. The process of logging on and off a terminal enables the system to keep accounting information about each user. The MCR commands that log a user on and off a terminal are HELlo and BYE.

2.7.2 The HELlo Command

The parameters to the HELlo Command are the user's UIC (or last name) and a password. The standard format for a UIC is [g,m], where g and m are octal numbers from 1 to 377 that represent the user's group and member numbers, respectively. The HELlo command allows several UIC formats; The UIC format specified affects the system's response to the command. A privileged user has a group number less than or equal to 10. Each UIC has an associated 1- to 6-character alphanumeric string password that guards against unauthorized access to the system. No one can successfully log onto a terminal without supplying the correct password for the specified UIC (or last name).

For example:

```
HELLO <CR>
ACCOUNT OR NAME:  [100,100] <CR>
PASSWORD:  TEST
```

Note that the HELlo command prompts for its parameters when the user presses <CR> after each element of the command string. Alternatively, a user can type the entire command string on one line, as follows:

```
HELLO [100,100] /TEST <CR>
```

where TEST (preceded by a mandatory slash) is the password for the specified UIC. (The slash is required only when the password is on the same line as the UIC or last name). When the user types a password in response to the prompt PASSWORD:, the system does not display the typed characters.

The user can also type the command in a mixed format, supplying one element of the string on one line and prompting for the other. The following example ensures that the system suppresses the user's password:

```
HEL PEEK <CR>
PASSWORD:
```

Note that this example of the HELlo command specifies the user's last name, from which the system determines the user's UIC.

After a user has successfully logged onto a terminal, the system automatically assigns the logical name SY: to the user's system disk (a login logical device assignment). If the user's group number is less than 11, the terminal becomes privileged; if the group number is 11 or greater, the terminal assumes nonprivileged status.

2.7.3 The BYE Command

A user who has completed a session at the terminal issues the BYE command to log off. This command has no parameters; it simply instructs the system to terminate the session. The system deletes all local and login logical device assignments, dismounts all mounted private devices, aborts all active nonprivileged tasks requested from the terminal, displays a termination message, and prevents further use of the terminal until another user logs on.

Example:

```
BYE
HAVE A GOOD AFTERNOON
12-SEP-77 2:45 TT6:  LOGGED OFF
```

Each user must log off the terminal after completing a session.

CHAPTER 3

Files and File Specifiers

3.1 File Fundamentals

Files are owner-named areas on direct access volumes. Volumes are data-carrying media which can be file structured. This structure and the software which supports it is called Files-11.

Files behave much like devices in that task I/O may be directed to them. However, since a file is just a portion of a random access volume, files can be created as long as space for them exists on the volume. Thus, they are much more dynamic and flexible than physical devices. Due to the fact that more than one programmer's files may exist on the same volume, conventions are required for naming the areas and techniques for protecting files belonging to different owner's from being altered, except by the owner or another user with the owner's permission.

Filenames are made unique by use of a user identification code (UIC). A file cannot be accessed unless the UIC under which it is stored is known. Knowing the UIC does not guarantee access, however, since each file existing under a given UIC can be individually protected. Individual file protection is discussed below.

Associated with each UIC is a user file directory (UFD). The UFD is a file which contains the names of all files currently existing under a given UIC. A UFD is itself a file belonging to a specific user and has all the characteristics of a file, including the manner in which access to it is granted.

UIC's are specified in the format [nnn.nnn]; nnn is a 3-digit number, usually represented in octal, having a range of 0 through 377(8). The first number in the pairs is referred to as the group number; the second, the member

number. To eliminate repetitive entry of UIC's in command strings, a default UIC can be established for each terminal in the system, either at system generation or dynamically from the terminal with the HELlo command.

3.2 File Protection

Four types of file actions exist:

Read;

Write;

Extend, and

Delete.

The users that are allowed to perform these actions are divided into four categories:

System -- All system software (all tasks with a group number of 10(8) or less and individual UIC's declared as privileged during the system generation).

Owner -- The user under whose UIC the file is stored,

Group -- Any user whose UIC has the first three digits in common with the owner,

World -- Any user not included in the above three categories.

When protection is specified, any combination of the four actions (read, write, extend, delete) can be assigned to each of the four groups. For example, the system and the owner can have read, write, extend, and delete privileges, the group can have read privileges, and the world can be denied access to the file.

The basic protectable entity is the file. The UIC is required to obtain the opportunity for accessing a given set of files, but each file stored under a given UIC is individually protected. Access to a specific file, given the UIC, is established by the owner. The owner can prevent access to each of his files, and he can extend different access rights to the four classes of users described above.

The owner of a file establishes access privileges to these four user classes in one of five ways:

1. At volume initialization time. The INItvolume command allows the user to specify default file protection for all files created on the volume.

2. The MCR User File Directory command. This command establishes access rights to the UFD file only. Often, a user will permit access to his UFD, but will restrict access to the files catalogued in the UFD.

3. PIP File Utility. PIP has an option which enables the owner to alter access privileges of his files.

4. With a user task. Using the WRITE ATTRIBUTES Files-11 I/O function, a user can write new access protecion attributes to files he owns.

5. At mount time. In the MOUnt command, the user may specify the default file protection to be given files created after the mount operation; this specification overrides that specified during volume initialization.

3.3 File Specifiers

Any task which has a requirement to refer to files does so via a standard command string with the following general format:

outfilespc1,...outfilespcn=infilespc1,...infilespcn

Outfilespc is an output file specifier, and infilespc is an input file specifier.

Any number of specifiers are possible, the actual number being determined by the task which will use the command string. In no case, however, may the total length of the command string exceed the maximum line length (80 bytes).

Each file specifier (whether input or output) has the following format:

dev:[g,m]filename.type;version/switch1.../switchn

where:

dev: The physical device-unit on which the volume, containing the desired file, is mount-

ed, for example, DK0:. The designation consists of a 2-character ASCII name followed by an optional 1- or 2-digit octal unit number and a colon.

[g,m] The user identification code (UIC) associated with the user file directory containing the desired file. The UIC consists of a group number and a member number.

filename The name of the file. A filename can be up to nine alphanumeric characters in length. The filename and type are always separated by a period.

type A means of distinguishing among various forms of one file. For example, a source FORTRAN program might be named COMP.FTN, while the object code associated with that program might be called COMP.OBJ. The file type and version are always separated by a semicolon (;).

version An octal number used to differentiate among various versions of a file. For example, when a file is first created, it is assigned a version number of 1. If the file is subsequently opened for editing, the original file is retained for backup, and a new file is created with the same filename and type, but with a version number of 2. The version number is in the range 0 through 77777.

/switch A 2-character ASCII name identifying a switch option. The switch itself may have three forms. If the switch designator, for examle is SW, then:

/SW Sets the switch action;

/-SW Negates the switch action, and

/NOSW Negates the switch action.

In addition, the switch identifier may be followed by any number of values. The permitted values are ASCII strings, octal numbers, or decimal numbers. The default for a value may be octal or decimal. Values which are terminated by a decimal point are explicitly decimal. Values preceded by a pound sign (#) are explicitly octal. Finally, any numeric value may be preceded by a + or - sign; plus is the defalt notation. If explicit octal (#) is used, the sign (if one is used) must precede the # (pound) sign. The following are valid switch specifications:

```
/SW:27:MAP:29.

/-SW

/NOSW:-#50:SWITCH
```

The number of permissible values and the switch interpretations themselves depend entirely on the particular task to which they are directed.

3.4 Volumes and Files

The following discussion of volumes and files is intended as a structural sketch of the procedures needed to create, name, and access files. Details appear under the discussion of the associated MCR commands.

In the previous section, filename strings were discussed. A filename, however, refers to a file on a file-structured volume, and several steps are required before such a file can actually be created and subsequently referred to by name.

Volumes are data carrying media which can be file structured. Disks are the normal file-structured volumes.

Before the file system can access or create a file from a user-supplied filename, the volume must be initialized, it must be mounted, and a user file directory must be created. These three steps are accomplished by the MCR INItvolume, MOUnt and User File Directory commands.

A file-structured volume is organized such that files contained on the volume can be retrieved by their names. Directories are used to map names to the physical locations on the volume where the referenced file is stored. The INItvolume command establishes the basic structural framework for these file directories. A volume which has not been processed by the INItvolume command is defined as a foreign volume; such a volume cannot be processed by the file system.

Each owner having a unique UIC should have a directory file of all the files created under his UIC. No other user of the system can access these files unless they know the UIC and have access privileges to the file. The User File Directory command establishes this directory file for a UIC. Once the directory file is established, the owner can create, reference, or delete files specific to his UIC.

Finally, any volume to be accessed by the file system must be made known to the system via the MOUnt command. Note that unmounted volumes can be accessed directly by user tasks, which is often a requirement in time-critical aplications.

Thus, before a filename can be used to access a file, the volume which is to contain the file must be initialized, the volume must be identified to the system via the MOUnt command, and a user file directory which will contain the name of the file must be created. Volume intialization and user file directory creation need only be performed once for a volume; mounting, however, must be done after each bootstrap of the system.

3.5 Defaults in File Specifiers

Any of the elements of the file specifier can be defaulted, except the filename. The system uses defaults as specified below:

String Element	System Default
dev:	SY:. The device-unit on which the system volume is mounted.
[g,m]	The default UIC specified during the sign-on procedure. This UIC is associated with the user terminal when the HELlo command is executed.
filename	No default. The filename must be specified explicitly.
type	A user meaningful file type. For example, if a listing file is created, the system assigns it a file type of LST when the type is not otherwise specifically given. A list of default file types is given below.
version	For input files, the most recent version number. For output files, the next higher version number.
switch	Switch defaults are established by the individual tasks which require them. The defaults are given in the task description.

3.6 Default File Types

The system has a set of file types which are used when the type element is omitted from a file specifier. Although the user can assign his own file types, system operations are facilitated if use is made of the file types presented in the following table. These mnemonic types are used by all system software to reflect actual file contents.

File Type	Meaning
CMD	A file containing a list of task or MCR commands (indirect file).
DAT	A data file, as opposed to a program file. Note that data files created by the ACQuire task use the short form of the experiment name for data files.
DIR	A directory file, for example, a user file directory.
EXP	A data file created by the ACQuire task. The type EXP is not explicitly used, rather a short form of the experiment name such as EX1 (EXperiment 1) as taken from the DASCOM Common partition is actually used.
FTN	A FORTRAN language source program.
LST	A listing file.
MAC	A MACRO-11 source program.
MAP	A Task Builder memory allocation file.
MLB	A user macro library.
MSG	The MO: pseudo-device message format file.
OBJ	An object program (output from MACRO-11 or FORTRAN).
OLB	An object module library.
SML	The system macro library.
TSK	A task image.

3.7 Indirect Files

An indirect file is a sequential file containing a list of commands exclusive to, and interpretable by a single task, usually a system-supplied component.

Indirect files are initiated by replacing the file specification command string required by a task with a filename string preceded by an at sign (@). For example, to initiate a file of DLG commands, the user would input:

```
>DLG @INPT.CMD
```

After DLG is initiated, it accesses the file INPT.CMD for all of its commands.

An indirect file may contain any command interpretable by the task to which it is directed, but no others. Indirect files may not contain indirect file references (i.e., only one level of command file indirection is permitted). A complete description of indirect files for use with MCR is contained in the RSX-11M Operator's Procedures Manual.

CHAPTER 4

GALE User Commands

4.1 Command Summary

GALE is implemented as a set of tasks running under the RSX-11M operating system. Users have access to all normal operating system commands and, additionally, to the GALE commands described in this chapter. Certain commands are privileged as under RSX-11M and as such are available only to a restricted number of users. The following is a list of all presently implemented GALE commands and RSX-11M commands required for operating GALE. Those denoted with a "*" are privileged or have some privileged functions. The commands denoted with a "+" also have other features which are fully described in the RSX-11M Operator's Procedures Manual. The operating instructions follow in the same order as this list.

ASN +

BYE

DISMOUNT * +

HELLO

MOUNT * +

4.2 Monitor Console Interface

The Terminal User communicates with the GALE system via a hardcopy terminal such as a DECwriter LA30 or LA36 or via a CRT device such as a Tektronix 4006 or 4012. The Monitor Console Routines (MCR) are the interface between the terminal and the GALE system. MCR input is initiated by entering unsolicited input on an unattached terminal which has displayed the default prompt, or by typing CTRL/C. If CTRL/C is entered, the system responds by displaying

```
MCR>
```

on the terminal, indicating that it will now accept an MCR command. When MCR is ready for another command, it displays the default prompt:

```
>
```

4.2.1 Command Syntax

It is not necessary to type the entire command name when submitting a command. MCR requires only the first three letters of a command, followed by the command parameters. If parameters exist, they must be preceded by at least one blank or tab.

The following example shows how the Time command may be specified. Note that the brackets [and] are used in the example only to indicate that the "E" is optional; the brackets are not actually entered as part of the command name.

```
MCR>TIM[E]<CR>
10:30:25 01-MAY-77
```

Thus, either

```
MCR>TIM<CR>
```

or

```
MCR>TIME<CR>
```

is acceptable.

NOTE

The angle brackets < and > are used to denote that the enclosed values are not taken literally as part of the command syntax. For example, <CR> indicates that a carriage return terminates the input line.

4.2.2 Keywords

Some commands use keywords which generally apply to a command argument. A keyword is similar in function to the switches described elsewhere. A keyword consists of a slash (/), followed by an ASCII identification, and optionally followd by an equal sign (=) and the value of the keyword, as follows:

```
/keyword=value
```

Keywords can be entered in any order. As an example of keyword usage, the Mount command for the Message Output pseudo-device requires the device name, MO:. A keyword can be appended to the device name specifying the name of the system message format file. Thus

```
MOU MO:/SMSG=NEWFILE.MSG
```

will cause the the file NEWFILE.MSG to be used for system message formats. Keywords are command specific and are defined with each command.

4.2.3 Comments

MCR will treat a line of text as a comment if the first character in the line is a semicolon (;). In addition, the exclamation mark (!) may be used to delimit comments in a command. The first exclamation mark starts the comment and the next exclamation mark or end of line terminates the comment.

All text between the two exclamation marks is ignored. For example:

```
MCR>;  THIS IS A COMMENT LINE

MCR>TIM !THIS IS A COMMENT STRING!
```

4.3 Terminal Commands

Command names are at least three characters in length. MCR accepts the first three characters of a command and searches for a blank, tab, carriage return or ESCape character. Therefore, embedded blanks are not allowed in a command name. If a command is entered incorrectly, an error message will be displayed at the entering terminal.

Syntactical descriptions of commands and messages described in this chapter observe the following notational conventions:

1. Lower case indicates a variable whose actual value is determined when the command is entered or the message is issued.

2. Brackets [and] enclose optional items. A syntactical element enclosed in brackets may or may not be included in the command, at the user's option. The one exception to this rule is the syntax for the specification for a UIC. The format for a UIC is [g,m], where the brackets are required syntatical elements.

3. Unless explicitly qualified, all numeric values required in a command may be entered as decimal or octal. Decimal values are indicated by a trailing period; octal values are indicated by the absence of a trailing period. Thus,

   ```
   255.
   ```

   and

   ```
   377
   ```

   have the same value. (The value 255. is base 10; the value 377 is base 8.)

4.3.1 Command Description Format

Each command is described using the following subheadings; those that do not apply are omitted.

COMMAND NAME (Abreviation)

This section is headed by the command name in capital letters. On the right margin, the full MCR command mnemonic is shown. Following the heading, the command's intent within the system is described.

Format:

The command format is given and all parmaters are described.

Examples:

Example uses are shown.

Command Error Messages:

Error messsages specific to the command appear in the section. The command may also produce messages listed in the Appendix pertaining to common error messages, however they are not listed here.

Notes

A list of special considerations that may prove helpful in assisting the user in the proper use of the command.

4.4 Commands

ASSIGN Command (ASN)

Function:

The ASsigN command defines, deletes, or displays logical device assignments. Logical device assignments are a means of associating logical names with physical devices, pseudo devices, or other logical devices. When a user assigns a logical name to a pseudo or logical device, the system resolves the assignment to the associated physical device. There are three types of logical device assignment: global, local and login.

1. Global assignments apply to all tasks running in the system. Only privileged terminals can issue ASsigN commands to define or delete global assignments.

2. Local assignments apply only to tasks initiated from the terminal used to make the assignments. Any terminal can define its own set of local assignments.

3. Login assignments occur only in systems that support multiuser protection. When a user issues a HELlo command to log into the system, the system automatically assigns the logical name SY: to the user's system device, which is the device that contains the user's files. The login assignments apply only to tasks initiated from the terminal while the current user is logged in. Only from a privileged terminal can an operator define or delete login assignments.

Local assignments have precedence over both login and global assignments; and login assignments have precedence over global assignments. When a user deassigns a local logical device name, the system defaults to any conflicting login assignments.

A logical device name has the same syntax as a real device unit; it consists of a 2-character ASCII name (alphabetic) and an optional 1- or 2-digit octal unit number, followed by a colon(:). The 2-character name can either be equivalent to a standard RSX-11M device name (for example, DK:) or it can be two letters picked at random (for example XY:).

When a user installs a task, or the system processes an ASSIGN LUN directive, the system scans the logical device table before the physical device table when searching for a specified device. As a result, a user can define logical device names that are identical to physical device names even though the logical and physical device names do not refer to the same physical device. The system resolves a specified device name to the first matching table entry found. If a logical device name is identical to a physical device name, the logical name has precedence.

Formats:

Nonprivileged Formats:

ASN ppnn:=llnn:

Assign the logical name llnn: to device ppnn:. The command establishes a local assignment for the issuing terminal.

ASN

Display at the issuing terminal all local and login logical device assignments.

ASN =[llnn:]

Delete the local assignment of the specified logical device name; or delete all local logical device assignments for the issuing terminal if the command omits llnn:.

Privileged Formats:

ASN ppnn:=llnn:/GBL

Define a global assignment that associates the logical name llnn: with device ppnn:.

ASN /GBL

Display all global, local, and login assignments for all terminals in the system.

ASN =[llnn:]/GBL

Delete the specified global logical device assignment or all global assignments if the command omits llnn:.

where:
pp - Physical, logical or pseudo device name
nn - Unit number
ll - Logical device name

Examples:

```
MCR>ASN DK:=DA:          !Define DA:

MCR>ASN DK2:=DA:/GBL     !Define DA:  as global

MCR>ASN =                !Delete all local assignments
```

Command Error Messages:

ASN -- LOGICAL DEVICE NOT IN SYSTEM

The specified logical device name was not defined and therefore could not be found in the logical device assignment table for the terminal.

BYE Command

Function:

The BYE command provides the user with a facility for logging-off from a terminal. The only command MCR recognizes after this command is executed is HELlo. Upon execution of the BYE command, all active nonprivileged tasks are aborted and a terminal termination message is printed.

Format:

BYE<CR>

Example:

```
MCR>BYE<CR>
HAVE A GOOD AFTERNOON
12-SEP-77 2:45 TT6:  LOGGED OFF
```

DISMOUNT Command (DMO)

Function:

The DisMOunt command declares that the GALE device specified in the command is logically offline. After a dismount operation, the device cannot be accessed by the associated Ancillary Control Processor (ACP). For GALE devices such as MO: and DA: a request is placed in the appropriate queue to delete the Volume Control Block (VCB), and the device is marked for dismount so that no additional requests for access will be honored. The VCB is a main memory resident control block that controls volume access in much the same way as for Files-11 volumes. The command is completed when the VCB is actually deleted; VCB deletion does not occur until all tasks accessing the device have released access. When the deaccessing process is complete, the system issues a message indicating that the dismount operation is complete. This command may only be issued from privileged terminals.

When the dismount has been completed, the message:

```
*** dev:  --  DISMOUNT COMPLETE
```

is issued to CO:. There may be a considerable delay between the issuance of the command and the printing of this message if a number of I/O requests are pending.

Format:

DMO[UNT] dev:

where dev: is the device-unit to be dismounted.

Example:

```
MCR>DMO MO:
```

This command causes the VCB for MO: to be deleted.

NOTE

The GALE system cannot function when MO: is dismounted.

Command Error Messages:

DMO -- ALREADY MARKED FOR DISMOUNT

The device-unit had been requested to be dismounted and was in the process of waiting for all accesses to be completed.

DMO -- NOT MOUNTABLE DEVICE

The specified device was not a mountable device and therefore could not be dismounted.

DMO -- NOT MOUNTED

The specified device was not mounted.

HELLO Command (HEL)

Function:

The HELLO command (HEL) allows the user to log onto a terminal and be identified as a valid user by the system. It also identifies the user as a general or privileged user.

Format:

HEL[LO] [UIC]<CR>

or

HEL[LO] user-name <CR>

The following prompt is then issued:

PASSWORD:

The user must enter the correct password immediately following the prompt. The user-entered password is not printed on the terminal.

Examples:

1. The log-on procedure using the user name is:

```
MCR>HEL USER<CR>
PASSWORD:    <CR>
MCR>
```

NOTE

The password entered is not printed on the terminal.

2. The log-on procedure using an explicit UIC is:

```
MCR>HEL [200,200]<CR>
PASSWORD:   <CR>
MCR>
```

NOTE

The square brackets, [and], are required around the UIC.

Command Error Messages:

HEL -- ACCOUNT FILE OPEN FAILURE

The account file was open for another user; or the disk containing the account file was not mounted. Retry the command.

HEL -- COMMAND INPUT ERROR

A system directive or the RUN command, rather than the HELLO command, has initiated the HELLO task. HELLO may only be initiated from a terminal directly.

HEL -- INVALID ACCOUNT

The name or UIC specified in the command is not stored in the account file; or the password specified does not match the name or the UIC given.

HEL -- LOGINS ARE DISABLED

The system was in the process of shutting down; or the command SET /NOLOGON has been issued. A user cannot log onto a terminal at these times.

HEL -- MESSAGE FILE ERROR nnn.

The system could not open the file LOGIN.TXT for a reason indicated by the FCS code nnn. Usually this means that the file does not exist.

HEL -- OTHER USER LOGGED ON

The issuing terminal was currently logged by another user. Only one user at a time can be logged onto a terminal.

HEL -- TERMINAL ALLOCATED TO OTHER USER

The issuing terminal has been allocated to another user. A user cannot log onto a terminal allocated to someone else.

MOUNT Command (MOU)

Function:

The MOUNT command (MOU) allows the user to start the Ancilliary Control Processor (ACP) associated with the device. Its usage is described in the RSX-11M Operator's Procedures Manual. The MOU command is also used to start the Message Output task associated with the MO: pseudo-device. This task is used for printing messages to the user terminal (TI:) and to the console log (CL:) devices. System and user message format file names may also be given as optional parameters.

Format:

```
MOU MO:[/UMSG=file-name2][/SMSG=file-name1] <CR>
```

NOTE

The file-names are in the standard RSX-11M file name format. The files may reside on any direct access volume, however the volume may not be dismounted until MO: is dismounted. If no system file-name is given, the default name is:

SY:[1,2]MOTFMT.MSG

and if the user file-name is omitted, the default name is:

SY:[1,2]USRFMT.MSG

Examples:

1. The message output pseudo-device may be mounted with the default message format files with the command:

```
MCR>MOU MO:<CR>
MCR>
```

2. To mount MO: with a specified user message format file, use the command:

```
MCR>MOU MO:/UMSG=DK1:[200,200]NEWFILE.MSG<CR>
MCR>
```

Command Error Messages:

MOU -- SYNTAX ERROR

The file descriptor is syntactically incorrect.

MOU -- CANNOT FIND <file-type> MESSAGE FILE

The given or default file of <file-type> was not found.

MOU -- ACP NOT IN SYSTEM

The ACP task is not installed.

MOU -- TASK NOT ACP

The task is not built as an ACP.

MOU -- NO POOL SPACE

There is not enough space in the system pool to allocate the required ACP control blocks.

MOU -- CANNOT MOUNT DEVICE

An error occurred in the ACP initialization phase.

CHAPTER 5

Data Acquisition Task (ACQ)

5.1 Introduction to ACQ

The Data Acquisition Program (ACQ) is used for reading experimental data and storing the data in an experiment data file. The logical program flow of ACQ is controlled via the GALE Data Acquisition Configuration File (DASCNF) which describes the actual hardware configuration of all modules to be included for acquisition. The ACQ timing, as described below, is controlled by 2 (or possibly 3) external hardware triggers. For each experiment, all of the data read is stored in one file which is named in a form corresponding to the experiment sequential number.

5.2 Experiment Organization

An experimental apparatus consists of a central apparatus with associated diagnostics. Each diagnostic is assigned to an experimentalist who is then responsible for its operation. All data read from all diagnostics is stored in one data file. Furthermore, data may only be taken when all of the experimentalists have their respective diagnostics prepared or have declared them explicitly or implicitly as off-line. As discussed below, diagnostics which are not fully prepared or which are improperly prepared are automatically set off-line for the forth-coming experiment. If the diagnostic is re-prepared by the time that the next experiment is to occur, the diagnostic is re-declared as on-line. In order to circumvent the problem of diagnostics being implicitly set off-line, generally only the System Manager or his delegate requests the taking of data. The experiment manager should coordinate the diagnostic preparations in order that the taking of data may be carried out successfully.

5.3 Preparing to Run ACQ

Before data may be taken using the ACQ task, the system must be prepared as given in the following steps:

1. The common area "DASCOM" must be installed in the system. DASCOM contains information necessary for coordinating ACQ with the user dialog task (DLG) and also parameters required in creating and retrieving data files. DASCOM may be installed in the system (by privileged users only) with the command:

```
MCR>INS [1,1]DASCOM[/PAR=DASCOM]
```

NOTE

Refer to the GALE System Programmers Handbook for information on creating the common partition DASCOM.

2. The pseudo devices MO: and DA: must be mounted; this is done with the commands:

```
MCR>MOU MO:
MCR>MOU DA:
```

3. The Ancilliary Control Processors containing the loadable driver routines necessary for the devices contained in the configuration file, and thus required for the successful operation of ACQ, must be installed. For example, if only one ACP is required, suppose it is called TSTACP, then the command:

```
MCR>INS TSTACP
```

would make the necessary ACP known to the system.

4. The logical devices which contain the configuration file DASCNF and to which the data files are to be written must be assigned to operational physical devices. The DASCNF file is contained on the logical device DS: and the data files will be written to the logical device DD:. These assignments may be accomplished with the commands:

```
MCR>ASN dev:=DS:/GBL
MCR>ASN dev:=DD:/GBL
```

where dev: is an appropriate physical device.

5. The ACQ task must be installed in the system. Installation may be accomplished with the command:

```
MCR>INS ACQ
```

5.4 Starting ACQ

The data acquisition task, ACQ, may only be requested by privileged users as discussed above. ACQ may be started by issuing the command:

```
MCR>ACQ[/cmd-sw1]...[/cmd-swn]
```

where /cmd-swn is an optional command switch as described below. ACQ must be installed and may not be started via a RUN command.

5.5 ACQ Commands

Commands to the task ACQ are in the form of switches appended to the MCR request to activate ACQ. The following is a summary of commands available:

/AUTO indicates that ACQ is to be automatically started upon receipt of the Start Trigger as described below under Trigger Requirements.

/BI provides information on the actual internal buffer requirements and the total number of errors detected.

/TI requests informational and error messages to be printed on the requesting terminal as well as CL:.

5.5.1 /AUTO Command

Initiating ACQ without the /AUTO switch performs a one-shot sequence of data acquisition; that is, only one data file is created and ACQ must be re-started for another data acquisition phase. If ACQ has been initiated without the /AUTO switch, the message:

```
ACQ -- START SHOT NUMBER <n>
```

is printed and ACQ is ready to receive data.

If ACQ is initiated with the /AUTO switch, the task goes into a wait state until activated by an external hardware trigger. Upon finishing a data taking sequence, ACQ returns to the wait state until another external trigger is received. This automatic mode is recommended for normal operations, as no main memory is required during the time that ACQ is awaiting the start trigger. In automatic mode, the message given above appears after each external start trigger is received. The default for the /AUTO switch is /NOAUTO, i.e., automatic mode must always be explicitly specified if desired.

5.5.2 /BI Command

Including the /BI switch with the request to activate ACQ causes the maximum used space of each of the four internally used buffers and the total number of detected errors to be printed. This switch is of particular use when optimizing the ACQ task size and as a check on the proper functioning of the DIOS I/O system.

After a data acquisition phase has been completed, ACQ prints the information in the following format:

```
ACQ -- BUFFER DIMENSIONS IN BYTES

        .$AST1  .$CNF1  .$CTR1  .$DAT1
           n1      n2      n3      n4

NUMBER OF ERRORS          m
```

where ni is the number of bytes used in the respective buffer and m is the number of errors detected.

This switch is always defaulted to /NOBI so that this information must always be explicitly requested. When used in conjunction with the /AUTO switch, /BI causes the indicated information to be printed after each data acquisition phase.

5.5.3 /TI Command

The /TI switch causes all information which is always printed on the CL: device also to be output on the terminal from which ACQ was requested. The default for /TI is an op-

tion at ACQ task build time, and may be set to either true or false. If /TI is default, it may be reset by using /NOTI so that messages are only output to CL:.

5.6 Trigger Requirements

The ACQ task requires at least two and possibly three triggers for proper operation. The triggers are listed below in order of their occurance.

1. START-TRIGGER

   The START-TRIGGER is required only in the event that ACQ was initiated with the /AUTO switch. This trigger indicates that the next experiment is about to be begun, and therefore activates ACQ. The control structure contained in the DASCNF file is loaded and the loadable drivers requested therein are allocated and loaded. The user should be certain that sufficient time is allowed for the required actions before the next trigger is received as otherwise data may be lost. After the START-TRIGGER is received, ACQ reports that it is active by printing the message:

   ACQ -- START SHOT NUMBER <n>

   ACQ runs as described above, with the exception that the task is not terminated after all actions are finished. Instead, ACQ returns to the wait state and awaits the next START-TRIGGER. This process is then repeated until terminated as discussed below.

2. PRE-TRIGGER

   Normally this trigger is received just prior to the start of an experiment. Upon receipt of the trigger, ACQ requests data from those modules which have been designated as delivering data before or during an experiment. In addition, ACQ reports the receipt of the trigger with the message:

   ACQ -- TRIGGER 1 RECEIVED

3. POST-TRIGGER

   Upon receipt of this trigger, all requests to devices which should have delivered data before or during the experiment, but which have not yet fin-

ished, are cancelled. Requests to deliver data are then sent to those devices which are so specified as to deliver data after the end of the experiment. The following message is also output:

ACQ -- TRIGGER 2 RECEIVED

5.7 Ending the Task

After all of the data have been read and the resulting data file is closed, ACQ indicates that it is finished by printing the message:

ACQ -- END

If ACQ was initiated without using the /AUTO switch, operation is then terminated. In order to take more data from the next experiment, the operator must re-start ACQ. If ACQ was started with the /AUTO option, it goes into a wait state until the next START-TRIGGER is received. In order to terminate ACQ when it has been initiated with the AUTO option, the operator must abort the task with the command:

MCR>ABO ...ACQ

5.8 Error Messages

In addition to the diagnostic messages given above, ACQ may report error conditions which occur due to file or other input/output operations. A full list of these messages is given in the Appendix.

NOTE

All messages are printed on the CL: pseudo device. It is recommended that CL: be directed to a physical device which has paper output and that the resulting day log be stored for future reference. In addition, ACQ prints messages on the requesting user's terminal as discussed in the /TI command.

In addition to the standard error messages, ACQ may also give the following messages:

ACQ -- DIAGNOSTIC <did> OPEN FOR MODIFY

A user is modifying the parameters of the diagnostic <did>. The diagnostic will be temporarily set off-line.

ACQ -- <buffer> SPACE EXHAUSTED

ACQ does not have enough buffer space to complete the requests contained in the configuration file. The ACQ task must be rebuilt using the Task Builder. The size of <buffer> should be increased as described in the GALE System Programmers Handbook. <buffer> may be one of: .$CNF1 or .$DAT1 or .$CTR1 or .$AST.

ACQ -- BAD SYNTAX

An invalid switch option was detected in the ACQ initiation request. The task must be re-started.

<sys-err-msg> MODULE <nam> <nmb>

This messages is appended to a system error message <sys-err-msg> when the cause of the error is determined to lie in a data acquisition module. <nmb> and <nam> are the unit number and module name as specified in the configuration file.

CHAPTER 6

AMOS File Transfer Task (AMS)

6.1 Introduction to AMS

The AMOS File Transfer Task (AMS) allows the user to transfer files from a PDP-11 to the Central Computer Facility or vice versa. Files may be transferred in transparent mode, wherein the object file is a bit for bit copy of the source file, or in translate mode, wherein the source file undergoes an EBCDIC to ASCII or ASCII to EBCDIC translation, depending upon the internal representation of the source file.

6.2 Preparing to Run AMS

Before calling AMS, the user must consider the following:

1. The pseudo device MO: must be known to the system and must be mounted. This may be accomplished with the command:

   MCR>MOU MO:

2. The pseudo device DA: must be known to the system and must be mounted. This may be done with the command:

   MCR>MOU DA:

3. The OS file transfer program must be submitted to OS/470. The user should be aware that the file transfer program does an ATTACH to the specified tele-communications line in the Central Computer Facility. Thus, only one file transfer program per

physical line is allowed. An attempt to start the file transfer program more than once for the same line will result in cancellation of the job by OS. The file transfer job may be submitted to OS/470 via the normal AMOS job submit commands. A typical submission might be:

```
XS PDE:TRANSFER.RUN
```

when the user has access to the AMOS file PDE:TRANSFER. After submitting the job to OS/470, files may be transferred as soon as the job has entered the "GO" step.

6.3 Starting AMS

One of the following methods may be used to start AMS:

1. MCR>AMS command string

   If this method is used, AMS will be loaded; it will execute the specified command and exit to MCR.

2. MCR>AMS

   If this method is used, after AMS is loaded, it responds with the following prompt:

   ```
   AMS>
   ```

   At this point, the user can enter an AMS command string to be executed, or an indirect file specifier may be entered to execute a series of commands. When AMS has completed processing a command string or has reached the end of file on an indirect file, AMS will once again issue a prompt. At this point, the user can enter another command string or indirect file specifier, or a CTRL/Z to terminate AMS and return to MCR.

6.4 AMS Command String

All commands to AMS are issued by entering AMS command strings through the operator console. The format of the AMS command strings is in general the following:

```
outfile/cmd-sw=infile/cmd-sw
```

where the elements in the command string may be described as:

outfile - is the output file specifier in the format: dev:[UIC]filename.typ;ver
infile - is the input file specifier in the format: dev:[UIC]filename.typ;ver
/cmd-sw - is the AMS command switch

NOTE

For AMOS file specifiers, dev:=AM: must be explicitly specified. The elements [UIC] and ;ver are ignored.

6.5 AMS Commands

AMS commands are in the form of switches appended to file specifiers. The following is a summary of the available commands.

/AS causes the source file to be translated from ASCII to EBCDIC or EBCDIC to ASCII, as appropriate

/RS Resets the link and stops the file transfer job running in OS/470

/XS Causes a job to be submitted to OS whose JCL is contained in the AMOS file.

6.5.1 /AS - Command

The /AS command is used to force ASCII to EBCDIC or EBCDIC to ASCII translation. The type of translation to be performed, depends upon the source code to be translated. The translation is performed in such a way, that the source image is preserved between the two files. If this switch is not given, the transfer defaults to no-translate.

NOTE

Source files transferred to AMOS containing TABs will have the TAB replaced with a back-slash character (\). AMOS files transferred to the PDP-11 will have backslash characters translated to TABs.

6.5.2 /RS - Command

The /RS command causes the logical link between OS/470 and RSX-11M to be reset, and in addition, the file transfer job running under OS/470 to be stopped. The user should be careful in using the file transfer job as it does not have a time limit. Thus, the job should only be submitted to OS/470 when files are actually to be transferrred, since the file transfer job blocks an OS Initiator which thus causes the normal OS operation to be slowed down.

6.5.3 /XS - Command

The /XS command is used to submit a job whose JCL is contained in the AMOS file-segment specified. In this case, only the output file specifier may be given in the AMS com-mand string.

6.6 Error Messages

AMS -- DEVICE NOT READY

The DA: pseudo-device is not mounted or the ACP containing the driver for the AMOS link is not installed.

AMS -- SYNTAX ERROR

The file-name parser has detected an error in the formal syntax of an RSX-11M or AMOS file-name.

CHAPTER 7

Configuration Task (CNF)

7.1 Introduction

The main function of the Configuration Task (CNF) is to convert the GALE Configuration File and the GALE Namelist File from task image to binary files as they are required in the Dialog Task (DLG) and the Data Acquisition Task (ACQ). Another function of CNF is to generate a listing of any system resident GALE-Structure.

7.2 Preparing to Run CNF

Before calling CNF, the user must consider the following:

1. The pseudo-device MO: must be known to the system and must be mounted:

   ```
   MCR>MOU MO:
   ```

2. The user must be logged on under the GALE system UIC, unless only GALE-Structure listings are to be produced.

3. The common DASCOM must be installed:

   ```
   MCR>INS DASCOM/PAR=DASCOM
   ```

7.3 Starting CNF

One of the following methods may be used to call CNF:

1. MCR>CNF command string <CR>

   If this method is used, CNF will be loaded; it will execute the specified command(s); and exit to MCR.

2. MCR>CNF <CR>

   If this method is used, when CNF is loaded, it responds with the following prompting message:

   CNF>

   At this point the user can enter a CNF command string to execute the desired CNF function, or he can enter an indirect file specifier.

When CNF has completed processing a command string, or has reached the end of file on an indirect file, CNF will once again issue a prompt. At this point the user can enter another command string or indirect file specifier, or he can enter a CTRL/Z to terminate CNF and return to MCR.

7.4 CNF Command String

All commands to CNF are issued by entering CNF command strings through the operator console. The format of the elements which comprise CNF command strings differs for each command. The terms used to describe these elements, are consistent, and described as follows:

outfile - is the output file specifier in the format:
dev:[UIC]filename.typ;ver

infile - is the input file specifier in the format:
dev:[UIC]filename.typ;ver

/cmd-sw - is the CNF command subswitch, normally in the format:
/sw : [opt. value]

7.4.1 List of File Specifiers

In general the CNF command string consists of only one input file specifier and one output file specifier. The only command which accepts a list of two output file specifiers is the /CH command. In this case input file specifiers are not allowed and no defaults are defined for missing fields in the output file specifiers.

7.4.2 Defaults in the File Specifiers

If any of the elements of the file specifiers, even the filename, is omitted, CNF uses a default, except during the /CH command. The default values are listed below:

dev:	SY0: -- the unit on which the system disk is mounted.
[UIC]	for input files the default is the UIC under which CNF is running (usually the GALE system UIC [1,101]), the UIC specified by the MCR SET command, or the default under which the user is logged into the system. For output files the default is the GALE system UIC.
filename:	There may be defaults for the output file specifier only. These defaults are dependent upon the command switches /NM or /CF (see /NM command and /CF-command).
typ	the default for the input file is TSK; The default for the output file is the three character experiment name as defined in the common DASCOM.
ver	The default for input files is the most recent version number. The default for output files is the next higher version number, or, version one if the file doesn't already exist on the output directory.

7.4.3 CNF Command Switches and Subswitches

The switch specification consists of a slash (/) followed by two letters, and is optionally followed by a subswitch seperated from the switch by a slash. The subswitch itself can have arguments. These arguments are seperated from the subswitch by a colon (:). Switch names can be of any length; but only the first two letters are significant.

7.5 CNF Commands

CNF commands are in the form of switches appended to optional output file specifiers. The following is a summary of commands and subswitches.

/NM generate the GALE Namelist File with recordlength equal to 16 bytes.

/CF generate the GALE Configuration File with recordlength equal to 64 bytes.

/CH change the internal defaults of the outfile specifiers. This command is only allowed in combination with the /LI [:n] subswitch.

7.5.1 /LI Subswitch

The only subswitch that CNF accepts is the list (/LI [:n]) switch in which case n listings of the current GALE-Structure are generated. CNF accepts this subswitch without any file specifiers as a List command.

Example:

CNF>/LI:3 <CR>

generates three copies of the current GALE-Structure using the following file specifiers:

SY0:[suic]DASNCB.exp for the GALE Namelist File
SY0:[suic]DASCNF.exp for the GALE Configuration File

where:
suic is the system UIC as defined in the common DASCOM
exp is the three character experiment name as defined in the common DASCOM.

7.5.2 /NM - Command

The /NM-command is used to create the GALE Namelist File. The optional subswitch is the /LI [:n] switch. Appending this switch to the output file specifier, the following default is used for the output file name:

filename: DASNCB

Example:

```
CNF>/NM/LI=DK1:DASNCB
```

Creates the GALE Namelist File with recordlength equal to 16 bytes by removing the label blocks from the input file and lists the current GALE-Structure. The file will be generated on the device SY0: under the GALE system UIC and will be named DASNCB.exp (exp is the three character experiment name).

NOTE

The /LI subswitch may only be given, if the GALE Configuration File is already resident on the system disk under the system UIC. If the default filename is to be used, only the switch need to be present as the output file specifier.

7.5.3 /CF - Command

The /CF-command is used to create the GALE Configuration File with recordlength equal to 64 bytes. The optional subswitch is the /LI [:n] switch. Appending this switch, the default filename for the output file is as follows:

filename: DASCNF

Example:

```
CNF>/CF/LI:2=DK2:DASCNF
```

Create the GALE Configuration File by removing the label blocks from the input file and generate two copies of the current GALE-Structure. The file will be generated on the device SY0: under the GALE system UIC and will be named

DASCNF.exp (exp is the three character experiment name).

NOTE

The /LI switch may only be given, if the GALE Namelist File is already present on the system device under the system UIC. If the default filename is to be used for the outfile, the /CF-switch is the only present part for the output file specifier.

7.5.4 /CH - Command

The /CH-command is used in combination with the /LI [:n] subswitch, to generate n copies of a listing from resident GALE system files whose names are different from the internal defaults.

Example:

```
CNF>DK:[10,10]DASNCB.EXP/NM/CH,DK:[10,10]DASCNF.EXP/CF/LI
```

Generate a listing of the GALE-Structure which is contained in the files described by the outfile specifier list.

NOTE

During the /CH-command there is no input file specifier, but a list of two output file specifiers. The /CH-switch must be appended to the first output file specifier, the /LI subswitch may be appended to any output file. No GALE system files will be generated.

7.6 CNF Error Messages

Errors encountered by CNF during processing are reported to the user via error messages listed on the operator console. In addition to the standard RSX-11M system error messages the following messages are possible:

CNF -- BAD INPUT FILE

The input file given in the CNF command string has an illegal format.

CNF -- SYNTAX ERROR

The CNF command string has an illegal format.

CHAPTER 8

System Dialog Task (DLG)

8.1 Introduction

The System Dialog task (DLG) permits the user to list and modify the contents of the system diagnostic configuration file (DASCNF). In addition, an overview of the configuration in the form of a diagnostic directory is automatically output.

The user should be aware of the fact that diagnostics contained in the system are password protected. This means that non-privileged users may only list and modifiy diagnostics (with their associated modules) which are assigned to his UIC. Other diagnostics contained in the system will not be listed and cannot be modified. On the other hand, priviliged users, in particular the system manager, can list and modify all diagnostics with their associated modules and in addition, the DASCNF Header Block. A number of parameters, as described below, are available to privileged users only.

8.2 Preparing to Run DLG

Before attempting to run the DLG task, the system must be prepared through the following procedure:

1. The volume containing the DASCNF file is assigned the logical device name DS:. This logical device must be assigned to a physical device, for example:

```
MCR>ASN DK2:=DS:/GBL
```

   Note that the logical name DS: is assigned globally.

2. The common area DASCOM must be installed in the system. DASCOM contains information necessary for coordinating the ACQ task with DLG. DASCOM may be installed in the system (by privileged users only) with the command:

```
MCR>INS [1,1]DASCOM[/PAR=DASCOM]
```

3. The MO: pseudo device must be mounted, for example:

```
MCR>MOU MO:
```

8.3 Starting DLG

After the system has been prepared as described above, DLG may be started by issuing the command

```
MCR>DLG
```

or

```
MCR>RUN $DLG
```

The first step in DLG is the output of a system diagnostic directory. This directory provides an overview of the diagnostics available for listing and modification to the current user. The directory is printed in the following format:

```
ID-CODE          UIC      DDB-NAME
```

After the directory is printed, the user options available are printed as follows:

```
OPTIONS ARE: /LI[ID] - LIST; /MO[ID] - MODIFY; ^Z - EXIT
[ID]::= ID CODE AS LISTED ABOVE
```

DLG is then ready to accept a command for listing or modification of a diagnostic, and indicates this by printing the prompt:

```
DLG>
```

When DLG has finished processing a command, the diagnostic directory is again printed and DLG is then ready for a new command or a CTRL/Z input to terminate operations.

8.4 DLG Command String

All commands to DLG are issued by entering a DLG command string through the user terminal. Commands consist of keywords followed by a diagnostic ID-CODE as given in the diagnostic directory. The commands have the format:

```
/keyword:<ID-CODE>
```

DLG input may also be in the form of an indirect file containing keyword commands and Namelist elements.

8.5 DLG Commands

The following is a summary of commands acceptable to DLG:

/LI:n list the control block parameters of diagnostic n.

/MO:n modify the control block parameters of diagnostic n.

For n=0, the DASCNF Header Block will be listed or available for modifications, depending upon the command.

8.5.1 /LI - Command

The list command prints the contents of the diagnostic control blocks associated with the given diagnostic ID-CODE. The output has the format:

```
<name>

<var1> [<code>]=<value list>
<var2> [<code>]=<value list>
         .
         .
<varn> [<code>]=<value list>

<name> MODULE <nbr> <type>

<var11> [<code>]=<value list>
<var12> [<code>]=<value list>
         .
         .
<var1n> [<code>]=<value list>
```

where the syntax elements have the meaning:

<name> Name of the diagnostic (max. 22 characters).

<nbr> Module number

<type> Module type (two characters).

<varn> Symbolic name of the variable (max. 6 characters). A list of all standard variable names is given below.

<code> The value type code for the values given in <value list>. <code> may be one of the following:

A	ASCII
B	Binary
D	Decimal word
E	Real
I	Decimal byte
O	Octal word
Y	Octal byte

<value list> A list (of length one or more) of the values contained in the variable whose name is <varn>.

In the event that the user has requested the listing of the Header Block, the following information is printed:

```
EXPERIMENT <exp-name> HEADER
NXT[D] = <shot-nbr>
TIM[I] = <time>
DAT[I] = <date>
```

where the syntax elements have the following meaning:

<exp-name> Name of the experiment

<shot-nbr> Sequential number of the next expected shot

<time> Time of day when the last experimental data was taken in the format hours - minutes - seconds

<date> Date when the last experimental data was taken in the format day - month - year

8.5.2 /MO - Command

The MO command is used to modify the parameter values of a configuration control block. After the modify command is received, DLG lists, as described in the /LI command, the contents of a control block and then awaits user input for modifying the contents of one or more parameters. The requests for modification are given in the form of Namelist inputs which have the syntax as outlined below. After all modifications on the current control block have be completed, DLG lists the contents of the next control block associated with the given diagnostic and again awaits modification requests. This process is repeated until all control blocks associated with the given diagnostic have been listed and/or modified. DLG then awaits a new command as described above.

8.6 Namelist Syntax

After a /MO command has been given, input is awaited conforming to the following syntax:

```
<input> ::= <text> | <text> ;  <input> | <ctrl>
<ctrl>  ::= <exit> | <retn> | <outp>
<exit>  ::= CTRL/Z *
<retn>  ::= // ** | two empty lines
<outp>  ::= / *** | one empty line
<text>  ::= <str> = <sequ> | <name> ?  ****
<name>  ::= ASCII character string (max.  6 char)
<sequ>  ::= <data> | <data> , <sequ>
<data>  ::= <val> | <cnt> * <val> | <vct> :  <vct> | <strng>
            | ' <strng> '
<strng> ::= a string of ASCII characters
<vct>   ::= <dec> | <oct> | <asc>
<asc>   ::= a single ASCII character
<oct>   ::= an octal integer
<dec>   ::= a decimal integer (with or without sign)
<val>   ::= <dec> | <oct> | <flt> | <asc>
<flt>   ::= <flt> E <exp> | real number
<exp>   ::= Decimal integer representing the exponent
<cnt>   ::= unsigned decimal integer
<str>   ::= <name> | <name> ( <cnt> )
```

* Causes DLG to exit

** Causes the Namelist input procedure to be ended and returns control to DLG. In indirect files, the // option must be used.

*** Interrupts the input mode and causes the current data block to be listed. In indirect files, the / option must be used.

**** Activates the Help function if defined for this variable. The Help function prints explanatory text, after which the user may then continue with more input.

8.7 DLG Standard Variable Names

A set of standard variable names has been defined to be used with DLG. The following three tables summarize these variable names, where "Name" gives the name of the variable, "Type" shows the value type code as outlined above, "P/N" denotes privileged (P) or normal access rights (N) and "Meaning" stands for a short explanation of the value accessed.

8.7.1 Symbolic Names for Header Block Offsets

Name	Type	P/N	Meaning
DAT	I	P	date of last shot with the format DD:MM:YY.
TIM	I	P	time of last shot with the format HH:MM:SS.
NXT	D	P	shot number of next data file to be written.

8.7.2 Symbolic Names for DDB Offsets

Name	Type	P/N	Meaning
CTL	B	P	monitoring and control diagnostic (CTL=1).
DAQ	B	P	data acquisition diagnostic (DAQ=1).
DID	Y	P	diagnostic ID code in the range 1 to 255.
LCL	B	P	local diagnostic (LCL=1).
MLN	Y	P	DDB message string length in characters (MLN<=22).
MSG	A	P	user message string consisting of up to 22 characters.
ONL	B	N	ONL=1 means diagnostic is online.
RMT	B	P	RMT=1 means diagnostic is remote.
RSA	O	P	remote station address.
UIC	O	P	UIC of diagnostic owner.

8.7.3 Symbolic Names for MCB Offsets

Name	Type	P/N	Meaning
ACP	A	P	two character ASCII partial task name of the ACP containing the driver for the module. This field is zero for build-in drivers.
ADR	O	P	device register address or CAMAC address.
CAM	B	P	if set, indicates that module is a CAMAC device.
CTL	B	P	control module.
DCT	D	N	number of data words desired.
DFM	Y	N	coded data format
DLN	Y	N	length in bytes of data item
GEN	B	N	if set, indicates that the module generates data
HLD	B	N	if set, do not unload the module until after the shot
INT	B	P	if set, indicates that the module can generate interrupts
MID	Y	P	module id code
PRE	B	N	begin reading data preceding shot
PRI	Y	P	module interrupt priority; ignored if CAM is set or if INT is not set
PST	B	N	begin reading data after shot
TYP	A	P	two character ASCII module type code
UNIT	Y	P	module unit number
VCT	Y	P	module interrupt vector address divided by 4; ignored if CAM is set or if INT is not set.

8.8 Example DLG Terminal Session

In order to illustrate the syntax features of DLG, a typical terminal session for a non-privileged user is given below. It is assumed that the user has successfully logged on to the terminal via the HELlo command and that the UIC thus associated with the terminal is known to GALE. DLG is then started with the command:

```
>DLG <CR>
```

whereupon DLG responds by printing the following:

```
        1       [200,200]       DIAGNOSTIC 1
        2       [200,200]       DIAGNOSTIC 2

OPTIONS ARE:  /LI[ID] - LIST;  /MO[ID] - MODIFY;  ^ Z - EXIT
[ID] ::= ID CODE AS LISTED ABOVE
DLG>
```

The above output indicates that the user has control over two diagnostics named DIAGNOSTIC 1 and DIAGNOSTIC 2. Note that other users known to the system, if any, are not shown. Furthermore, it is evident that the user is logged on under the UIC [200,200].

8.8.1 Listing a Diagnostic

The user may now select to list or modify the parameters of a diagnostic or to exit from the DLG task. Assume that the user wishes to list the parameters associated with DIAGNOSTIC 1. This may be accomplished by giving the input:

```
DLG>/LI:1 <CR>
```

The parmaters available to the user are then listed, for example:

```
DIAGNOSTIC 1

DAQ      [B] =1
CTL      [B] =0
UPR      [A] =TEST DIAGNOSTIC
UPR2     [D] =25
UPR3     [E] =5.0000E 01
UPR4     [D] =9    10    11    12

DIAGNOSTIC 1 MODULE 1 MX

ECHAN    [Y] =7
EXTMAP   [D] =0
MUXMOD   [A] =S
DCT      [D] =0
DFM      [Y] =0
CTL      [B] =0
GEN      [B] =1
DLN      [Y] =1
PRE      [B] =0
PST      [B] =0
HLD      [B] =0
```

```
        1       [200,200]       DIAGNOSTIC 1
        2       [200,200]       DIAGNOSTIC 2

OPTIONS ARE:  /LI[ID] - LIST;  /MO[ID] - MODIFY;  ^ Z - EXIT
[ID] ::= ID CODE AS LISTED ABOVE
DLG>
```

It is evident from the above listing, that DIAGNOSTIC 1 contiains only one Module, MX1:. DLG indicates that it is ready to accept a new command by the prompt:

```
DLG>
```

8.8.2 Modifying a Diagnostic

Now that the user has listed the current settings of the diagnostic, he may have determined that some of them need to be changed. Using the above listing as an example, DIAGNOSTIC 1 may be modified by entering the DLG command:

```
DLG>/MO:1 <CR>
```

whereupon DLG enters the modification mode. The first data block available for modification is listed again, as follows:

```
DIAGNOSTIC 1

DAQ      [B] =1
CTL      [B] =0
UPR      [A] =OPERATIONAL
UPR2     [D] =25
UPR3     [E] =5.0000E 01
UPR4     [D] =9     10     11     12
INPUT?
```

The message INPUT? indicates that the user may now enter Namelist elements to be modified. The elements in the current block may have been changed so:

```
INPUT?  DAQ=T;  UPR=NOT OPERATIONAL;
MORE?  UPR2=14;
MORE?  UPR3=52.;
MORE?  /
```

After the first request for input, DLG shows that it is ready for more input by printing "MORE?", as shown above. Entering a slash (/) causes DLG to temporarily go into list mode and list the contents of the current block. Continuing the above example, DLG would then respond with:

```
DIAGNOSTIC 1

DAQ      [B] =1
CTL      [B] =0
UPR      [A] =NOT OPERATIONAL
UPR2     [D] =14
UPR3     [E] =5.2000E 01
UPR4     [D] =9     10     11     12
MORE?
```

Modifications may be terminated on the current block by typing a double slash (//). This causes DLG to terminate modifications on the current block and the print contents of the next block. If the current block is the last block, then the contents of the directory are again printed and DLG awaits a new command.

8.8.3 DLG Help Function

The Help function, when requested, provides the terminal user with additional information about the variable. In the example, had the user replied with the request for more input with:

```
MORE?  DAQ?
```

DLG would reply with the message:

```
BIT SET MEANS:  DATA ACQUISITION DIAGNOSTIC
MORE?
```

and then await more input.

8.8.4 Vector Input

Two types of vector input are recognized by DLG. In the first case, the elements of the vector may be individually set to the desired values. In the example, the variable UPR4 may be modified in one line with:

```
INPUT?  UPR4=5,6,8,9;
MORE?  /
```

which would result in:

```
DIAGNOSTIC 1

DAQ      [B] =1
CTL      [B] =0
UPR      [A] =TEST DIAGNOSTIC
UPR2     [D] =25
UPR3     [E] =5.0000E 01
UPR4     [D] =5     6     8     9
```

In the second case, a vector may be modified in a sequential manner through the use of the colon (:) facility. For instance, the user might reply with:

```
INPUT?  UPR4=15:18;
MORE?  /
```

in which event the elements of the vector UPR4 would be sequentially set to the values 15 through 18, as follows.

```
DIAGNOSTIC 1

DAQ      [B] =1
CTL      [B] =0
UPR      [A] =TEST DIAGNOSTIC
UPR2     [D] =25
UPR3     [E] =5.0000E 01
UPR4     [D] =15    16    17    18
```

Note that this mode is not available for real [E] variables.

8.9 DLG Error Messages

Errors encountered by DLG during processing are reported to the user via error messages on the user terminal. Messages may be issued either from DLG direct or from the Namelist processor; these are differentiated by the error message heading, either DLG or NML. In addition to the standard RSX-11M system error messages, the following error messages are possible:

8.9.1 DLG Messages

DLG -- NO NCB DEFINED FOR THIS BLOCK

There exist no NCB's for this control block (the NCB pointer is zero), or only privileged NCB's are defined for this control block and the requesting user is not privileged.

DLG -- <buffer> SPACE EXHAUSTED

The internal buffer space is filled. The DLG task is terminated. This condition is usually caused by an NCB list which is longer than the maximum allowed or by an incorrectly defined NCB list. If the condition persists, DLG must be rebuilt specifying a larger buffer area. Buffer may be .$BUF1 or .$BUF2 or .$BUF3.

DLG -- BAD SYNTAX

The input line is syntactically incorrect. Those values which proceed the detected error are modified as requested.

DLG -- PRIVILEGED COMMAND

The user is not known to the system or the ID Code requested does not belong to the user. This message is also issued in the event that a non-privileged user attempts to access the Header.

DLG -- ID CODE NOT IN SYSTEM

The user has attempted to access a non-existent module.

DLG -- DIAGNOSTIC IN USE; TRY LATER

The requested module is currenty being modified by another user. Caution should be exercised when this condition arises, as two or more users have access to the same module.

8.9.2 NML Messages

NML -- NCB LIST ERROR

An invalid type code was detected in the NCB list. The list will not be processed

NML -- INDEX OVERFLOW

The index given for a vector element lies outside of the valid index values.

NML -- COUNT LESS THAN OR EQUAL 0

The user has input an invalid numeric vector value. When using the form A*B, the condition A>0 must be true.

NML -- BUFFER TOO SMALL

More data elements were input than were defined in the vector variable.

NML -- NAME NOT FOUND

The input line contains an unknown variable name.

NML -- NEGATIVE VECTOR

The values given for a vector variable A:B are such that A>B.

NML -- BAD SYNTAX

The input line contains a syntactically incorrect element.

NML -- NAME LONGER THAN 6 CHARACTERS

A variable name longer than 6 characters was detected.

NML -- BAD SYNTAX FOR REAL

A real variable was detected with an invalid format. The real value format is: +/- XXX.YYY E +/- ZZZ.

NML -- BAD SYNTAX FOR BINARY INPUT

Binary input may only be given in the form:

1 or Y or T to set the variable

0 or N or F to reset the variable

NML -- NO HELP DEFINED FOR THIS NCB

The user has requested further information about a variable for which none is available.

APPENDIX A

GALE Error Messages

ACQ -- DIAGNOSTIC <did> OPEN FOR MODIFY

A user is modifying the parameters of the diagnostic <did>. The diagnostic will be temporarily set off-line.

ACQ -- <buffer> SPACE EXHAUSTED

ACQ does not have enough buffer space to complete the requests contained in the configuration file. The ACQ task must be rebuilt using the Task Builder. The size of <buffer> should be increased as described in the GALE System Programmers Handbook. <buffer> may be one of: .$CNF1 or .$DAT1 or .$CTR1 or .$AST.

ACQ -- BAD SYNTAX

An invalid switch option was detected in the ACQ initiation request. The task must be re-started.

<sys-err-msg> MODULE <nam> <nmb>

This messages is appended to a system error message <sys-err-msg> when the cause of the error is determined to lie in a data acquisition module. <nmb> and <nam> are the unit number and module name as specified in the configuration file.

AMS -- DEVICE NOT READY

The DA: pseudo-device is not mounted or the ACP containing the driver for the AMOS link is not installed.

AMS -- SYNTAX ERROR

The file-name parser has detected an error in the formal syntax of an RSX-11M or AMOS file-name.

ASN -- LOGICAL DEVICE NOT IN SYSTEM

The specified logical device name was not defined and therefore could not be found in the logical device assignment table for the terminal.

CNF -- BAD INPUT FILE

The input file given in the CNF command string has an illegal format.

CNF -- SYNTAX ERROR

The CNF command string has an illegal format.

DLG -- NO NCB DEFINED FOR THIS BLOCK

There exist no NCB's for this control block (the NCB pointer is zero), or only privileged NCB's are defined for this control block and the requesting user is not privileged.

DLG -- <buffer> SPACE EXHAUSTED

The internal buffer space is filled. The DLG task is terminated. This condition is usually caused by an NCB list which is longer than the maximum allowed or by an incorrectly defined NCB list. If the condition persists, DLG must be rebuilt specifying a larger buffer area. Buffer may be .$BUF1 or .$BUF2 or .$BUF3.

DLG -- BAD SYNTAX

The input line is syntactically incorrect. Those values which preceed the detected error are modified as requested.

DLG -- PRIVILEGED COMMAND

The user is not known to the system or the ID Code requested does not belong to the user. This message is also issued in the event that a non-privileged user attempts to access the Header.

DLG -- ID CODE NOT IN SYSTEM

The user has attempted to access a non-existent module.

DLG -- DIAGNOSTIC IN USE; TRY LATER

The requested module is currently being modified by another user. Caution should be exercised when this condition arises, as two or more users have access to the same module.

DMO -- ALREADY MARKED FOR DISMOUNT

The device-unit had been requested to be dismounted and was in the process of waiting for all accesses to be completed.

DMO -- NOT MOUNTABLE DEVICE

The specified device was not a mountable device and therefore could not be dismounted.

DMO -- NOT MOUNTED

The specified device was not mounted.

HEL -- ACCOUNT FILE OPEN FAILURE

The account file was open for another user; or the disk containing the account file was not mounted. Retry the command.

HEL -- COMMAND INPUT ERROR

A system directive or the RUN command, rather than the HELLO command, has initiated the HELLO task. HELLO may only be initiated from a terminal directly.

HEL -- INVALID ACCOUNT

The name or UIC specified in the command is not stored in the account file; or the password specified does not match the name or the UIC given.

HEL -- LOGINS ARE DISABLED

The system was in the process of shutting down; or the command SET /NOLOGON has been issued. A user cannot log onto a terminal at these times.

HEL -- MESSAGE FILE ERROR nnn.

The system could not open the file LOGIN.TXT for a reason indicated by the FCS code nnn. Usually this means that the file does not exist.

HEL -- OTHER USER LOGGED ON

The issuing terminal was currently logged by another user. Only one user at a time can be logged onto a terminal.

HEL -- TERMINAL ALLOCATED TO OTHER USER

The issuing terminal has been allocated to another user. A user cannot log onto a terminal allocated to someone else.

MOU -- SYNTAX ERROR

The file descriptor is syntactically incorrect.

MOU -- CANNOT FIND <file-type> MESSAGE FILE

The given or default file of <file-type> was not found.

MOU -- ACP NOT IN SYSTEM

The ACP task is not installed.

MOU -- TASK NOT ACP

The task is not built as an ACP.

MOU -- NO POOL SPACE

There is not enough space in the system pool to allocate the required ACP control blocks.

MOU -- CANNOT MOUNT DEVICE

An error occurred in the ACP initialization phase.

NML -- NCB LIST ERROR

An invalid type code was detected in the NCB list. The list will not be processed

NML -- INDEX OVERFLOW

The index given for a vector element lies outside of the valid index values.

NML -- COUNT LESS THAN OR EQUAL 0

The user has input an invalid numeric vector value. When using the form A*B, the condition A>0 must be true.

NML -- BUFFER TOO SMALL

More data elements were input than were defined in the vector variable.

NML -- NAME NOT FOUND

The input line contains an unknown variable name.

NML -- NEGATIVE VECTOR

The values given for a vector variable A:B are such that A>B.

NML -- BAD SYNTAX

The input line contains a syntactically incorrect element.

NML -- NAME LONGER THAN 6 CHARACTERS

A variable name longer than 6 characters was detected.

NML -- BAD SYNTAX FOR REAL

A real variable was detected with an invalid format. The real value format is: +/- XXX.YYY E +/- ZZZ.

NML -- BAD SYNTAX FOR BINARY INPUT

Binary input may only be given in the form:

1 or Y or T to set the variable
0 or N or F to reset the variable

NML -- NO HELP DEFINED FOR THIS NCB

The user has requested further information about a variable for which none is available.

READER'S COMMENTS

NOTE: THIS FORM IS FOR DOCUMENT COMMENTS ONLY.

Did you find errors in this manual? If so, specify by page.

Did you find this manual understandable, usable and well organized? Please make suggestions for improvement.

Is there sufficient documentation on associated system programs required for use of the software described in this manual? If not, what material is missing and where should it be placed?

Please turn over

Please indicate the type of user/reader that you most nearly represent.

[ ] Assembly language programmer

[ ] Higher-level language programmer

[ ] Occasional programmer (experienced)

[ ] User with little programming experience

[ ] Student programmer

[ ] Non-programmer

If you desire to have your name put on the PDE documentation mailing list, please indicate so here.....

[ ]

NAME_____DATE_____

ORGANIZATION_____

STREET_____

CITY_____

STATE_____ZIP CODE_____

COUNTRY_____

RETURN TO:

PDE PROJEKT DATENERFASSUNG
INSTITUTE FOR PLASMAPHYSICS
D-8046 GARCHING
WEST GERMANY