GOOSY
Id.: gm_vaxvms
Version: 4.0
Date: September, 16 1987
Revised: August, 01 1995

GSI **O**nline **O**ffline **S** **Y**stem

OpenVMS Introduction

H.G. Essel, M. Richter, W. Schiebel

August, 01 1995

GSI, Gesellschaft für Schwerionenforschung mbH
Postfach 11 05 52, Planckstraße 1, D-64220 Darmstadt
Tel. (0 6159) 71–0

List of Figures

Chapter 1

Preface

GOOSY Copy Right

Conventions used in this Document

[Fn], [PFn], [1], [Do], or [Return] key — All key in frame boxes refer to the special keypads on VTx20 compatible terminals like VT220, VT320, VT330, VT340, VT420, VT520, PECAD, PERICOM terminals or DECterm windows under DECwindows/Motif on top or right to the main keyboard, to control characters, or to the delete and return keys of the main keyboard.

<Fn>, <PFn>, <KPn>, <Do>, or <Ctrl>— This is the alternative way of writing the keypad or control keys.

[GOLD], <GOLD>— The [PF1] key is called [GOLD] in most utility programs using the keypad.

PERICOM— On the PERICOM terminal keyboard the function keys are marked opposite to all other terminals, i.e. the 4 [PFn] of the rightmost VTx20 compatible keypad are named [Fn] and the 20 [Fn] keys on the top of each VTx20 compatible keyboard are named [PFn] on a PERICOM.

[Return]— The [Return] is not shown in formats and examples. Assume that you must press [Return] after typing a command or other input to the system unless instructed otherwise.

[Enter]— If your terminal is connected to IBM, the [Enter] key terminates all command lines.

`Ctrl` **key** — The `Ctrl` box followed by a letter means that you must type the letter while holding down the `Ctrl` key (like the `Shift` key for capital letters). Here is an example:

- `Ctrl` `Z` means hold down the `Ctrl` key and type the letter `Z`.

`PFn` **key** — The `PFn` followed by a number means that you must press the `PFn` key and *then* type the number. Here is an example:

- `PF1` `6` press the `PF1` key and then type the number 6 on the main keyboard.

`PFn` **or** `Fn` **keys** — Any `PFn` or `Fn` key means that you just press this key. Here is an example:

- `PF2` means press the `PF2` key.

Examples— Examples in this manual show both system output (prompts, messages, and displays) and user input, which are all written in `typewriter` style. The user input is normally written in capital letters. Generally there is no case sensitive input in GOOSY, except in cases noted explicitly. In UNIX all input and with it user and file names are case sensitive, that means for TCP/IP services like Telnet, FTP, or SMTP mail one has to define node names, user names, and file names in double quotes "name" to keep the case valid for OpenVMS input. Keywords are printed with uppercase characters, parameters to be replaced by actual values with lowercase characters. The computer output might differ depending on the Alpha AXP or VAX system you are connected to, on the program version described, and on other circumstances. So do not expect identical computer output in all cases.

1.1 OpenVMS Advisory Service

In any case of computer or network problems please contact the Operators in the Computer Center room 1.250, tel. 2515 or the User's Consulting office room 2.244, tel. 2555.
The authors of this manual and their main fields for OpenVMS advisory services are:

Wolfgang Schiebel OpenVMS and network system manager (room 1.252, tel. 2498)

Hans-Georg Essel OpenVMS advisor (room 2.262, tel. 2491)

Mathias Richter OpenVMS advisor (room 2.262, tel. 2394)

Hans-Georg Schmidt Ethernet network manager (room 1.253, tel. 2497)

Ruth Thieme OpenVMS advisor (room 1.251, tel. 2556)

Digital Equipment (DEC) Field Service please contact only if none of the above persons are available in case of hardware problems (container in the yard between entries B and C, tel. 2413, paging 122471...)

1.2 Further GOOSY Manuals

The GOOSY system is described in the following manuals:

- GOOSY Introduction and Command Summary
- GOOSY Data Acquisition and Analysis
- GOOSY Data Management
- GOOSY Data Management Commands
- GOOSY Display
- GOOSY Hardware
- GOOSY DCL Procedures. GOOSY Error Recovery
- GOOSY Manual
- GOOSY Commands

Further manuals are available:

- GOOSY Buffer structures
- GOOSY PAW Server

- GOOSY LMD List Mode Data Generator
- SBS Single Branch System
- TCP-Package
- TRIGGER Bus
- VME Introduction
- OpenVMS Introduction

1.3 Intended Audience

This is a short introduction for the Alpha AXP or VAX users at GSI. It familiarizes the user with the basic functions of the Alpha AXP or VAX operating system (OpenVMS). This introduction cannot be complete nor can it replace any Alpha AXP, VAX, or OpenVMS manual. It should just help a newcomer to get started. This manual is available for printing on the laser print server in room 2.223 with the OpenVMS DCL command

```
$ PS G POSTD GOO$DOC:GM_VAXVMS.PS
```

This manual is not a manual for the use of GOOSY. For GOOSY users we recommend the 'GOOSY Introduction' and/or any detailed GOOSY manual.

The authors would be grateful for any critical comment or any suggestion about this manual.

Chapter 2

Login and Logout

2.1 The Computer Account

As a new user of a GSI computer you have to be accounted for. Through this procedure the system manager (for OpenVMS see names on page 5) sets up a disk directory for you and you will receive a username and a password.

You must change your password immediately after the very first login by the command

```
$ SET PASSWORD
```

Only after you have received an account you can proceed to any terminal for login.

2.2 General Remarks on GSI Computers

At GSI a large number of Alpha AXP or VAX computers namely Alpha workstations or VAXstations is used by experiment groups for data collection and data analysis, for the accelerator control system, for the safety department control system, and for a printed circuit layout CAD system. The OpenVMS operating system is very popular and gratefully accepted by most experiment groups at GSI, besides the IBM mainframe under MVS and several UNIX workstations from DEC, IBM, and HP.

Most computers at GSI are connected to each other by a large Ethernet/FDDI network (FDDI Fiber Distributed Data Interface with transparent bridges to Ethernet) available in almost all rooms. This network is separated by bridges into several segments reducing the overall data traffic and avoiding disturbances of the whole network by one local error source. Several (up to 32) FDDI rings are separated by a very fast cross bar switch, the GIGAswitch, which switches FDDI packets from source rings to destination rings (different rings simultaneously). Several Alpha AXP and VAX computers are connected directly to the GIGAswitch (building their own little ring). FDDI is used mainly as a backbone for Ethernet segments. In fig. 2.1 on page 10 you see a generalized picture of the computer network.

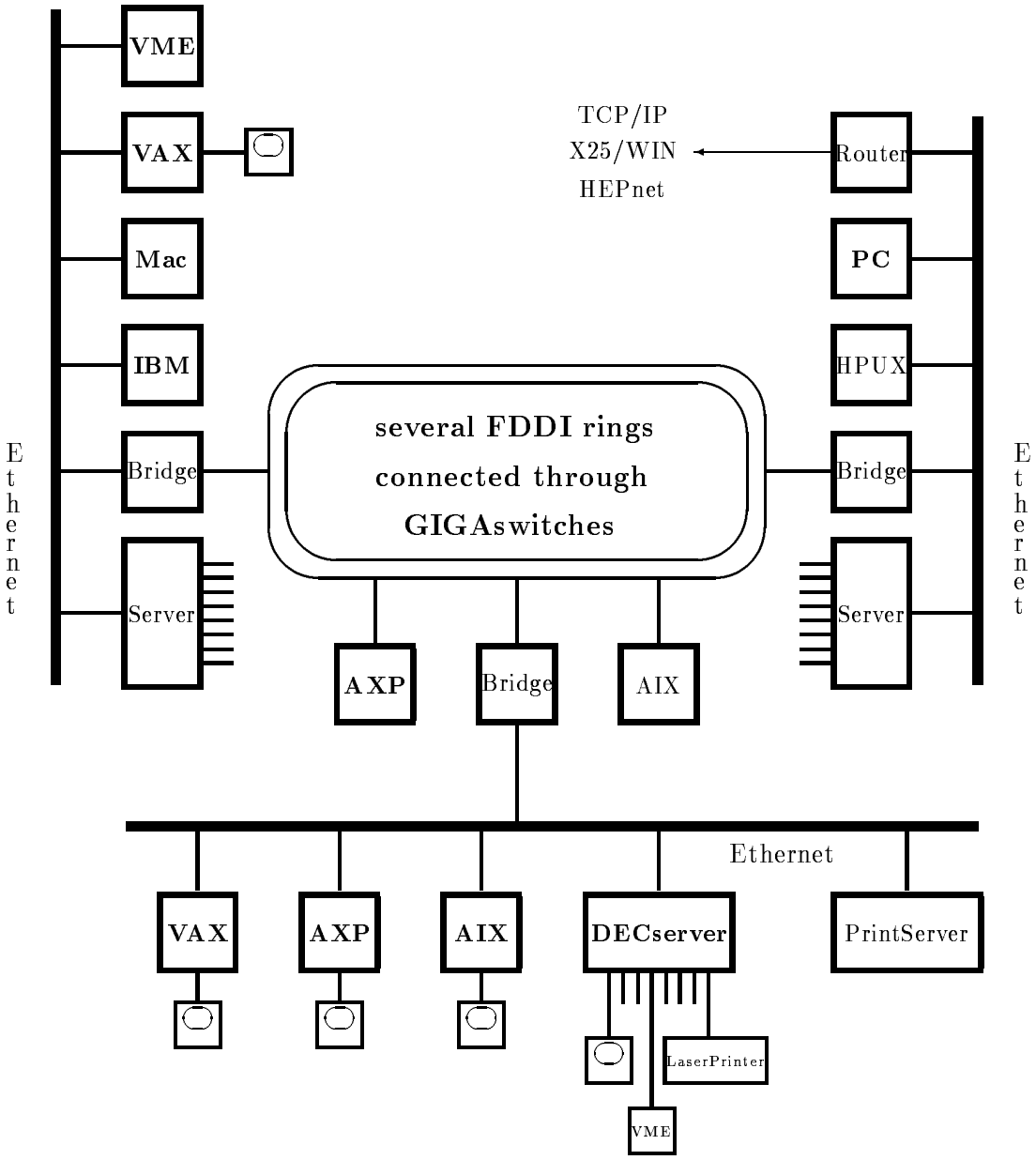


Figure 2.1: Schematic view of computers and terminals in the Ethernet/FDDI network

Although the Alpha AXP or VAXstations and the Ethernet are easy to use, users are strictly prohibited from making any changes or re-configurations on the computer hardware or on the Ethernet cabling by their own. Specifically:

- NEVER use 93 Ω IBM 327x cable or 75 Ω video cable as a thin wire Ethernet cable. Use ONLY specific 50 Ω Ethernet cables available in the GSI stock.
- Never make a stub from a T-connector to the computer, i.e. never place the T-connector close to the wall and feed only one cable to the computer, this corrupts the network.
- The T-connector of the thin wire Ethernet cabling must be as close to the computer interface as possible, i.e. from each of the two cable connectors on the wall a separate cable must go to the T-connector close to the computer.
- NEVER leave the thin wire Ethernet cable connection open. Never remove the cable bridge on the wall connectors without immediate replacement.

Everyone must contact the Computer Advisory Service (for OpenVMS, Ethernet and terminals see names on page 5) in case of hardware re-configurations. They will provide you with the necessary service. They are also trying to keep a record of the whole network hardware which requires your help.
Keep your hands off the computer hardware and the Ethernet cabling!
All computers at GSI might be corrupted even by a single source of hardware malfunction!

Since most of the computer and network components are under maintenance contract, please, never unpack or install new computer equipment by your own (like Alpha AXP, VAXstations, DECstations, add-on memory, DEC terminals, magnetic disks, interfaces, printers, or tape drives). Call the Computer Advisory Service in any case for unpacking and installation. Otherwise GSI might run into problems with the warranty, the service and the software installation. A copy of the delivery sheet (from the stockroom personal) should be passed to the Computer Advisory Service. They also need the software license sheets (PAK) coming together with the computer. Also, never re-configure existing computer equipment by your own (like Alpha AXP, VAXstations, DECstations, add-on memory, DEC terminals, magnetic disks, interfaces, printers, or tape drives). You should also contact the Computer Advisory Service for connecting Xwindow terminals, IBM compatible PCs or Apple Macintosh to Ethernet. Not all PC Ethernet cards are supported by the network software and may disturb the network hardware.

2.3 User Interfaces to GSI Computers

The user communicates with the computers by text, graphics, or Xwindow terminals. Graphics terminals can be part of a workstation, i.e. connected directly to a computer. There are also Xwindow terminals connected via Ethernet to the computer. Text terminals and some simple

graphics terminals are connected to Terminal Servers which for their part are connected to Ethernet. A Terminal Server, e.g. a DECserver, is a device connecting terminals and printers via an Ethernet coaxcable to all Alpha AXP, VAX, ULTRIX, IBM RS/6000 UNIX, HP UNIX, and Digital UNIX (former OSF/1), computers at GSI, to the IBM mainframe computer, and to several VME processor boards (OS/9, LynxOS, and pSOS). Laser printers are also connected to Terminal Servers allowing access from all computers. The DECservers are using the LAT (Local Area Transport) protocol for the connections between themselves and the computers. In other words, all computers, terminals, and printers are linked together allowing access to each other across the Ethernet/FDDI network.

There are several types of terminals available for GSI computers

1. a text (ASCII) terminal compatible to a VTx20 (e.g. VT220, VT320, VT330, VT340, VT420, VT520, PECAD, or Pericom) connected via a Terminal Server to Ethernet at GSI to login for OpenVMS, ULTRIX (on a DECstation), and MVS (on the IBM mainframe) or to VME processors. Connected to a DECserver-300 or -700 a direct TCP/IP Telnet connection can be established.

2. a text (ASCII) terminal compatible to a VTx20 connected directly to an Alpha AXP or VAX or DECstation.

3. a window graphics terminal under DECwindows/Motif connected directly to an Alpha AXP or a VAXstation or via a Xwindow terminal or an IBM PC or an Apple Macintosh remotely connected to an Alpha AXP or VAX via Ethernet.

4. a window graphics terminal connected directly to a VAXstation under the older VWS window system which will not be described in this manual.

5. a text terminal compatible VT-window (DECterm) under DECwindows/Motif (or VWS).

6. a simulation of a Terminal Server session after logged in a OpenVMS system using the DCL command `$ CON` or `$ SET HOST/LAT`

7. an IBM compatible PC with Pathworks for DOS simulating a text terminal to login on an Alpha AXP or a VAX or a DECstation.

8. an Apple Macintosh with Pathworks for Mac simulating a text terminal to login on an Alpha AXP or a VAX or a DECstation.

9. a window graphics terminal under DECwindows/Motif connected directly to a DECstation with ULTRIX or via a Xwindow terminal remotely connected to a DECstation with ULTRIX via Ethernet.

10. a window graphics terminal under AIX or HP-UX connected directly to an IBM RS/6000 or HP 9000 workstation or via a Xwindow terminal remotely connected to such a workstation via Ethernet.

11. an IBM 327x or IBM 5080 compatible terminal connected directly to the IBM mainframe.
12. remote login via TCP/IP Internet with Telnet on OpenVMS, ULTRIX, AIX, HP-UX, or MVS or any IBM PC or Apple Macintosh running TCP/IP software.
13. remote login via X-25 (DATEX-P, WIN) on OpenVMS or MVS.
14. remote login via HEPnet (DECnet) on OpenVMS.
15. remote telephone modem login on OpenVMS or MVS.
16. using TCP/IP Internet (Telnet) or HEPnet (DECnet) or X-25 (DATEX-P, WIN) to login on any computer system outside GSI running Telnet or X-25 if you have an account on such a system. You can use any OpenVMS, ULTRIX, AIX, HP-UX or MVS session, any IBM PC or Apple Macintosh running TCP/IP software to do so or you may use a text terminal connected to a DECserver-300 or -700.

Different types of terminals are handled in a different manner, some of them will be described in the following sections.

2.4 Logging In

2.4.1 The Alpha AXP or VAX Text Terminal

In this context a text terminal is any terminal out of the following types:

- all text terminals compatible to the VTx20 series (e.g. VT220, VT320, VT330, VT340, VT420, VT520, PECAD, or Pericom),
- the emulations under DECwindows/Motif (DECterm) and VWS,
- a window from a Xwindow terminal with LAT or TCP/IP,
- a connection under Pathworks for DOS and Mac,
- a remote login via Telnet (TCP/IP Internet),
- a remote login via X-25,
- a remote login via HEPnet (DECnet),
- or a remote login via telephone modems.

They are all using the same VTx20 keyboard layout (for PCs, MACs, and some Xwindow terminals with keyboard simulations). The usage of this Alpha AXP or VAX text terminal keypad (VTx20 compatible) is described in the appendix A on page 99.

The following description shows the OpenVMS login procedure for different terminal types. Normally, only the first phase of the OpenVMS login differs between the different terminal types.

1. A text terminal connected to a Terminal Server.
   After pressing Return, *one* of the following is possible:

   (a) The `S200xx Local>` prompt is displayed and you can proceed.

   (b) This is displayed:

   ```
   DECserver 200 Terminal Server V3.1 (BL37) - LAT V5.1
   LTA421-LTA428
   Please type HELP if you need assistance
   Enter username>
   ```

   After entering your name, you can proceed. (Note: Please, enter only your "real" name or your username.)

   (c) If your terminal has a multisession option (VT330, VT340, VT420, VT520) and the terminal and the Terminal Server port is enabled to use this option there will be the following highlighted prompt in the lowest line on your terminal

   ```
   Service Name:
   ```

   You may now enter a desired service (see later) or just Return which will produce the prompt

   ```
   S200xx Local>
   ```

   At this point you may enter

   ```
   S200xx Local> HELP
   ```

   showing the available commands. Of the possible commands only one is of importance at this time, namely:

   ```
   S200xx Local> SHOW SERVICES
   ```

   This shows the services available. In the following example only some of the offered services are shown:

   ```
   Service Name     Status       Identification
   ...
   AXP601        Available    A X P 6 0 1   Digital 2100 Server Model A500MP
   AXP602        Available    A X P 6 0 2   Digital 2100 Server Model A500MP
   AXP610        Available    A X P 6 1 0   DEC 3000 Model 400
   AXP611        Available    A X P 6 1 1   DEC 3000 Model 300
   ```

```
...
CLEX1          Available    The GSI Online Cluster 1
...
DSAA           Available    ULTRIX 4.4 (RISC)
DSAB           Available    ULTRIX 4.4 (RISC)
...
GOOSY          Available    The GSI Online Cluster 1
...
IBM            Available    IBM-DECserver Terminal Lines
...
MVIIC          Available    O T T O       VAXstation II
MVIID          Available    P A U L A
MVIIE          Available    M V I I D     MicroVAX II
MVIIG          Available    M V I I G     VAXstation 3600 Series
...
V6000A         Available    V 6 0 0 0 A   VAX 6000-340
...
VME1           Available    VME El/Ex-Lab S200CC Port 1
VME2           Available    VME El/Ex-Lab S200CC Port 2
VME3           Available    VME El/Ex-Lab S200CC Port 3
...
VSAA           Available    W S A A       VAXstation 2000
VSAB           Available    W S A B       VAXstation 3100/GPX
VSAC           Available    W S A C       VAXstation 3100/GPX
...
VSCG           Available    W S C G       VAXstation 4000-60
VSCH           Available    W S C H       VAXstation 4000-60

S200xx Local>
```

If you enter the GOOSY service, you will be connected to a GSI Online/Offline System VAX, i.e. either V6000A (FRITZ) or VSCN, whichever has more CPU time available.

If your have been logged in already on a OpenVMS system you may simulate such a Terminal Server command by using the command

```
$ SSERVICE
```

to get a list of all services available.

You can enter an available (Status: `Available`) service by typing:

```
S200xx Local> CONNECT service
or just
S200xx Local> C service
```

or from a current OpenVMS session such a Terminal Server connection can be simulated by using the command

```
$ CON service
or
$ SET HOST/LAT service
```

`service` should be replaced by the available service name. For example: To enter AXP601, the ALPHA AXP node AXP601, you would type:

```
S200xx Local> CONNECT AXP601
or just
S200xx Local> C AXP601
```

Now your text terminal is connected to the desired Alpha AXP. Proceed to login.

2. If your text terminal is connected directly by hardware to an Alpha AXP or a VAX computer at GSI proceed to login.

3. If you are using a PC with MicroSoft Windows start the eXcursion Control Panel and from that the Applications icon, and from that the OpenVMS Terminal application, the Host name (e.g. V6000A) and then click Run. You will be prompted for the OpenVMS account user name and its password. Please, type in both in capital letters and go from the Username input field to the Password field not by the `Return` key but by the `Tab` key. or by a mouse click to the Password field. Otherwise your OpenVMS account might be disabled (if so, please contact the OpenVMS Advisory Service). Then proceed to login by clicking the OK field. For configuration help contact the OpenVMS Advisory Service (for OpenVMS see names on page 5).

4. If you are using a text terminal simulated on a Apple Macintosh the network components must have been installed and configured. For help contact the Advisory Service (for OpenVMS see names on page 5). Select the menus Apple → MacTerminal Folder → MacTerminal. Set your default terminal and communication setups with the menus Setting → Terminal... or Setting → Connections... → LAT Tool or → CTERM Tool (DECnet). Then open the terminal using the menu Session → Open Connection. Then proceed to login.

5. If you want to login from another OpenVMS session using DECnet within the GSI use the command

```
$ SET HOST node
or just
$ HOST node
e.g.
$ HOST AXP601
```

and proceed to login.

6. If you login from remote via Telnet (TCP/IP Internet) you need the Internet address or name of the GSI computer you would like to connect to. The GSI naming convention is: `node.gsi.de`, e.g. `axp601.gsi.de`, `v6000a.gsi.de` or `mvs.gsi.de` for the IBM mainframe. Not all computers at GSI are reachable from remote. You can get the valid Internet addresses using the DCL command `$ UCX SHOW HOST name`. The valid Internet address should be typed out. To get detailed information and the allowance for remote login please contact the Advisory Service (for OpenVMS see names on page 5). You connect to a remote host via Telnet with the DCL command

```
$ TELNET name
or
$ TELNET
TELNET>OPEN name
or
TELNET>CONNECT name
e.g.
$ TELNET "vscn.gsi.de"
or
$ TELNET
TELNET>OPEN "vscn.gsi.de"
```

7. If you login from remote via X-25 (DATEX-P, WIN) you need the X-25 address of the GSI computer you would like to connect to. Only the VSCN and the IBM mainframe are reachable from remote. To get detailed information and the allowance for remote login please contact the Advisory Service (for OpenVMS see names on page 5).

8. If you login from remote via a telephone modem you need the telephone number of the GSI computer modem and your own telephone number must be registered for the automatic call back facility. To get detailed information and the allowance for remote login please contact the Advisory Service (for OpenVMS see names on page 5).

9. If you login from remote via HEPnet (DECnet) you need the DECnet address or name of the Alpha AXP or VAX computer you would like to connect to. Not all computers at GSI are reachable from remote. To get detailed information and the allowance for remote login please contact the OpenVMS Advisory Service (see names on page 5).

Now you can login on the Alpha AXP or VAX system by entering your username.

```
AXP601 (Digital 2100 Server Model A500MP, OpenVMS-V6.1) Please login

Username: user
```

Type in the username you received from the OpenVMS system manager when you got your account

followed by the password belonging to your username

```
Password: password

   Welcome to OpenVMS AXP (TM) Operating System, Version V6.1 on node AXP601
    Last interactive login on Wednesday, 10-AUG-1994 20:38:43.48
    Last non-interactive login on Friday, 30-AUG-1991 17:25:28.40

 System up since 26-JUL-1994 08:22:21.00

 Hello <user> is AXP601, have a nice day !
 You logged in at 18:43:19 on Thursday, August 11, 1994
 You are on terminal AXP601$RTA5.

There are no news for GSI
News for the following facilities are available:
CERNLIB
GOOSY
GSI
UPDATE
AXP601 $
```

You are now connected, and the DCL (Digital Communication Language) Prompt (`AXP601 $`) is displayed. That means, that you are presently on the DCL Command Level. In this manual the prefix defining the Alpha AXP or VAX you are logged in (like `AXP601 $` in the above example) will be neglected, i.e. only `$` is used in this manual for the DCL prompt. You may change the DCL prompt to any string you like by the DCL command `$ SET PROMPT=string`. The terminal name differs depending on the way you logged in (LTAxxxx for LAT, TNxxxx for Telnet, and so forth). Now proceed to 'First Steps under OpenVMS' on page 21.

2.4.2 The Xwindow Terminal

In this context a Xwindow terminal is any terminal out of the following types:

- DEC VXT-2000 terminal with a DEC VT220 like keyboard

- Tektronix TekXpress terminal with a DEC VT220 like keyboard
- Tektronix TekXpress terminal with an IBM-PC like keyboard

In the case of a DEC VXT-20000 you may connect to a host as a TCP/IP Xwindow, a LAT Xwindow, a TCP/IP terminal, or a LAT terminal session. In case of a Xwindow session to a VAX or Alpha AXP the Motif session manager and the Motif windows manager will be started automatically.

In case of a Tektronix TekXpress terminal you may connect to a VAX or Alpha AXP as a TCP/IP terminal (e.g. `OPEN AXP601`) or a LAT terminal session. In these cases you must start the Motif session manager and the Motif windows manager explicitly with the DCL command

```
$ XSESSION name
e.g.
$ XSESSION XWTAD
or
$ XSESSION 140.181.96.13
```

where 'name' stands for the terminal your are working on. It might be its name or the Internet address. The command XSESSION will connect to this terminal for opening the display windows.

Only in case of a Tektronix TekXpress terminal with an IBM-PC like keyboard not all keys are mapped corresponding to a DEC VT220 like keyboard. This makes problems mainly for the editors. Therefore specific calls for the two main editors are available

```
$ XEDT file
or
$ XLSE file
```

With these specifically initialized editors the non-functional keypad keys PF1 to PF4 are simulated by the keys F9 to F12 of the IBM-PC like keyboard.

2.4.3 The DECwindows/Motif Terminal

A DECwindows/Motif session can be started on each Alpha AXP or VAXstation graphics terminal connected directly to an Alpha AXP or a VAXstation, from a Xwindow terminal connected via Ethernet or from a Xwindow terminal emulation on a PC or Mac running Pathworks. Only for directly connected graphics terminals the login prompt with the Digital logo is seen on your screen. In all other cases you must connect your Xwindow terminal or the emulation to an Alpha AXP or a VAX getting this logo after the successful connection. If you login via `$ SET HOST 0` (i.e. on the same DECnet node) no DECwindows application can be started directly.

Type in your username and your password each followed by a Return.

The following is a brief description of the handling and the set-up of the window system DECwindows/Motif:

When you have more than one window open, you must make the window active you want to work with. To do so, point to a location in the window or window frame by moving the mouse and click the left mouse button (MB1). The window moves to the front of the screen and the window frame is highlighted.

If one window partially obscures another, you might want to arrange them so that each is visible. To move a window, position the pointer anywhere in the window's title bar (except on a button in this field), press and hold the left mouse button (MB1), and drag the window outline to the new location.

You can change the size of your windows to suit your needs by using the window's resize borders. To change the size of a window, position the pointer on one of the window's resize borders. The pointer changes into a resize cursor. Press and hold MB1 and drag the resize cursor to the size you want.

If you have several applications running at the same time, you can free up space on your screen by minimizing a window (shrink to an icon). All processes continue to execute while the application window is an icon. To minimize a window, point to the window's minimize button (the left button on the upper right corner of the window) and click MB1. To restore an icon to a window, point to the icon and double click MB1.

The window menu contains menu items for working with windows. To display the window menu, click on the Window Menu button (button on upper left side of a window).

After starting a session, you use the Session Manager to manage your session and your workstation environment. When you start a session, DECwindows/Motif displays the Session Manager's menu bar. You can use the Session Manager's Options menu to customize your environment. Select the Options field with the pointer and click MB1. You will see a menu of options including

Automatic Startup...
(to select applications starting automatically during login to Motif.
The Window Manager must be selected in the Automatic Startup.),
Window... (window layout),
Menus... (selection of applications available in the menu),
Menu Bar... (layout of the Session Manager Menu Bar),
Pause Screen...,
End Session Prompts...,

Screen Background... and Window Colors...,
Keyboard.. (with keyclick setup in it),
Language... (select English or German e.g. for DECwrite or DECdecision),
Pointer...,
Security... (to allow other users the opening of windows on your screen).

You can select any option with the pointer and a click with MB1. After setting your environment you can save these settings with the option 'Save Session Manager'.

To put your current session on hold, choose 'Pause' from the Session Manager's Session menu. To end your session, choose 'End Session' from the Session Manager's Session menu.

Standard applications include DECterm, Bookreader, Calendar, Calculator, Mail, Paint, or FileView. Short descriptions are given in appendix F on page 119. Details can be found in Help or with the Bookreader (see the chapter 3 on page 27).

The DECterm option simulates a VT330 (black/white) or a VT340 (color) terminal. Each selected DECterm opens a window on the screen and runs the user's login procedure automatically without asking again for a username or password.

2.4.4 First Steps under OpenVMS

You must change your password immediately after the very first login by the command:

```
$ SET PASSWORD
```

If you are using a PECAD terminal the first time give the command

```
$ PEVAX
```

to set up the PECAD terminal correctly. With the command this setup is store permanently.

Now you can get information about the directory of your private files by entering

```
$ DIRECTORY /DATE/SIZE=ALL
or just
$ DIR
```

Your directory is displayed as e.g.:

```
Directory GSI$ROOT:[user]

LOGIN.COM;1               2/3       11-JUN-1992 18:25

Total of 1 files, 2/3 blocks.
```

The directory displays the following information:

- The name of files; in the above example it is `LOGIN`
- The type extension of files; above it is `COM`
- The version number of files; above it is `1`. The version number is automatically upgraded each time you save the file with the same name, e.g. when you edit an existing file the new, changed file will get a new, higher version number.

`LOGIN.COM` is, as its name suggests, a command procedure executed whenever you log in. You can add commands to this file. These commands are then executed whenever you log in. An example of a login file is shown in appendix B on page 103. You may get this template login file by the DCL command:

```
$ COPY/LOG GOO$EXE:USER_LOGIN.COM SYS$LOGIN:*
```

Do not forget to edit this template for your personal needs.

2.4.5 The IBM-Terminal (Ethernet) Connection

Terminals hooked to Terminal Servers on the Ethernet network or any active OpenVMS session can access the IBM via LAT.

A terminal connected to a Terminal Server you want to use with the IBM must respond after pressing Return in one of the following ways:

1. The `S200xx Local>` prompt is displayed and you can proceed.
2. This is displayed:

   ```
   DECserver 200 Terminal Server V3.1 (BL37) - LAT V5.1
   LTA421-LTA428
   Please type HELP if you need assistance
   Enter username>
   ```

   (Note: Please, only enter your "real" name or your username.) After entering your name the prompt `S200xx Local>` will appear and you can proceed.

When you are in the `S200xx Local>` mode of a Terminal Server,

```
S200xx Local> HELP
```

will show the available commands. Of the possible commands only one is of importance at this time, namely:

```
S200xx Local> CONNECT IBM
```

If you want to connect from a running OpenVMS session give the DCL command

```
$ SET HOST/LAT IBM
or just
$ CON IBM
```

Now your terminal or session is connected to the IBM.

After your are connected to the IBM first type Ctrl G, the Master Reset of an IBM terminal line to cleanup the communication line to the IBM. Now the GSI logo with the IBM terminal device number VDnn will appear on the screen. You can login on the IBM system by typing in your account string followed by the Enter key and then after the prompt your password followed by the Enter key.

The Enter key is the command line delimiter on the IBM and **not** the Return key which just will move the cursor downwards.

After connecting to the IBM mainframe the following commands and keys are available (see also the keypad layout in figure 2.2 on page 25):

Ctrl G — Master Reset; should be used directly after CONNECT IBM.

Enter — The Enter key is the command line delimiter on the IBM and **not** the Return key which just will move the cursor downwards.

Insert Here — The Insert Here key switches from the default overstrike mode to the insert mode and back.

PF4 — The PF4 key is the **Attention** key for the IBM.

Ctrl R **or** Ctrl G — Error Reset; this should be used if the cursor hangs.

Ctrl X — Flush the input buffer

Ctrl V — Reshow the last logical screen

All terminals connected to the IBM are initialized for VT220 operation and also provide PERICOM or PECAD graphics, if wanted (not available for OpenVMS sessions connected to the IBM).

From an Alpha AXP or VAX you may also connect to the IBM MVS mainframe with the TN3270 utility which connects your VT220 like terminal via TCP/IP Telnet. The corresponding DCL command is

```
$ TN3270 mvs
```

You will get the IBM Netview Access Services panel for logging in. Please ask the Computer Center operators (room 1.250, tel. 2515) for a specific Netview Access Services account. Type in your username and password. Remember: the Enter key is the command line delimiter on the IBM and **not** the Return key which just will move the cursor downwards. After entering your correct username and password you will get to the Application Selection input panel. After selecting `TSOPASS` you will be ask `ENTER CURRENT PASSWORD FOR username-`. Enter your password again to get finally logged in to a MVS/TSO session. After `LOGOFF` your terminal input will return to the Alpha AXP or VAX again.

Device Separation is supported with another graphic terminal connected to a Terminal Server. That means the separation of an alphanumeric terminal for commands and a graphic terminal on the IBM. (Notice: An IBM terminal connected from a OpenVMS session has no graphics options available!) You have to connect the graphics terminal with the same procedure as your alphanumeric terminal to the IBM. By typing Ctrl `G` on the graphics terminal after the connection you will get the IBM VDnn device number of that terminal. Do not log in on the graphics terminal to the IBM, because you want to use it with the device separation as an output device only.

Please, do not forget to free your port after you logged off from IBM (as the GSI logo with the IBM terminal device number VDnn appears). Otherwise the network still holds the connection and makes the port unavailable to other users. If your are on a terminal connected directly to a Terminal Server do this by pressing F5 (or Ctrl F5 and Return on a PECAD) to enter Terminal Server "local" mode. The `S200xx Local>` prompt should appear. Then enter

```
S200xx Local> DISCONNECT SESSION n
or just
S200xx Local> DIS SESSION n
```

`n` should be replaced with the session number used for the IBM. To check which one it is, you can enter `SHOW SESSIONS`.

If you use a PECAD terminal on Alpha AXP or VAX and IBM alternately, use the DCL commands `$ PEVAX` or `$ PEIBM` respectively when you are logged in to an Alpha AXP or a VAX to set the terminal characteristics in the right way.

If you have been connected from a running OpenVMS session disconnect from the IBM by the key Ctrl \. You will be back to your original OpenVMS session.

In the figure 2.2 on page 25 you see the IBM keypad layout.

2.5 Logging Out From the Alpha AXP or VAX

Logging out from an Alpha AXP or a VAX is as easy as logging in, you simply enter the following:

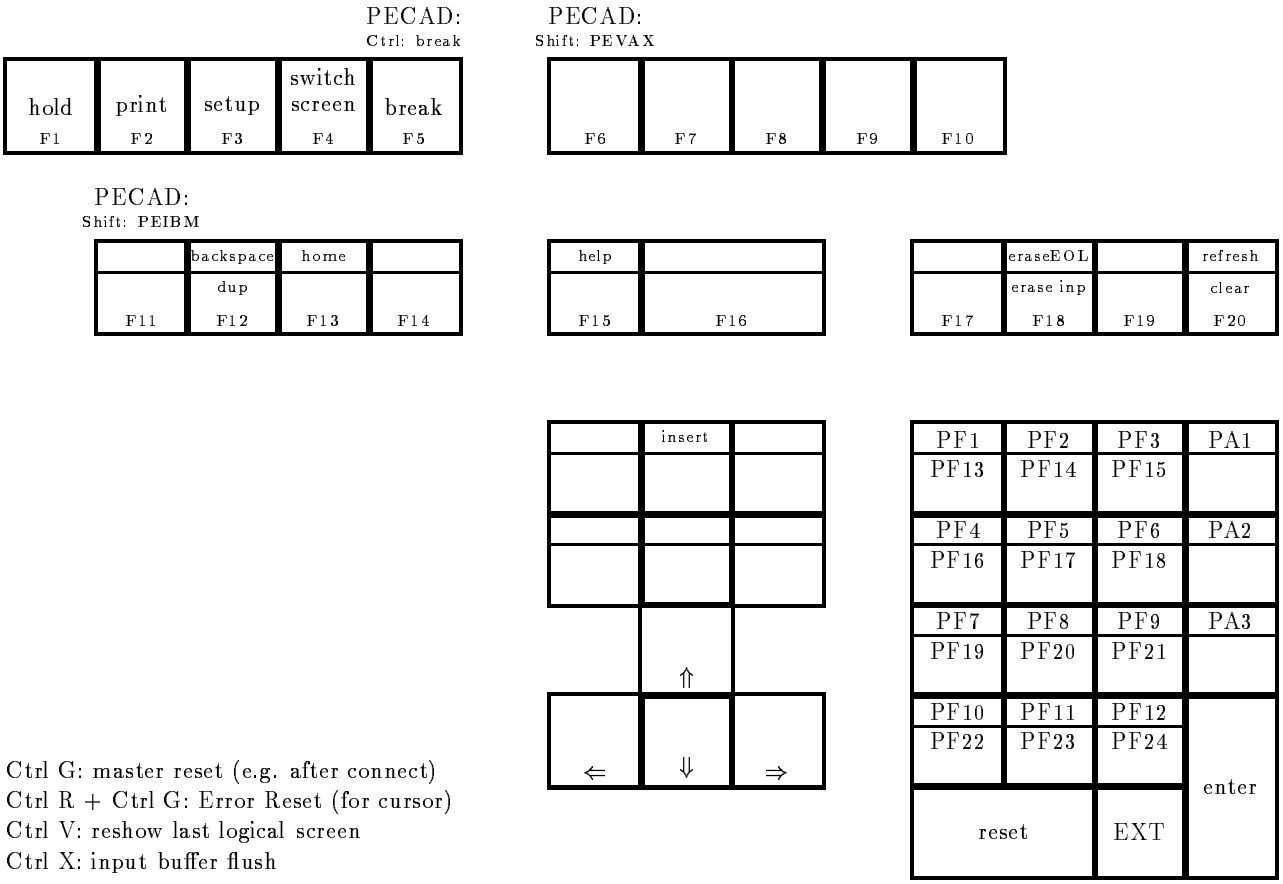


Figure 2.2: The Special Keypad Layout for IBM.

```
$ LO
```

You will be asked

```
$ PURGE SYS$LOGIN:[...]*.* ? (Y,N def: N)
```

If your answer is yes (`Y`), all the old versions of all your files are deleted, and only the highest versions are kept. This is very useful, since old versions are usually not needed anymore. If, however, you answer no (`N`) or just press `Return` the old versions will be kept. After the answer the computer displays the following:

```
Goodbye <user>, you are leaving Digital 2100 Server Model A500MP AXP601
Have a nice time
 <user>       logged out at 11-AUG-1994 19:04:45.17
```

After logging out one is again returned to the Terminal Server "local" mode or the original OpenVMS session depending on the login method. On a Terminal Server session, the prompt `S200xx Local>` appears. The `LOGOUT` command at this level logs you out of the Terminal Server and also terminates all remaining sessions.

In case of a DECwindows/Motif session you may logout each DECterm window individually or you may exit the whole session. Therefore, select within the Session Manager Session menu the End Session option. You will be asked whether really to leave the whole DECwindows session or not. If you select YES, all windows are closed and the login window with the Digital logo appears on the screen. If you have used DECwindows/Motif from a DEC Xwindow terminal VXT-2000 the session termination will automatically reboot the whole terminal. This behavior seems to be obscure but it is correct.

2.6 Terminal Server Sessions

While using a terminal connected to a Terminal Server, you can have different sessions of the Terminal Server. You can have only two sessions if you use the multisession option together with a VT330, VT340, VT420, or VT520 terminal.

Once you have started the first session (by logging in), you can "break" out of your session by pressing `F5` (or `Ctrl` `F5` and `Return` on a PECAD). After the `S200xx Local>` prompt connect to any Alpha AXP or VAX or to the IBM by typing `CONNECT service` and then simply log in to the chosen service. The number of simultaneous sessions is limited by default to 4. You can have only two sessions if you use the multisession option together with a VT330, VT340, VT420, or VT520 terminal.

You can move through your established sessions by pressing `Ctrl` \, or by breaking out to the Terminal Server local mode with key `F5` (or `Ctrl` `F5` and `Return` on a PECAD) and then using the following `S200xx Local>` commands: `FORWARD` or `BACKWARD`.

If you are using the Multisession option together with a VT330, VT340, VT420, or VT520 terminal the `FORWARD` and `BACKWARD` switches are not allowed. You switch between the two possible session using the `F4` key of your keyboard. You may also split and unsplit the screen by using `Ctrl` `F4`.

Chapter 3

Getting Interactive Help

3.1 OpenVMS DCL HELP

To obtain on-line documentation for a command, enter the command HELP with the name of the command as a parameter.

```
$ HELP ALLOCATE

ALLOCATE

     Provides your process with exclusive access to a device until
     you deallocate the device or terminate your process. Optionally
     associates a logical name with the device.

     Requires read (R), write (W), or control access.

     Format

       ALLOCATE  device-name[:][,...] [logical-name[:]]



  Additional information available:

  Parameters Qualifiers
  /GENERIC   /LOG
  Examples

ALLOCATE Subtopic?
```

If you need help, but do not know what command or system topic to specify, enter the command `HELP` with the word `HINTS` as a parameter. Each task name is listed in the `HINTS` text is associated with a list of related command names and system information topics.

```
$ HELP HINTS

HINTS

   Type the name of one of the categories listed below to obtain a list
   of related commands and topics.  To obtain detailed information on a
   topic, press the RETURN key until you reach the "Topic?" prompt and then
   type the name of the topic.

   Topics that appear in all upper case are DCL commands.

  Additional information available:

  Batch_and_print_jobs  Command_procedures    Contacting_people
  Creating_processes    Developing_programs   Executing_programs
  Files_and_directories Logical_names         Operators_in_expressions
  Physical_devices      Security              System_management
  Terminal_environment  User_environment

HINTS Subtopic?
```

When HELP prompts you for a topic or subtopic, you can enter one of the listed subtopics to obtain additional information (command and topic names can be abbreviated). Alternatively, you can press Return to move back a level, enter a question mark to redisplay the current text, or press Ctrl Z to exit.

Using wildcard characters when specifying a topic allows you to obtain various amounts of information.

`HELP command` — The command or topic and all related information.

`HELP command *` — All related information on that HELP level.

`HELP com*` — All commands or topics beginning with the specified character(s).

`HELP *` — All the commands and topics available in the HELP file.

`?` — Get the last seen help information on each HELP level

To get help information from the various HELP libraries listed at the end of the first HELP level type the following, for example:

```
$ HELP @GSIHELP GSILOG
```

The libraries of general interest are:

GSIHELP:	Several general utilities available at GSI, i.e. TEX
SYSMSGHELP:	OpenVMS system error message descriptions
KERMITSYS:	KERMIT help library
VPW:	FermiLab software like DTC and CALC
UTILITY:	GSI and GOOSY command procedures
COMMAND:	GOOSY commands
PROGRAM:	GOOSY main programs
MODULE:	GOOSY program modules
MESSAGE:	GOOSY error message descriptions
RECOVER:	GOOSY error recovery descriptions
DMTYPES:	GOOSY data element declarations

3.2 DECwindows/Motif Bookreader

If your are working directly on an Alpha AXP or a VAXstation or via a Xwindow terminal on an Alpha AXP or a VAXstation running DECwindows/Motif, and only then, a general utility is available to get all OpenVMS manuals on-line in windows on your screen.

The manuals are stored on several CD-disks mounted on a centralized DEC InfoServer connected to Ethernet only. This allows read access from all Alpha AXP and VAXstations at GSI to the same CD-disk drives.

Select from the Session Manager window Applications menu the Bookreader item. It will pop up a directory of available libraries. Select the Online Documentation Library Contents with a mouse MB1 (left mouse key) double click in that line. The sub directory appears. Select the Master Listing line with a mouse MB1 double click. The directory of all main documents appears in alphabetical order. Select the topic you are interested in, e.g. DEC FORTRAN, with a mouse MB1 double click on this line. The list of all DEC FORTRAN manuals appears from which you may select the desired one, e.g. DEC FORTRAN Language Reference Manual. Select this line with a mouse MB1 double click. A new window pops up with the Contents of this manual. Select the topic within this contents with a mouse MB1 double click. A new window pops up showing the start page of this topic. You can navigate through this topic or the whole manual using the Screen or Topic arrows on the bottom line.

Selecting in the View menu the Hotspots option by pressing and holding MB1 then moving the mouse cursor to the this line and release MB1 on the line Hotspots. Hotspots are cross references in the manual text embraced by a frame box, e.g. Figure 3-1. Select such a Hotspot with a mouse MB1 double click. A new window pops up showing the corresponding information, e.g. the Figure 3-1.

Close the text and Hotspot windows by selecting the Close option on the bottom line of the

windows.

In the manual's Contents window you may select from the View menu the list of the contents (start default), examples, figures, tables, or the index. The contents can be collapsed or expanded.

Leave the manual by selecting from the File menu the Close Book option. Exit the Bookreader by selecting from the File menu the Exit option or type Ctrl e (the keyboard Lock to upper case letters should not by active).

3.3 CNEWS

News of general interest are stored by the computer Advisory Service in a specific facility called CNEWS. This facility is available on all computer platforms at GSI as OpenVMS, AIX, HP-UX, and LynxOS. The news are ordered by the field of interest into the following topics: CERNLIB, DAQ, GOOSY, and GSI. Each article is indexed within each topic. For new, unread articles you will get informed briefly during the login procedure.

You will get a list of available topics by the command

```
$ CNEWS topic
```

You select the unread news of a specific topic by the DCL command

```
$ CNEWS topic
e.g.
$ CNEWS GSI
```

A list of unread news are written to the screen.

You can get a short overview about all news including the unseen by the DCL command

```
$ CNEWS topic /ALL    or    $ CNEWS topic -a
e.g.
$ CNEWS GSI /ALL      or    $ CNEWS GSI -a
```

You can read a specific news article selecting its index

```
$ CNEWS topic n
E.G.
$ CNEWS GSI 3
```

To mark all news indices of a specific topic as 'seen' type

```
$ CNEWS topic /SEEN   or    $ CNEWS topic -s
```

A menu is invoked for this command by typing

```
$ CNEWS ?
```

3.4 WorldWideWeb WWW

The WorldWideWeb (WWW or W3) is the universe of network-accessible information, an embodiment of human knowledge. It is an initiative started at CERN , now with many participants. It has a body of software, and a set of protocols and conventions. W3 uses hypertext and multimedia techniques to make the web easy for anyone to roam, browse, and contribute to. Future evolution of W3 is coordinated by the W3 Organization. The World Wide Web is the vision of programs that can understand the numerous different information-retrieval protocols (FTP, Telnet, NNTP, WAIS, gopher, ...) in use on the Internet today as well as the data formats of those protocols (ASCII, GIF, PostScript, DVI, TeXinfo, ...) and provide a single consistent user-interface to them all. In addition, these programs would understand a new protocol (HTTP) and a new data format (HTML) both geared toward hypermedia.

Documents on the Web are referred to using URLs (Uniform Resource Locators). An URL looks like http://www.vuw.ac.nz/campus/home.html. It consists of three parts the method of retrieving the document (http), an option machine name (www.vuw.ac.nz) and a pathname (/campus/home.html). The URL format is nearly an Internet standard. Think of the so called Uniform Resource Locator (URL) as a networked extension of the standard filename concept: not only can you point to a file in a directory, but that file and that directory can exist on any machine on the network, can be served via any of several different methods, and might not even be something as simple as a file: URLs can also point to queries, documents stored deep within databases, the results of a `finger` or `archie` command, or whatever.

The GSI home page is accessible via the URL "http://www.gsi.de/gsi.html". You will find a lot of information like this manual in the GSI WWW pages.

To start WWW on OpenVMS you must call `PUBLICLOGIN` first, e.g. within your `LOGIN.COM` procedure (see or copy example `GOO$EXE:USER_LOGIN.COM`). Call the WWW on OpenVMS if you are running a Xwindow (Motif) session on a workstation or a Xwindow terminal by typing

```
$ XWWW.
```

3.5 Interactive Training Courses

There is a general interactive training course available for VAX OpenVMS users on a text terminal or compatible DECterm window. Enter the interactive training course from any running OpenVMS session (on a VAX from the CI cluster, only, not on Alpha AXP) using the DCL command

```
$ STUDENT
```

You will be guided through the course by a menu. Please, follow the instructions strictly, i.e. read all comments presented by the guide.

Chapter 4

Using Communication Utilities

4.1 Using the OpenVMS Mail Utility

The interactive Mail utility (MAIL) allows you to send and receive messages, as well as to file, forward, delete, and reply to messages that you have received. To invoke the interactive Mail Utility from a text terminal, specify the DCL command MAIL without parameters.

```
$ MAIL
MAIL>
```

You may also call the Mail utility together with DECwindows/Motif selecting from the Session Manager's Applications menu Mail. A new window pops up which you may tailor to your needs using the Options menu. After selecting the desired layout save these settings with the Save Settings option from the Options menu. Only then this layout will be reproduced after each further start of Mail.

You can display information on your text terminal about MAIL commands by entering HELP in response to the MAIL> prompt or by selecting the Help menu under DECwindows/Motif. To exit from MAIL on a text terminal enter the MAIL command EXIT or press Ctrl Z. To exit from Mail under DECwindows/Motif select Exit from the File menu or type Ctrl e.

In the following only the text terminal interactive Mail will be described. The DECwindows/Motif Mail is similar to use, only buttons, instead of commands, must be selected popping up new specific windows.

4.1.1 Mail File Directory

Your mail file, MAIL.MAI containing all information about the mails you ever received and kept is by default created in your default login directory the very first time you receive a mail message. To get a better structure of your file system the very first time your are using the OpenVMS

Mail utility you should create a Mail subdirectory in your default file system. You do this by the command

```
$ MAIL
MAIL>SET MAIL_DIRECTORY
MAIL>SHOW MAIL_DIRECTORY
```

4.1.2 Sending Mail

You can create and send mail messages interactively with the Mail Utility. You can send files to other users within the same Alpha AXP or VAX Cluster, to other DECnet Alpha AXP or VAX nodes at GSI, or to HEPnet, Internet, X-25 (DATEX-P /WIN), and indirectly to BitNet (EARN) (for these and other networks see also the section 'CMAIL Utility' on page 40) from within the Mail Utility or from the DCL command level.

After invoking the Mail Utility, specify the `SEND` command to create and send a mail message. `MAIL` prompts you for the names of the users to whom you want to send the message, the subject of the message (optional), and the text of the message (optional). The following example sends a message to a user named Anne on a node of the same VMScluster:

```
MAIL> SEND
To:   Anne
Subj: Meeting of June 9
Enter your message below. Press CTRL/Z when complete, or CTRL/C to quit:
Sorry I cannot make the meeting:  I'll be on vacation
during that week. Let me know how it goes.
                    Joe
```

Note that pressing Ctrl Z actually sends the message.

If the user has no account on the same Alpha AXP or VAX VMScluster you are currently logged in you have to specify the node where to find the user. There are four possibilities:

1. DECnet or HEPnet node:
   In the case of DECnet or HEPnet specify the addressee as `node::user`, e.g. node AXP601

   ```
   MAIL> SEND
   To:   AXP601::Anne
   Subj: Meeting of June 9
   etc.
   ```

2. Internet node:
   In the case of Internet specify the addressee as `SMTP%"user@internet-address"`, e.g. node MVS (IBM mainframe) at GSI

```
MAIL> SEND
To:   SMTP%"Anne@mvs.gsi.de"
Subj: Meeting of June 9
etc.
```

The prefix `SMTP%` and the double quotes must be given exactly in this spelling.

3. BitNet or EARN node:
   In the case of BitNet or EARN specify the addressee as `SMTP%"user%node.BITNET@gsi.de"`, e.g. node IBM mainframe at GSI = DDAGSI3

```
MAIL> SEND
To:    SMTP%"PR99%DDAGSI3.BITNET@gsi.de"
Subj: Meeting of June 9
etc.
```

   The prefix `SMTP%` and the double quotes must be given exactly in this spelling.

4. X-25 (DATEX-P/WIN) node:
   In the case of X-25 (DATEX-P or WIN) specify the addressee as `VSCN::PSI%DATEXP.number::user`, e.g.

```
MAIL> SEND
To:   VSCN::PSI%DATEXP.1234567890::Anne
Subj: Meeting of June 9
etc.
```

   The prefix `VSCN::PSI%` must be given exactly in this spelling without double quotes.

If you decide not to send the message, enter `Ctrl` `C`, which cancels the `SEND` operation without exiting from `MAIL`.

If you want to send a text file use the MAIL command

```
MAIL> SEND file.type
To:   AXP601::Anne
Subj: Meeting of June 9
```

where `file.type` is the text file specification. In this case you cannot enter an additional message text. Only the contents of `file.type` and the subject text will be sent.

You may also use the DCL command line

```
$ MAIL file.type "AXP601::ANNE" /SUBJECT="Meeting of June 9" /EDIT /SELF
```

where `file.type` is the text file specification and `/EDIT` starts the text editor first. The `/SELF` option will send a copy of the mail to your own account.

You may set up a forwarding address permanently within Mail by the Mail command `SET FORWARD`, e.g.

```
MAIL> SET FORWARD Anne
or
MAIL> SET FORWARD "AXP601::Anne"
```

In case of Internet or BitNet the special handling of double quotes must be taken into account, e.g.

```
MAIL> SET FORWARD "SMTP%""Anne@mvs.gsi.de"""
or
MAIL> SET FORWARD "SMTP%""PR99%DDAGSI3@gsi.de"""
or
MAIL> SET FORWARD "VSCN::PSI%DATEXP.1234567890::ANNE"
```

4.1.3 Mail Addressing of GSI from Remote

If you want to receive electronic mail from remote computers there are three different possibilities.

- EARN/BitNet from remote:
  The only GSI computer available via EARN/BitNet is the IBM mainframe (address DDAGSI3). The VAX (former address DDAGIS5) are not longer connected to EARN. A correct mailing address of a GSI user from a remote EARN/BitNet computer is

  ```
  account@DDAGSI3          for IBM mainframe
  e.g.
  PR99@DDAGSI3
  ```

  The account or the username must be valid on the IBM, respectively.

- Internet TCP/IP:
  Be aware of case sensitivity of user and node name in UNIX. Therefore it is mostly necessary to set addresses in double quotes "address".
  Any user of the Alpha AXP VMScluster can be reached via the AXP601 or AXP602, any user of the VAX CI-cluster can be reached via the V6000A VAX, and any user of the accelerator VAX VMScluster can be reached via the ALICE VAX from any remote Internet computer with the following address

  ```
  F.Name@gsi.de
  or
  ```

```
username@axp601.gsi.de
or
username@v6000a.gsi.de
or
username@alice.gsi.de
e.g.
anne@v6000a.gsi.de
```

The first addressing method is the common GSI addressing with "F" is the initial of the first name and "Name" is the surname of the user, e.g. for Mathias Richter it would be: "M.Richter@gsi.de". The DVEE department keeps a translation list of all users at GSI with their preferred destinations. Therefore e.g. the address "M.Richter@gsi.de" will be translated automatically into "goori@vsae.gsi.de". This method is easy for outside users since they have not to care about usernames and node names.

The IBM mainframe and all UNIX workstations are also available with their Internet addresses

```
account@mvs.gsi.de      for the IBM mainframe
or
username@rzri6a.gsi.de  for the RZRI6A IBM RS/6000 workstation
e.g.
pr99@mvs.gsi.de
anne@rzri6a.gsi.de
```

Again, the account of the username must be valid on the computer.

Alpha AXPs or VAXs outside the CI-cluster are reachable indirectly via their DECnet address

```
"node::username"@v6000a.gsi.de
e.g.
"vsab::anne"@v6000a.gsi.de
```

Again, the account of the username must be valid on the computer "node".

- X-25, DATEX-P, WIN (PSI)
  The only GSI computers available via X-25 are the IBM mainframe and the VSCN VAX. The addresses are available on request from the OpenVMS Advisory Service (see names on page 5). A correct mailing address of a GSI user from a remote Alpha AXP or VAX system is

```
PSI%DATEXP.1234567890::username
```

where "1234567890" is to be replaced by the GSI X-25 number you may get on request from the OpenVMS Advisory Service (see names on page 5). and where "username" is a valid user account on VSCN.

4.1.4 Reading Mail

Invoke the Interactive Mail Utility to read a mail message. Messages that you receive are stored in mail files, which have a default file type of `.MAI`. Your default mail file, `MAIL.MAI` is created in your default directory the first time you receive a mail message (see also Section **??** on page **??**).

When you are logged in and receive a mail message, notice of the new message appears on your screen. For example, a message sent by a user named Jim would appear as:

```
New mail from Jim
```

You are also notified that you have new mail when you log in and when you invoke `MAIL`. To read a new mail message, invoke `MAIL` interactively; `MAIL` prompts for a command and, if you have received mail, displays the number of mail messages you have received.

```
$ MAIL

You have 1 new message.

MAIL>
```

To read the new message, just press `Return` and the message appears on your screen.

```
#1            1-JUN-1994  14:12:27     NEWMAIL

From: Jim
To:   Joe
Subj: Meeting of June 9

When is your vacation planned?
I'm planning to take off at the same time.

MAIL>
```

If there is no mail although the mail facility claimed one or several mails to be there use the command

```
MAIL> READ /NEW
```

to get rid of the fake.

To continue reading your new mail messages, press Return in response to the MAIL> prompt. Pressing Return in MAIL is equivalent to specifying the READ command without parameters.

To read a specific Mail use the Mail command

```
MAIL> READ n
e.g.
MAIL> READ 12
```

where n is the number of the Mail seen by the Mail command DIRECTORY.

When you have read all new messages, MAIL issues the message "%MAIL-E-NOMOREMSG, no more messages," and continues to prompt for commands until you exit by entering EXIT or by pressing Ctrl Z.

If you receive a mail message while you are in MAIL, specify the READ/NEW command to read the new message.

To delete a message you must read the message first and then type

```
MAIL> DELETE    or just    D     or     D i-j,k
```

where i-j deletes all mails with numbers i up to j inclusively and the number k in addition. (For more information about the MAIL Utility see the OpenVMS User's Manual or type HELP within Mail.)

4.1.5 Organizing Your Mails

To get a listing of all mails you received already just type the command

```
MAIL> DIRECTORY    or just      DIR
```

To store received mails under different topics you may move them into so called folders just after you read the mail by the command

```
MAIL> MOVE folder-name
or
MAIL> MOVE folder-name /ALL      for moving all mails from one folder to another
```

The folder will automatically be created if not yet existing. You can select a folder by the command

```
MAIL> SELECT folder-name
or
MAIL> SELECT folder-name /FROM="string"       mails containing string
```

```
or
MAIL> SELECT folder-name /SUBJECT="string"    subjects of mails containing string
MAIL> DIRECTORY   or just   DIR    shows the selected list
```

to read mails from that folder. The `DIR` command will list all mails from the selected folder. You get all existing folder names by the command

```
MAIL> DIR/FOLDERS
```

If you want to get the contents of a mail into a file use the command

```
MAIL> EXTRACT file-name
```

just after you read the mail.

To print a mail on a laser printer you first have to extract the mail into a file by `EXTRACT file-name` just after you read the mail, then leave the `MAIL` utility by `EXIT` or just Ctrl Z, and print the file by `POP -P08WAS file-name` depending on the laser printer location (see Appendix D on page 109).

4.1.6 CMAIL Utility

It is invoked by `CMAIL` and it allows you to send a text file as a mail to external users, to a remote network. One may specify network addresses in the form:

```
INTER::<node>::<user>          Internet
EARN::<node>::<user>           EARN/BitNet
DFN::<node>::<user>            DFN (EAN), Internet and all special networks
ARPA::[<net>::]<node>::<user>  ARPA net
FI::[<net>::]<node>::<user>    Finland Internet
AU::[<net>::]<node>::<user>    AU (Australian Universities)
JANET::<node>::<user>          JANET (GB)
HEP::<node>::<user>            High Energy Physics DECnet
PSI::<node>::<user>            VAX-PSI (X-25)=DATEX-P (WIN)
```

For example, to send the file `EXAMPLES.TXT` to the user Frank at node LBL in the BitNet network, enter:

```
$ CMAIL EXAMPLES.TXT EARN::LBL::FRANK
```

or to create and to edit the file `TEXT.TXT` and then to send it to the user Miller on node VAX.HMI.DBP.DE in the German DFN net, enter:

```
$ CMAIL TEXT.TXT DFN::VAX.HMI.DBP.DE::MILLER /EDIT /SUBJECT="Any text"
```

or to send the file `TEXT.TXT` to the user Miller on node DSAA.GSI.DE in the Internet, enter:

```
$ CMAIL TEXT.TXT DFN::DSAA.GSI.DE::MILLER /SUBJECT="Any text"
```

If you would like more information about the usage of it at GSI, enter `HELP CMAIL`.

To get the nodes available in the BitNet (EARN) network use the `EARN` command followed by any string you know from this node. All nodes will be found, that include that string fully or partially. For example,

```
$ EARN ARGONNE
$ EARN ANL
```

The IBM mainframe has the BitNet (EARN) address DDAGSI3.

4.2 Using the Phone Utility

To 'talk' interactively with another user on any node at DECnet, use the `PHONE` command:

```
$ PHONE user              ! phone user on Your node
$ PHONE ANSWER            ! answer a phone call
$ PHONE node::user        ! phone user on other node
$ PHONE                   ! enter interactive PHONE
```

Then the PHONE display appears. Now the following commands are available:

```
  % DIRECTORY             ! List of current users
  % DIAL user             ! phone a user
  % ANSWER                ! answer a call
```

Ctrl Z hangs up, i.e. leaves PHONE.

The screen is splitted and the participants write simultaneous on both screens.

Chapter 5

Command Formats

The DIGITAL Command Language (DCL) provides you with a direct connection to the OpenVMS system and the software running on Alpha AXP or VAX. In response to the DCL prompt (which is initially one or two letters defining the node you are logged in and a dollar sign, e.g. `F $`), you enter a command name followed by any desired parameters and qualifiers. DCL interprets the command, and either executes it directly (a so–called "built–in" command) or calls an appropriate program to execute it, passing to that program any parameter and qualifier information.

Some DCL commands invoke utilities that themselves accept interactive subcommands. You then work interactively with the program by entering subcommands and other information in response to the utility's command prompts. You continue to work with the utility until you exit from it and return to DCL command level. The `MAIL` and `LSE` (Language Sensitive Editor) command fall into this category.

A program that is associated with a command (a command image) can be DEC or user supplied. The built–in commands and the commands that execute system programs are supplied by DEC as part of the operating system. For a complete description of the DCL command syntax turn to the *OpenVMS General User's Manual, Using DCL* and for a complete description of all commands turn to the *OpenVMS General User's Manual, DCL Dictionary.*

5.1 DCL Command Format

A DCL command follows the general format (the `[ ]` mark the optional parts of the command):

```
command [/command-qualifier...] [parameter[/parameter-qualifier...]]...
```

where `command` is the name of the command, `command-qualifier` is the name of a command qualifier, `parameter` is the name of a parameter, and `parameter-qualifier` is the name of a

parameter-qualifier.

Lowercase and uppercase characters in command and qualifier names are equivalent. Lowercase and uppercase characters in parameter and parameter-qualifier values are equivalent unless enclosed in quotation marks.

The following COPY command line consists of the command name, a command qualifier, and two parameters

```
$ COPY/LOG FORMAT.TXT WATER.TXT
```

The command instructs the system to copy the file FORMAT.TXT to another file named WATER.TXT and display (log) the status of the operation on your terminal. DCL interprets the command, the qualifiers, and parameters and calls the system program SYS$SYSTEM:COPY.EXE, passing it the qualifier and parameter information for execution.

You must observe the following rules in entering DCL commands:

Delimiters— Delimit the command name and parameters with one or more blanks, or tabs, or qualifiers. Begin each qualifier with a slash (/); the slash serves as a delimiter and need not be preceded by blanks or tabs.

Line Wrapping— You can continue a command line by terminating it with a hyphen, pressing Return, and entering more of the command on the next line (although a single command line cannot exceed 255 characters), as demonstrated:

```
$ COPY/LOG FORMAT.TXT, WATER.TXT, SOIL.TXT -
_$ SAVE.TXT
```

The system responds to the hyphen and Return with the prompt string dollar preceded by an underscore (_$). Note that the space delimiting command names and parameters must be supplied (Return is not treated as a delimiter).

Size limit— An element in a command (for example, a qualifier and associated values) must not exceed 255 characters. The number of elements in a command must not exceed 128. The entire command must not exceed 1024 characters after all symbols and lexical functions are converted to their values.

Abbreviations— You can abbreviate a command name by truncating it if the abbreviated name is still unique among all the DCL command names. You can abbreviate a qualifier name if it remains unique among all qualifier names for the same command. (For clarity, not all examples in this manual abbreviate commands or qualifiers.)

All command and qualifier names are unique within four characters only (not

counting the slash before qualifiers). Only if you have defined a DCL symbol which you use as a command synonym the whole command will be checked (see section 5.6 at page 48). The following commands, for example, are equivalent:

```
$ SHOW TIME
$ SH TI
```

In interactive mode, you will work faster if you abbreviate. The abbreviations may feel awkward at first but you will soon get used to them. You should not abbreviate commands within command procedures because: (1) your command procedure will be difficult to read, and (2) the abbreviations might not be valid after new DCL commands are added at a later date.

5.2 DCL Parameters

A parameter consists of a value or a list of values. You must position it in a specified order within the command, as demonstrated:

```
$ COPY WATER.TXT  FORMAT.TXT
```

This command causes the file `WATER.TXT` to be copied to `FORMAT.TXT`. The following command reverses the order of the parameters, copying the file `FORMAT.TXT` to `WATER.TXT`

```
$ COPY FORMAT.TXT  WATER.TXT
```

Specify a parameter list by separating the values with commas (in some commands, you can use plus signs to denote concatenation of files). The following example copies a number of files into one new file.

```
$ COPY FORMAT.TXT,WATER.TXT,SOIL.TXT  SAVE.TXT
```

The entire list `FORMAT.TXT`, `WATER.TXT`, `SOIL.TXT` constitutes the first parameter. `SAVE.TXT` is the second parameter.

5.3 DCL Qualifiers

A qualifier consists of keyword, or a keyword followed by a value or list of values. The keyword starts with a slash. Three general classes of qualifiers exist:

Command qualifiers— A command qualifier applies to the entire command; the best practice is to place it after the command name (or after other command qualifiers following the command name). The following example prints two copies each of `WATER.TXT` and `SOIL.TXT`:

```
$ POP -PO8WAS -O /COPIES=2 WATER.TXT,SOIL.TXT
```

Positional qualifiers— A positional qualifier has different meanings depending on where you place it in the command string. If you place a positional qualifier after the command verb but before the first parameter, the qualifier affects the entire command string. If you place a positional qualifier after a parameter, the parameter affects only that parameter. In the following example, the first `PRINT` command requests two copies of both files whereas the second requests two copies of the first file `SPRING.SUM`, but only one copy of `FALL.SUM`.

```
$ PRINT /QUEUE=PO8WAS /COPIES=2 SPRING.SUM,FALL.SUM
$ PRINT /QUEUE=PO8WAS SPRING.SUM/COPIES=2,FALL.SUM
```

Parameter qualifiers— A parameter qualifier applies only to the parameter value it follows. The following example prints two copies of `WATER.TXT` and three copies of `SOIL.TXT`:

```
$ PRINT /QUEUE=PO8WAS WATER.TXT/COPIES=2,SOIL.TXT/COPIES=3
```

Within the confines of the above rules, the relative position of qualifiers in a command does not matter. Qualifiers take one of the following formats:

Positive–negative qualifiers— Positive–negative qualifiers have a value of true or false. You do not specify a value, but indicate a true by simply naming the qualifier, or negate the qualifier by inserting the prefix `no`. The first example that follows, notifies you when your print job has been completed or aborted, while the second example does not.

```
$ PRINT /QUEUE=PO8WAS /NOTIFY WATER.TXT,SOIL.TXT
$ PRINT /QUEUE=PO8WAS /NONOTIFY WATER.TXT,SOIL.TXT
```

Value qualifiers— If the qualifier accepts a value, you specify it by appending an equal sign and the value, as demonstrated:

```
$ PRINT /QUEUE=PO8WAS /COPIES=2 WATER.TXT
```

The `/COPIES` qualifier has a value of 2.

Lists of values for qualifiers— If the qualifier accepts a list of values, you must enclose the values in parentheses and separate them with commas, as demonstrated:

```
$ DELETE/ENTRY=(230,231) SYS$BATCH
```

The command deletes jobs 230 and 231 from the batch queue SYS$BATCH.

Value and positive–negative combinations— Some qualifiers combine positive–negative and value characteristics so that the qualifier accepts a value. The `LINK` command, for example, permits the following choices for the `/EXEC` qualifier.

```
$ LINK INFILE
$ LINK INFILE /EXEC=OUTFILE
$ LINK INFILE /NOEXEC
```

In the first example the object file named `INFILE` with the default file type `.OBJ` will be linked and an executable image will be produced located in the file `INFILE.EXE`. In the second example the resulting executable image will be in the output file `OUTFILE.EXE`. In the third example no executable image will be produced, i.e. this is only a formal check of the link step.

Defaults— Most of the qualifiers take defaults or do not affect existing values when they are not specified. For example, the following commands are equivalent because the qualifiers `/KEEP=1` and `/NOLOG` are the defaults for the `PURGE` command.

```
$ PURGE [.MEMOS]
$ PURGE [.MEMOS]/KEEP=1/NOLOG
```

The following command affects only the number of characters per line of your terminal screen. All the other terminal characteristics remain the same.

```
$ SET TERMINAL/WIDTH=132
```

5.4 DCL Comment Lines

Comments in a DCL command line are separated by an exclamation mark `!`. It can be either at the beginning of a line or somewhere in between

```
$ ! The whole line is just a comment line

$ SHOW TIME   ! only the text behind the exclamation mark is a comment
```

5.5 DCL Prompting for Parameters

If you omit a required parameter from a command, the DCL prompts you for that parameter, and for any additional parameters, as demonstrated:

```
$ ALLOCATE
_Device: DL
_Log_Name: ACCOUNTS_DISK
```

You can omit parameters by just pressing Return. On any prompt, you can enter one or more of the remaining parameters and any additional qualifiers. You can terminate the terminal input request by entering Ctrl Z.

5.6 DCL Symbols

DCL symbols are very often used as abbreviations for DCL commands and to call DCL procedures or programs.

There are **global** and **local** symbols. Local symbols are known only at the command procedure level where they are created, whereas global symbols are known to all levels. Global symbols are deleted only by `DELETE/SYMBOL/GLOBAL` or by logout. You should use mainly local symbols interactively and in DCL procedures. A symbol as first word in a command line is replaced by its string value. An asterisk in the symbol creation marks the abbreviation length.

Symbols are created (deleted) by

```
$ symbol = "string"    ! local symbol defined only for the current command procedure
$ symbol == "string"   ! global symbol defined for all following command procedures
$ ALPHA == "DIRECTORY [ALPHA]"  ! you can use ALPHA as a new command
$ GA*MMA == "DIRECTORY [GAMMA]" ! you can abbreviate the new command GAMMA to GA
$ DELETE/SYMBOL symbol          ! Delete a local symbol
$ DELETE/SYMBOL/GLOBAL symbol   ! Delete a global symbol
```

Lots of examples for symbols are shown by

```
$ SHOW SYMBOL/GLOBAL/ALL ! show all global symbols defined
or
$ SSYM                   ! which is a symbol itself
$ SSYM SSYM              ! shows what SSYM really is
$ SSYM M*                ! shows all symbols beginning with M
$ SSYM * /SEARCH=GOO$EXE ! shows all symbols with a GOO$EXE in the replacement
```

Because symbols are deleted, if you logout, create those in your LOGIN.COM procedure which you would like to use normally.

Most often symbols are used for individual command shorthands. But they can also be used inside command lines. Then they must be marked by single quotes ' when used as replacements:

```
$ MR == "GOORI"
$ PHONE 'MR'
```

Now the `PHONE` command rings GOORI instead of MR. More about that in the chapter 8 at page 83. Some symbols are used by certain utilities as defaults. Add your specific definitions in the LOGIN.COM file:

```
$ DEFCOMPI == "PPL"      ! default file type for COMPILE command
$ IBM_ACC == "account"   ! your IBM account
$ IBM_PW == "password"   ! your IBM password
$ TEX_OUTPUT == "LN03_A" ! default Alpha AXP or VAX laser printer for TEX
```

5.7 Interactive Commands

You invoke an interactive command by typing its name and pressing Return. (Some interactive commands accept parameters and qualifiers on the command line.) The command responds with a prompt, as shown:

```
$ MAIL
Mail>
```

You can now enter subcommands recognized by that interactive command. To enter another DCL command, you must first exit from the interactive command, usually by typing EXIT (and pressing Return) or pressing Ctrl Z in response to the command prompt.

To obtain help for an interactive command, invoke the command and then type HELP (and press Return) as you would at DCL command level. (See chapter 3)

5.8 Interrupting Commands

You can interrupt the execution of a command by pressing Ctrl Y, Ctrl C, or Ctrl T (the current program may have redefined Ctrl C, or Ctrl T in which case the usual system actions are overridden). Ctrl T interrupts execution of the command, displays a line of information (node name, process name, system time, elapsed CPU time, page faults, direct and buffered I/O operations, and pages in physical memory), and resumes execution without interfering with the execution of the program. The following example interrupts the copy operation to display Ctrl T information and then resumes the copy operation.

```
$ COPY [.MEMOS]*.* *
```

Ctrl T

```
VSCN::USER 16:54:17 COPY CPU=00:00:02.16 PF=241 IO=47 MEM=141
```

Ctrl Y interrupts a command and returns you to DCL command level without completing execution of the command.

```
$ COPY [.MEMOS]*.* *
```

Ctrl Y

```
INTERRUPT
$
```

After interrupting a command with
Ctrl Y
you can:

Continue— You can continue execution of the program by entering the command `CONTINUE`. Any number of built–in commands (but only built–in commands) can be entered after Ctrl Y and before `CONTINUE`. The following example interrupts the execution of the `CLEANUP` command procedure, sets verification, and then continues execution of the command procedure.

```
$ @CLEANUP

Ctrl Y

INTERRUPT
$ SET VERIFY
$ CONTINUE
```

STOP— To terminate a program immediately enter the DCL command `STOP`.

Ctrl C works like Ctrl Y unless the program you are executing responds explicitly to Ctrl C (which is a common practice).

Ctrl Z exits a program or a command procedure only on input request. Nevertheless, the Ctrl Z should be the main terminating control character of programs and command procedures.

In case Ctrl Y or Ctrl T do not work you may try the DCL command:

```
$ SET CONTROL=(T,Y)
```

to enable these control key again. You may also disable a control key by i.e.

```
$ SET NOCONTROL=Y
```

5.9 Processes

The general environment (do not mix up with a GOOSY environment) in which you use the OpenVMS system is called your process. A process contains identification and status information that the system needs to execute programs for you. Within a process, programs execute one at a time in the order in which they are invoked. The system also creates special system processes to perform various functions.

The DCL command

```
$ SHOW SYSTEM   or  $ SSYS
```

displays both user and system processes.

To get information about your own process use the command

```
$ SHOW PROCESS   or  $ SPRO
```

Detailed informations about all current processes are available with the command

```
$ SIN
or
$ SIN132                        !for terminals set to 132 character width
```

Detailed informations about a specific process are available by the command

```
$ PWATCH process-name   or   $ PWA/pid
```

where 'process-name' is the name of the process to be monitored and 'pid' is the process ID-number, obtainable by the `$ SSYS` command.

If you want to get various informations about the processor load use the command

```
$ MONITOR class
```

For the class you may select PROCESS/TOPCPU, SYSTEM, STATES, DISK, PAGE and others. Use `HELP MONITOR` to get all possibilities. Several of these are defined as command symbols, like the one to get the top CPU users:

```
$ MCPU
```

You will get these symbols by the command `$ SSYM */SEARCH=MONITOR`.

5.10 Logical Names

Logical names are used for nodes, devices, files, directories, and GOOSY data bases. They are defined (deleted) by

```
$ DEFINE logname equiv_name
or
$ ASSIGN equiv_name logname
e.g.
$ DEFINE MYROOT KP1$ROOT        !MYROOT is the new logical name
$ DEASSIGN logname
```

The `DEFINE` and `ASSIGN` commands are identical except the order of logical name and equivalence name.

Many of these names are defined already. E.g. disks on our Alpha AXP or VAXs may have names like D0, D1, DISK$DL111, KP1$DEVICE,... and the Tapes maybe named M0 (+ MU0 + MUA0), M1, LM0,... and the Alpha AXP or VAX nodes have their logical names, too,

A = ALICE, F = FRITZ = V6000A, AXP601, etc. Therefore you should be careful in using abbreviations for filenames, GOOSY Data Base names, or others. If you just use the filename 'A' without a file type declaration, the system will open a network link to the VAX called ALICE, instead of using the file 'A'. This will not happen with e.g. 'A.TXT'.

Logical names are defined in several 'scopes'. The scopes are SYSTEM, GROUP, JOB, PROCESS, and USER. Normally you can define logical names only in the JOB (valid for all subprocesses you create), PROCESS (valid just for your process), and USER (valid only for the next program executed) scopes.

```
$ DEFINE/JOB logname equiv_name
$ DEASSIGN/JOB logname
$ DEFINE/USER logname equiv_name
```

PROCESS is the default. All these **logical names are deassigned, if you logout**. Therefore you should define all logical names in your LOGIN.COM procedure.

You get an overview about active logical names by the commands (wildcards * are allowed for the logical names):

```
$ SHOW LOGICAL ! show all logical names
or
$ SLOG         ! show all logical names
$ SLOG/PROC    ! scope PROCESS
$ SLOG/GROUP   ! scope GROUP
$ SLOG/JOB     ! scope JOB (all subprocesses)
$ SLOG/SYS     ! scope SYSTEM
$ SLOG D*      ! disk names (all logical names starting with D)
$ SLOG M*      ! tape names (all logical names starting with M)
$ SLOG name    ! translation of one logical name
$ SLOG *ABC*   ! translation of all logical names containing the string ABC
```

GOOSY users may define GROUP or JOB logical names, too. These names are valid for all processes with the same account. The GROUP logical names remain active after logout, but will be deleted after a system restart.

A user GROUP will be defined by the system manager when he creates your Alpha AXP or VAX account. A so called User Identification Code, UIC, will be defined consisting of a 15 bit group number and a 15 bit user number. A UIC is noted as `[group,user]`, e.g. `[20010,1]`. The group and user number defines the access rights for disk files and for the logical name tables. If two users have the same group number, they will share the same group logical names. The UIC numbers are normally replaced by a User Identifier, i.e. you will see [name] instead of [n,m].

The user access to disk devices is not done directly but via specific logical names, the so called roots. A root is the logical name for a hardware disk name and a specific directory. These directories are created by the system manager. The root logical name can be used instead of the physical device name. E.g. on the VAX VSCN the logical name TEX$DEVICE translates to 2DKB100: and TEX$ROOT translates to TEX$DEVICE:[TEX.]. This method allows the system manager to access all files belonging to TEX in a simple way (see also chapter 6 at page 55).

Chapter 6

Files

6.1 File System and Directories

All kind of data on an Alpha AXP or VAX are kept in files. These files are sorted in directories and subdirectories, up to 7 levels deep. A full file specification is:

```
node::device:[directory]filename.filetype;version_number
```

where 'node::' is any valid Alpha AXP or VAX or computer connected by DECnet or HEPnet to your current Alpha AXP or VAX computer, 'device:' is the physical disk or its logical name equivalence, '[directory]' is the directory and subdirectory specification, 'filename' is the name of the file with up to 39 characters (including A-Z, 0-9, $, _ and -), 'filetype' is the type extension of the filename with up to 39 characters, and a version number.

All fields except the filename are defaulted by OpenVMS. Different to the IBM and other operating systems **any modifications on a file result in a copy with the version number incremented!** To get rid of the old versions, use the `PURGE` command. **NEVER** keep valid information in two files differing only in their version number. Leaving the Alpha AXP or VAX, you are asked to purge your files: please answer `Y` saving expensive disk space.

The file types may be freely chosen, but OpenVMS defaults certain types to certain kinds of files. See appendix C on page 107.

You create a subdirectory 'name' by

```
$ CREATE/DIRECTORY [.NAME]
```

This command creates a file 'name.DIR' on your current directory.

You can change your current default directory by

```
$ SET DEFAULT [newdir]
$ SET DEFAULT [.newdir]
$ SET DEFAULT [-]
```

In the first case, `newdir` is set 'absolute', in the second relative to your current directory. The third command gets you back one level.

You find where you are by

```
$ SHOW DEFAULT  or shorter:  $ SDEF
```

A list of all files in a directory is output to terminal by

```
$ DIR filespec
```

Filespec may be omitted, or specified partially using wildcards. Wildcards are * and % for any number of letters or one letter, respectively. They can be used in file names and types. * is allowed for version numbers, too. For directories a * means all directories on this level, whereas ... means all subdirectories of the current. Most commands requiring file specifications allow wildcarding! Some examples:

```
$ DIR *.PPL         ! all PPL files on current directory
$ DIR [...]*.TMP    ! all TMP files on current and subdirectories
$ DIR TEST.*;*      ! is same as
$ DIR TEST          ! all files named TEST
$ DIR GOO$EXE       ! all files on GOO$EXE (a logical name for a specific directory)
```

The various options for the `DIRECTORY` command are shown by `HELP`.

The `CFILTYPES` command lists all occurring types on current directory. It is useful to find 'dead' files.

```
$ CFILTYPES filespec
```

The `TDIR` command outputs the directory tree:

```
$ TDIR [directory]
```

The user access to disk devices is not done directly but via specific logical names, the so called roots. A root is the logical name for a hardware disk name and a specific directory. These directories are created by the system manager. The root logical name can be used instead of the physical device name. E.g. the 6DKA400:[KP2] is defined as root named KP2$ROOT. All users of KP2 get the pseudo device KP2$ROOT as there default disk. This method allows the system manager to access all files of KP2 in a simple way.

6.2 File Structure at GSI

There is a sub-structure in the main VAX VMScluster to increase the availability and performance. The VAX V6000A (FRITZ) is connected via a so called Computer Interconnect CI (a 70 Mbit/s serial bus) to a central disk controller HSC-70. The disks on this controller are named like 1DUAn: where n is a number, e.g. 1DUA21:. The VAX V6000A is also connected via Ethernet/FDDI to all VAXstations (more than 60) of the whole cluster. Each VAXstation has access to the HSC-70 controlled disks via the VAX V6000A, the HSC-70 disks are served by the VAX V6000A. That leads to I/O and network bottlenecks. Therefore each VAXstation has at least a locally connected disk for its own page and swap files. To reduce the access to only one system disk in the cluster four VAXstations and an Alpha AXP are installed as sub-servers having a copy of the system disk connected locally to them. These sub-servers support either all VAXstations like the AXP101 or the VAXstations belonging to a GSI group, namely KP1/KP3, KP2, AP, and ElEx. The sub-server of a group serves its system disk copy to all VAXstations belonging to this group. In addition, the sub-servers have the main user disks belonging to the individual group locally connected and they serve them like the system disk copy. The locally connected disks of the sub-servers are also available from the VAX on CI. Therefore user programs running on V6000A have access to the group disks, e.g. KP2$ROOT. But in this case the I/O and network load is higher than using a VAXstation of the group. If you want to access a locally connected disk of your VAXstation from another VAX (cross mounting), please contact the OpenVMS Advisory Service (see names on page 5).

The Alpha AXP VMScluster has two Digital 2100 Server Model A500MP as server machines. They are connected to each other and to centrally served disks via a local, fast data link (DSSI). Both servers have direct access to these central disks. They both have a local copy of the cluster's system disk which are shadowed. Therefore one of the server Alphas may crash without disturbing the satellite Alphas. The server Alphas are connected via FDDI directly to the FDDI crossbar switch, the GIGAswitch. Via FDDI they serve about 20 Alpha AXP satellites.

A common data root is the `DAY$ROOT` where all users may create there own directories to store larger data temporarily. The corresponding disks are connected to the HSC-70, i.e. the VAX V6000A serves these disks. This root might be purged by the VAX system managers if no more space is available. So, please store data only **temporarily** on the DAY$ROOT. No backup will be performed for this root. There are also several scratch disks available connected locally to the sub-servers or individual VAXstations. Please, use these local scratch disks mainly.

6.3 File Backup

The main disks or roots like SYSTEM, KP1$ROOT, KP2$ROOT, KP3$ROOT, FRS$ROOT, AP$ROOT, HAEDS$ROOT, KC1$ROOT, KC2$ROOT, THD$ROOT, US1$ROOT are fully

backed up approximately every fortnight by the OpenVMS Advisory Service (see names on page 5). Between two full backups each night all files not saved in the full backup will be backed up partially on disk. This partial backup will be overwritten each night, i.e. this is no incremental backup! If you lost a file please contact the OpenVMS Advisory Service immediately for restore (see names on page 5).

The scratch disks like DAY$ROOT or the locally attached user disks will not be backed up automatically. Please, backup all files or directories of such disks or roots you are interested in by your own. Contact the OpenVMS Advisory Service for details (see names on page 5).

6.4 File Handling

In the following, some useful DCL commands and procedures are listed concerning the file handling on the Alpha AXP or VAX.

list filenames — To list filenames inclusively types and versions in one or more directories use the DCL command `DIR` command, e.g.

```
$ DIR file1.type1;version1 !list exactly one file from your default directory

!list all files with type1 and type2 from your default directory
$ DIR *.type1,*.type2;*
$ DIR [...]*.type1      !list all files with type1 from all your directories
```

list filenames with FTP — To list filenames inclusively types and versions on remote computers connected to the Internet network use the File Transfer Protocol (FTP with TCP/IP Internet). Use the DCL command

```
$ FTP
```

within `FTP` use the following sequence of commands

```
FTP> CONNECT "host_name"
or
FTP> OPEN "host_name"
  ! use double quotes preserving lower case characters
    now login on the remote computer
FTP> DIR  "remote_file"   ! to get the remote directory (file name optional)
```

copy file — To copy files to other files and/or directory and/or root and/or Alpha AXP or VAX node use the DCL command `COPY`, e.g.

```
$ COPY file1.type1 file2.type2                !copy file1 to file2
$ COPY file1.type1 [directory]file1.type1     !copy into another directory
$ COPY file1.type1 xx$ROOT:[directory]file1.type1    !copy on to another root
$ COPY file1.type1 node::root:[directory]file1.type1 !copy on to another node
$ COPY node1::root1:[directory1]file1.type1;vers1 -
_$ node2::root2:[directory2]file2.type2;vers2
```

To copy several files into separate new files use wildcard characters, e.g.

```
$ COPY file1.type1,file2.type2,file3.type3   *
$ COPY *.type   *
```

To copy several files into one new files use the following command, e.g.

```
$ COPY file1.type1,file2.type2,file3.type3   newfile.typenew
$ COPY *.type   newfile.type
```

copy file with FTP — You can copy files with the File Transfer Protocol (FTP with TCP/IP Internet) to other files and/or directory to and from remote computers connected to the Internet network. Use the DCL command

```
$ FTP
```

within `FTP` use the following sequence of commands

```
FTP> CONNECT "host_name"
or
FTP> OPEN "host_name"
  ! use double quotes preserving lower case characters
    now login on the remote computer
FTP> GET  "remote_file" local_file    ! to get a file from remote
FTP> PUT  local_file "remote_file"    ! to write a file to remote
```

To preserve OpenVMS specific file attributes use the `/FDL` option for the `GET` and `PUT` commands.

append file — To copy one or more files to the end of another file use the DCL command `APPEND`, e.g.

```
$ APPEND file1.type1,file2.type2,file3.type3   oldfile.type
```

append file with FTP — You can append to files on remote computers connected to the Internet network with the File Transfer Protocol (FTP with TCP/IP Internet). Use the DCL command

```
$ FTP
```

within FTP use the following sequence of commands

```
FTP> CONNECT "host_name"
or
FTP> OPEN "host_name"
  ! use double quotes preserving lower case characters
    now login on the remote computer
FTP> APPEND  local_file "remote_file" ! append to a remote file
```

rename file — To rename one or more files inclusively types and versions use the DCL command RENAME, e.g.

```
$ RENAME file1.type1;version1   file2.type2;version2

!rename all files with type1 into files with type2
$ RENAME *.type1;*   *.type2;*
```

delete file — To delete one or more files use the DCL command DELETE. You must give the filename, the type, and the version number or wildcards to delete a file, e.g.

```
$ DELETE file1.type1;version1,file2.type2;version2

!delete all files of type1 and type2 in the default directory
$ DELETE *.type1;*,*.type2;*
```

create file — To create one file use the editor LSEDIT (see chapter 7 section 7.1 on page 65) or the DCL command CREATE, e.g.

```
$ CREATE file1.type1
now type in the text
and finish this text by Ctrl Z
```

type file — To type one or more text files on your terminal use the DCL command TYPE, e.g.

```
$ TYPE file1.type1;version1,file2.type2;version2

!type all files of type1 from the default directory on your terminal
$ TYPE *.type1;*
```

print file — To print one or more text files on a laser printer use the DCL command POP -Pnnxxx (where nn is the printer device and xxx is the format; see also HELP POP or HELP PRINTER and appendix D on page 109) e.g.

```
$ POP -Pnnxxx file1.type1;version1,file2.type2;version2

!print all files of type1 from the default directory on the laser printer P08
! with 80 characters per line on white paper:
$ POP -P08WAS  *.type1;*
```

dump file — To dump the contents of a file in hexadecimal, decimal or octal use the DCL command `DUMP`, e.g.

```
$ DUMP file1.type1            !dump in hexadecimal on your terminal
$ DUMP/OUT=ofile file1.type1  !dump in hexadecimal into file ofile.DMP
$ DUMP/OCT file1.type1        !dump in octal on your terminal
$ DUMP/DEC file1.type1        !dump in decimal on your terminal
$ DUMP/BLOCKS=(START:1,END:5) file1.type1   !dump from block 1 to 5
```

sort file — To sort the contents of a text file use the DCL command `SORT`, e.g.

```
!the contents of file1 will be copied in alphabetic order to file2
$ SORT file1.type1 file2.type2

!the contents of file1 will be copied in alphabetic order to file2
! using just the first 5 characters for the ordering
$ SORT/KEY=(POSITION:1,SIZE:5) file1.type1 file2.type2

!the contents of file1 will be copied in alphabetic order to file2
! using FIRST the first 5 characters for the ordering and then the
! 3 characters starting at row 10
$ SORT/KEY=(POSITION:1,SIZE:5)/KEY=(POSITION:10,SIZE:3) file1.type1 file2.type2
```

get file differences — To get differences of the contents of two text files use the DCL command `DIFFERENCES` (see also the GSI utility `CDIFFER` on page 93).

```
!list the differences of file1 and file 2 on your terminal
$ DIFF file1.type1 file2.type2

!list the differences of file1 and file 2 in parallel on your terminal
$ DIFF/PARALLEL file1.type1 file2.type2

!write the differences of file1 and file 2 into file ofile.DIF
$ DIFF/out=ofile file1.type1 file2.type2
```

search strings in files — To search text strings in the contents of text files use the DCL command `SEARCH` (see also the GSI utility `LIBSEARCH` on page 77).

```
!search "string" in file1 and file2 and list lines found on your terminal
$ SEARCH file1.type1,file2.type2 "string"

!search "string" in all files of type1 and write lines found into file ofile.LIS
$ SEARCH/OUT=ofile *.type1 "string"

!search "string" in all files of type1 and list +- 5 lines found on your terminal
$ SEARCH *.type1 "string" /WINDOW=(5,5)

!search "string" in all files of type1 and list only filenames on your terminal
$ SEARCH *.type1 "string" /WINDOW=0

!search lines containing "string1" AND "string2" in all files of type1
$ SEARCH *.type1 "string1","string2" /MATCH=AND
```

6.5 Magtape Handling

Magnetic tapes are handled by OpenVMS like directory structured devices. That means you can copy files to and from magnetic tapes with the identical commands like described in the section 6.4 on page 58. To initialize a new magnetic tape from the stock use the following sequence:

1. Look for a free magnetic tape drive in the Messtation (room 1.124), on a MicroVAX like MVIIG, or an EXAbyte or TZ86 or TZ87 or DLT2000 cassette drive locally attached to an Alpha AXP or a VAXstation (e.g. AXP601 or VSCN) and remember its number (label on the front panel of the tape drive).

2. Mount your tape on this drive manually.

3. Go to your Alpha AXP or VAX terminal and type the following command to lock the tape drive for your own usage:

   ```
   $ ALLOCATE Mn:
   ```

4. Then initialize the tape only if it is a brand new tape or an old tape you want to overwrite:

   ```
   $ INITIALIZE Mn: label /DENSITY=d
   ```

   with 'Mn' equals the tape drive number (e.g. M2 or MK500), 'label' = any character string of max. 6 alphanumeric characters, and 'd' = the write density in bit per inch (bpi) of the tape (default: 6250, alternative:1600, not applicable for EXAbyte, TK50, TZ86/87, or DLT2000 drives). There must be a space before the 'label'. This command writes a standard ANSI-label on your tape. This label allows you to mount the tape by software later.

Currently only EXAbyte cassettes must be initialized specifically, if you want to copy files on a new EXAbyte cassette. You must first write an arbitrary (short) Backup saveset on it

```
$ MOUNT /FOREIGN tape:
$ BACKUP file.type tape:file.bck /SAVE
$ DISMOUNT /NOUNLOAD tape:
e.g.
$ MOUNT /FOR LM1:
$ BACKUP ADAM.TXT LM1:ADAM.BCK /SAVE
$ DISMOUNT /NOUNL LM1:
```

Now you can use the cassette for the normal DCL `COPY`.

5. If you want to write files on your tape or read files from it you have to mount the tape by software using the following command:

   ```
   $ MOUNT Mn: label
   ```

   with 'Mn' =the tape drive number (e.g. M2:) and 'label' = any character string of max. 6 alphanumeric characters. There must be a space before the 'label'. If you do not know the label of the tape you may also ignore it by the DCL command

   ```
   $ MOUNT Mn: /OVER=ID
   ```

6. Now you may copy files from or to the tape, use the `DIRECTORY` command, etc.

7. To dismount the tape from software use the command:

   ```
   $ DISMOUNT Mn:
   ```

   with 'Mn' =the tape drive number (e.g. M2:). The `DISMOUNT` command will rewind and unload your tape from the drive automatically. To avoid this, use the `/NOUNLOAD` qualifier.

8. Now free the tape drive again by:

   ```
   $ DEALLOCATE Mn:
   ```

9. Finally remove the tape physically from the tape drive.

Chapter 7

Program Development

7.1 Editing

We recommend to use the language sensitive editor `LSEDIT` invoked from a text terminal by

```
$ LSEDIT filename.type    or just   $ LSE filename.type
!If you edit the same file again just type
$ LSE
!The editor remembers the last filename edited.
```

You may also start LSEDIT from a DECwindows/Motif DECterm with

```
$ LSEDIT/INTERFACE=DECWINDOWS filename.type
or just
$ LSE/INT=DECW filename.type
```

or select from the DECwindows/Motif Session Manager Application Menu LSEDIT. A new window pops up for LSEDIT. If this entry is not part of the Application Menu select first the option from the Session Manager Option menu. A new window Menus pops up. Select Applications from the Menu Names by a single MB1 mouse click, write `LSEDIT` in the field just above DCL Command and write `@VUE$LIBRARY:LSE$EDIT.COM` in the field right to DCL Command. Select the up-arrow right from Optional Qualifiers which places LSEDIT in the Item Names list. Select LSEDIT in this list and select the left-arrow between the Item Names list and the Applications list which places LSEDIT finally into this list. Exit by selecting the OK button. From the Session Manager window select within the Options menu the Save Session Manager option to save the current set-up for later login.

In the following we summarize the main `LSEDIT` editor features: (In the LSEDIT manual and the LSEDIT help the `PF1` key is called `GOLD` key.)

- Files have never line numbers or NULLs like on the IBM. Nevertheless you can get the line number where the cursor is located currently by typing `GOLD` `Prev Screen`. You can also move the cursor to a desired line number n by typing `GOLD` `Insert Here`

- Upper and lower case characters are always displayed as they are.
- You switch between **insert** and **overstrike** mode by Ctrl A.
- A new line is entered by the Return key instead of the Enter key. To proceed the cursor to a new line use 0 or ⇓ for going down or ⇑ for going up with the cursor within your text.
- The screen splitting is entered or left by PF1 =. To switch between the two windows, use F20 or GOLD ⇑.
- To delete the character left from the cursor, use Delete. To delete the character on the cursor position (right), use ,. You can undelete the character by PF1 ,.
- To delete a word left from the cursor, use F13. To delete a word on cursor position (right), use -. You can undelete the word by PF1 -.
- Delete a line from the cursor position left by Ctrl U. To delete the rest of a line up to the begin of the next line right from the cursor position use PF4. To delete the rest of a line right from the cursor position use PF1 2. You can undelete the line by PF1 PF4.
- Search a string by Find or PF1 PF3. The string to be searched is prompted. Upper and lowercase characters are equivalent. Note that the search with wildcards is not supported by default! To search with the wildcards * or % use Do SEARCH/PATTERN Return and then enter the string containing the wildcards. If you want to include the characters * or % in the string you search precede them by a \, e.g. **adam*\%** would search for 'adam' followed by any number of characters followed by one % character.
  Pressing just PF3 continues the search in any case. The search direction is defined by 4 for downwards and 5 for upwards. The search string is replaced by the content of the paste buffer (see below) by PF1 9. Search and replacement (substitution) can be done at once by PF1 Enter.
- Shift text left use F11, shift text right use F12.
- Move the cursor to the end of a line by 2.
- Move the cursor to start of the next line by 0.
- Scroll text several lines up/down by Next Screen or Prev Screen or 8. The direction for 8 is defined by 4 for downwards and 5 for upwards.
- The direction of move cursor, scroll text, and search commands is changed by 4 for downwards and 5 for upwards. The direction will be kept until 4 or 5 are pressed again.
- Move the cursor to the bottom of the text buffer by PF1 4.

- Move the cursor to the top of the text buffer by PF1 5.
- Rewrite (refresh) the whole screen by Ctrl W.
- To open a new text file and with it a new buffer use F9. It prompts you for the file name. The buffer name will become equal to the file name.
- To switch the current buffer to another buffer, use F10. The buffer name will be prompted.
- To show all available buffers use GOLD Select. Select one of them by moving the cursor to the buffer line and use Select.
- To move and/or delete blocks, use the following sequence:

  Select or . to enter the select mode,
  move cursor, the selected range is displayed reverse,
  Remove or 6 the selected range is deleted (moved to paste buffer),
  use Insert Here or PF1 6 to restore the deleted range if wanted,
  move cursor to the new position,
  Insert Here or PF1 6 the previous deleted range is inserted.

  The contents of the paste buffer is kept, if you switch between windows or edit buffers. It will only be overwritten by 6.

- A repetition factor for any key may be entered by PF1 and the number (on main keypad), e.g.:

  PF1 10 0 to move the cursor 10 lines.
  PF1 10 6 to insert the the contents of the paste buffer 10 times.
  PF1 1000 PF1 Enter to replace 1000 times the searched strings by the contents of the paste buffer.

- Enter command level by Do. The following commands may be useful:

```
SPELL           ! invoke English spell checker for the current buffer

READ file       ! inserts file at cursor position
WRITE file /SEL ! writes selected range to file
WRITE file      ! writes current buffer to file

EXIT            ! exit, save all files
QUIT            ! exit, do not change any file

SHOW BUFFER     ! list of buffer names
```

```
HELP            ! to get help
```

A shorthand for READ is PF1 F9 (the filename is prompted).
A shorthand for WRITE/SELECT is PF1 F10 (the filename is prompted).

- To get a keypad layout, enter PF2. To get all keypad definitions, enter PF1 F7.
- Enter the OpenVMS help by F7.
- Recovery: If the Alpha AXP or VAX crashes during an LSEDIT session or if you exit by mistake with Ctrl Y (**never do that!**), you recover the last session by

  ```
  $ LSEDIT filename /RECOVER
  ```

  NOTE: After test compile (see below), you must recover the file version you started with which is NOT the last one because test compile writes the buffer to the file creating a new version.

- LSEDIT macros can be written in TPU syntax. Examples are in GSI$MANAGER:LSEINIT.TPU.

Besides the screen editing facilities there are some more useful features:

- Language sensitive commands:
  LSEDIT recognizes from the file type the kind of file. E.g. if type is .PPL or .PLI, it assumes a PL/I program, if type is .FOR a FORTRAN program and if type is .COM a DCL procedure. The language constructs of the language are implemented as **placeholders**, **tokens**, and **procedure calls**. These are inserted in the text enclosed in [ ] or { }. To replace a placeholder just type ahead. Other controlling sequences are:

  Ctrl E to expand a placeholder, token, or procedure call

  Ctrl N to go to the next placeholder, token, or procedure call

  Ctrl K to delete a placeholder, token, or procedure call

  If { } appears, you must insert something. If you type Ctrl E, you get a description of what is required here. You can type a placeholder name, press Ctrl E and the placeholder will be expanded either to a set of other placeholders or tokens. You get a list of present tokens and placeholders by PF19. The best way is to play with this! We implemented some structures for DCL procedures which are not very comfortable in DCL, i.e. IF_ELSE, DO loops and temporary file names: LOOP Ctrl E expands a DCL DO loop, IF_ELSE Ctrl E expands a DCL IF-THEN-ELSE construct

- Test compile:
  This is a very useful feature of the LSEDIT. Press F17 and the current file will be compiled. The error messages are displayed in the top window. You may now correct your source following the messages. **NOTE** that the current buffer is written to its file! Ctrl N will skip to the next error, Ctrl P will skip to the previous error.

- Include PL/I calling sequence for a module:
  With the the F8 key you can include a calling statement for a module. The module name is prompted. This supports all runtime library modules, all system modules, and the GOOSY modules.

- Include text modules from libraries:
  With PF1 F8 you can include a module from a text library. The library and the module are prompted.

- Execute one DCL line from your text:
  If you have a DCL line in the text your are editing place the cursor in front of this line and press F19. The command will be executed as a DCL command and the output will be placed into your text just after the command line.

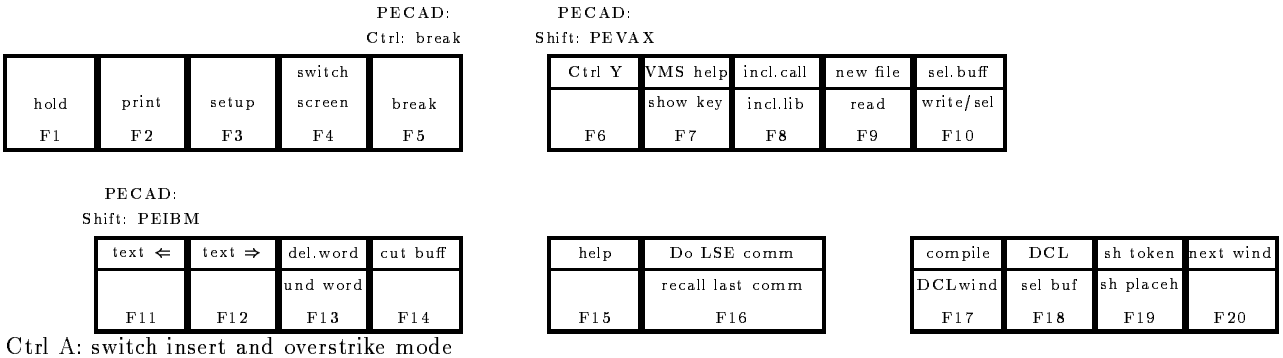
- Execute several DCL lines:
  Press GOLD F17 to split your screen. Enter DCL commands to the prompt in the lowest line of the screen. The output will be placed in the upper new window. Leave this mode by just typing Return.

- Create another process (SPAWN):
  With the F18 key you can execute a DCL line (which is prompted) in a subprocess. After the execution of the DCL command, LSEDIT returns immediately to the edit session. If you press F18 Return you are in a new, spawned DCL process and you may enter all DCL commands or run any program. To leave this mode and return to your current LSEDIT session, type LOGOUT on DCL level.

To leave the editor, use GOLD Remove or press DO-key and type EXIT Return to exit with writing changed text into a new version of the file or use GOLD Next Screen or press DO-key and type QUIT Return to quit (no modifications are saved, no editing in the file is saved, but note that sometimes a buffer is written into its file during a session, e.g. with the compile command!).

In the figure 7.1 on page 70 you see the LSEDIT keypad layout.

The upper key values are the simple key hits, the lower are entered with a preceding GOLD(=PF1)-key hit.



Ctrl A: switch insert and overstrike mode
Ctrl B: go to previous error (step)
Ctrl D: go to symbol declaration
Ctrl E: expand placeholder, token, alias
GOLD Ctrl E: unexpand
Ctrl F: go to next error (step)
Ctrl G: go to source
Ctrl H: move cursor to begin of line
Ctrl J: delete word left to cursor
Ctrl K: delete placeholder
GOLD Ctrl K: undelete placeholder
Ctrl N: go to next placeholder
Ctrl P: go to previous placeholder
Ctrl R: reshow complete screen
Ctrl T: shows CPU and elapse time
Ctrl U: delete line left to cursor
Ctrl W: reshow complete screen
Ctrl Y: interrupt current program
Ctrl Z: do LSE command or end of input
GOLD Ctrl Z: do LSE command or end of input
GOLD =: split/unsplit screen

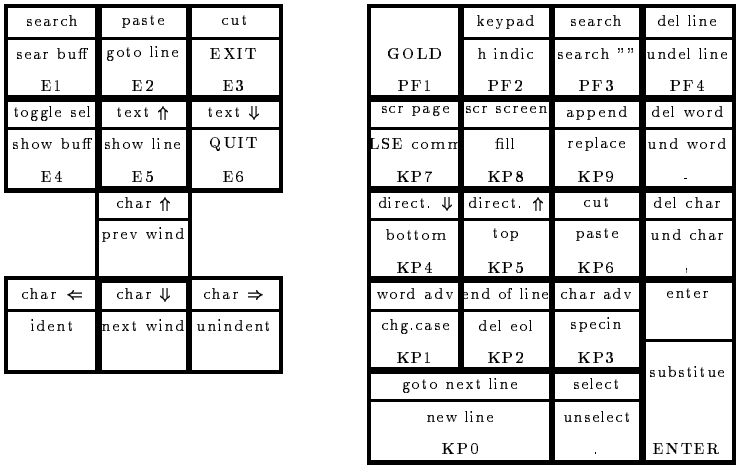


Figure 7.1: The Special Keypad Layout for the LSEDIT.

7.2 Compiling

You may call the following compilers directly as DCL commands:

```
$ BASIC
$ CC                    ! for the C compiler
$ CXX                   ! for the C++ compiler on AXP only
$ FORTRAN
$ MACRO                 ! for the assembler
$ MODULA                ! on the VAXes only
$ OPS5
$ PASCAL
$ PLI
```

But the recommended command to call a compiler is

```
$ COMPILE filename.type
```

The compilers are invoked by `$ COMPILE filename.type`, depending on the file type. Standard file types are:

.C = C sources
.FOR = Fortran sources
.MAR = VAX Assembler sources
.MOD = Modula 2 sources
.PAS = Pascal sources
.PLI = PLI sources
.PLITEMP = temporary PLI-source generated by GOOSY preprocessor.
.PPL = GOOSY PLI preprocessor sources

The default type for COMPILE is .PPL, but it can be changed in the login procedure by DEFCOMPI :== <type>. The COMPILE command is also available on IBM for GOOSY preprocessor code. If a file has been compiled already and the source has not changed, it is NOT compiled until the /COM qualifier is specified. Thus one can compile a set of files in a command procedure. Only modified files are compiled saving time. The output of a compilation is (are) object file(s) named like the source file, but with type .OBJ. These files are input for the linker. Some examples:

```
$ COMPILE X             ! compile X.PPL
$ COMPILE X.FOR         ! compile X.FOR
$ COMPILE X,Y           ! compile X.PPL and Y.PPL
$ COMPILE X/DEB         ! compile X.PPL with DEBUG (see below)
$ COMPILE X/COM         ! compile X.PPL again
$ COMPILE               ! compile last set
$ COMPILE X /KEEP       ! do not delete X.PLITEMP (PL/I code)
$ COMPILE X*            ! compile all .PPL files X*
```

```
$ COMPILE X/OLB=OPRIV  ! insert object of X in library OPRIV
$ COMPILE X/LIB=TPRIV  ! Search TPRIV for includes
$ COMPILE X /QUALIFIER=(LIST,SHOW=ALL)
                        ! write a file X.LIS containing line numbers
                        ! PL/I switches as shown by HELP PLI must
                        ! be passed to COMPILE by this way
```

You may also compile programs during a LSEDIT session pressing the F17 key. In case of compilation errors the screen will split showing the error conditions in the upper screen part and the source in the lower part. You may step forwards through the errors pressing Ctrl F and Ctrl B for stepping backwards. With Ctrl G the cursor will move to the faulty source line which can be edited immediately. If a pre-compiler was used, the source displayed is the source after the pre-compilation. In such a case you must edit the original pre-compiler source to eliminate errors. Select the buffer with the original source using GOLD Select. Unsplit the screen with GOLD =.

7.3 Linking

A program must be linked with all called modules to be executed. This is done by the LINK command. The modules, except the main program, may be in libraries. Libraries may be specified with the LINK command. These are scanned first. Then a list of default libraries is scanned. This list is displayed by

```
$ SLOG LNK$*
```

The output of the link step is a so called executable image, the program in a file which can be executed by the RUN command. The default output of the linker is the image file of type .EXE. The name is the filename of the main program. Examples for the LINK command:

```
$ LINK X,Y,Z            ! link object files X.OBJ, Y.OBJ, Z.OBJ to X.EXE
$ LINK X,OPRIV/LIBRARY  ! Link object file X.OBJ with modules from OPRIV to X.EXE
$ LINK X,Y/DEBUG        ! Link X.OBJ and Y.OBJ with debugger to X.EXE
```

7.4 Executing

A linked program is executed by

```
$ RUN program
```

Another, more elegant way is to create a symbol 'command' to execute the program:

```
$ command == "$device:[directory]program.EXE"
e.g.
$ BETA == "$KP1$ROOT:[ADAM]BETA.EXE"  ! BETA can be used now as a command
```

This should again be done in the LOGIN.COM file. Then the program is executed by `command`. The LIB$GET_FOREIGN routine called in the program returns any characters typed behind `command`. This is the method to pass parameters to a program together with the execution.

A program can be canceled by Ctrl Y. **All utility programs should be rather terminated on terminal input request by** Ctrl Z.

7.5 Debugging

The OpenVMS debugger is a very powerful tool to find errors. It allows to set break points on source lines, step line by line, inspect and set variables etc. Modules to be debugged must be compiled with the /DEB option. The main program must be linked with the /DEB option,

```
$ COMPILE program/DEB
$ LINK program/DEB
```

If you run your program with

```
$ RUN program
```

the DEBUG mode is entered by default. If you want to run the same program without DEBUG start it with

```
$ RUN/NODEBUG program
```

If you are under DECwindows/Motif new Debugger windows are popped up. The handling of the DECwindows/Motif Debugger is somewhat unhandy. If you want to suppress the window mode, i.e. you want to run in the text terminal command mode, you must define the following logical

```
$ DEFINE /JOB DBG$DECW$DISPLAY ""
```

If you want to separate the Debugger terminal input and output to another terminal use the DCL command

```
$ DEBWIN
```

It will define the logicals

```
$ DEFINE /JOB DBG$INPUT term:
$ DEFINE /JOB DBG$OUTPUT term:
```

where term: is a valid terminal line, e.g. LTA123:. If you want to have this terminal to be a DECwindows/Motif DECterm window executed the following

```
$ CREATE/TERMINAL/NOPROCESS -
    /WINDOW_ATTR=(TITLE="Debugger",ICON_NAME="Debugger",ROWS=40) -
    /DEFINE_LOGICAL=(TABLE=LNM$JOB,DBG$INPUT,DBG$OUTPUT)
$ ALLOCATE DBG$OUTPUT
```

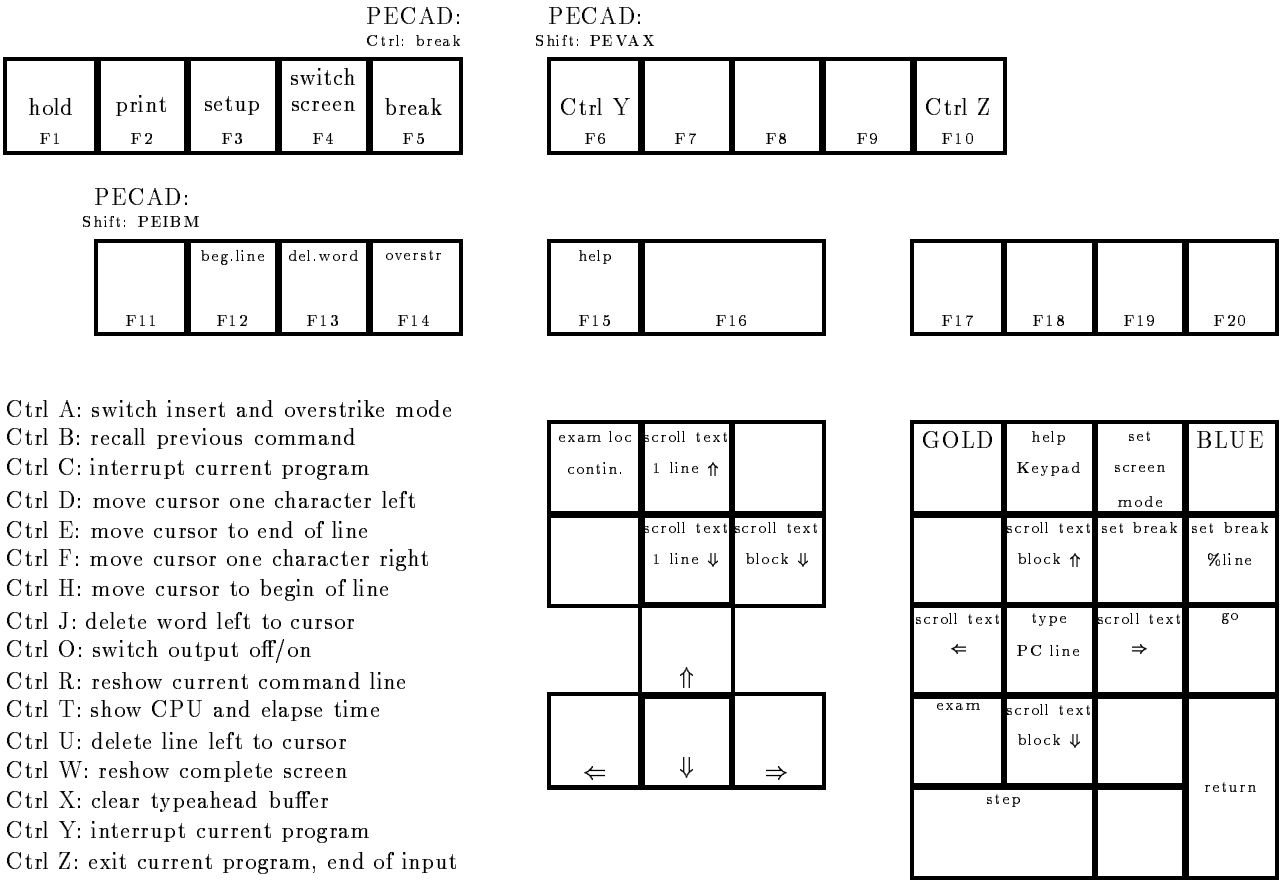
In the following only the text terminal command mode will be described. For the DECwindows/Motif mode refer to on-line Help or the Debugger manual. When you enter the DEBUG menu you may get detailed information with the DEBUG command `HELP`.

Some most often used commands are shown in the following:

```
DBG> SET BREAK module\%LINE # ! set a break point in line #
DBG> GO                       ! Execute to next break point
DBG> STEP or <KP_0>           ! Execute one line
DBG> EXAMINE variable         ! Examine a variable in decimal
DBG> EXAMINE                  ! Examine the following variable in decimal
DBG> EXAMINE .                ! Examine the last variable again
DBG> EXAMINE @address         ! Examine a location whose address is in 'address'
DBG> EXAMINE /HEX variable    ! Examine a variable in hexadecimal
DBG> EXAMINE /OCT variable    ! Examine a variable in octal
DBG> EXAMINE /ASCII variable  ! Examine a variable as an ASCII character
```

You may examine any single variable, an array, a structure, or a member of a structure. If you examine an array without defining the index limits, e.g. EXAM BETA instead of EXAM BETA(10:20), you will get the contents of all array members. This might be a long procedure if your array is large. You can interrupt the output without leaving the whole debug session by typing Ctrl C). You may although suppress the output by typing Ctrl O but since the debugger continues to examine your array it will still will take some time. **So be careful with the examination of arrays**.

```
DBG> DEPOSIT variable=value   ! Set a variable in decimal
DBG> DEPOSIT variable=%HEX value   ! Set a variable in hexadecimal
DBG> DEPOSIT variable=%OCT value   ! Set a variable in octal
DBG> DEPOSIT variable='string'! Set a variable with a string
DBG> <INSERT HERE>            ! Scroll source up 1 line
DBG> <PREV>                   ! Scroll source down 1 line
DBG> <KP_8>                   ! Scroll source up
DBG> <KP_2>                   ! Scroll source down
DBG> <KP_5>                   ! Return to cursor position
DBG> TYPE #                   ! Type line # of the current module
DBG> TYPE module \#           ! Type line # of module
DBG> SEARCH module string     ! Search string in module text
DBG> SEARCH                   ! Proceed searching the string in module text
```



Ctrl A: switch insert and overstrike mode
Ctrl B: recall previous command
Ctrl C: interrupt current program
Ctrl D: move cursor one character left
Ctrl E: move cursor to end of line
Ctrl F: move cursor one character right
Ctrl H: move cursor to begin of line
Ctrl J: delete word left to cursor
Ctrl O: switch output off/on
Ctrl R: reshow current command line
Ctrl T: show CPU and elapse time
Ctrl U: delete line left to cursor
Ctrl W: reshow complete screen
Ctrl X: clear typeahead buffer
Ctrl Y: interrupt current program
Ctrl Z: exit current program, end of input

More commands are available after pressing the GOLD or BLUE key. Use GOLD Help or BLUE Help to get these keys.

Figure 7.2: The Special Keypad Layout for the OpenVMS DEBUG.

```
DBG> SET LANGUAGE               ! If mixed code of FORTRAN and C set language
DBG> SET SYMBOL                 ! If mixed code of FORTRAN and C set symbols
DBG> EDIT                       ! Call the LSEDIT for the current module source
DBG> SPAWN                      ! Spawn a new DCL process, exit it with $ LOGOUT
DBG> <Ctrl>W                    ! Rewrite (refresh) the whole screen
DBG> EXIT or <Ctrl>Z            ! Exit debugger
```

In the figure 7.2 on page 75 you see the DEBUG keypad layout.

7.6 OpenVMS Libraries

The OpenVMS libraries are similar to IBM PDS's. But IBM **members** are called here **modules** (that's live). OpenVMS libraries are supported only by specific commands and utilities. Only libraries of specific types are supported: TEXT, HELP, OBJECT and MACRO libraries. Libraries are created by

```
$ LIBRARY/CREATE/TEXT   filename.tlb
$ LIBRARY/CREATE/HELP   filename.hlb
$ LIBRARY/CREATE/OBJECT filename.olb
$ LIBRARY/CREATE/MACRO  filename.mlb
```

Generally one should define a **logical name** for **any library** file including the device and directory where it resides. Again, these definitions must be done in the LOGIN.COM procedure. The `LIBRARY` command handles libraries. In the following we give most used commands and applications:

HELP Libraries

Used by HELP command.

```
$ LIB/INSERT  library file.HLP     ! insert new help text into library from file
$ LIB/REPLACE library file.HLP     ! replace help text in library from file
$ LIB/EXTRACT=module/OUT=file.HLP library ! extract help text from library
```

The format of the help files is described in Appendix G. File type `.HLP` is defaulted.

OBJECT Libraries

Used by LINK command.

```
$ LIB/REPLACE library file.OBJ              ! replace objects from file
$ LIB/EXTRACT=module/OUT=file.OBJ library   ! extract objects from library
```

The module names are determined by the objects in the file. Several objects may be in one file! File type `.OBJ` is defaulted.
To delete modules use for all library types:

```
$ LIBRARY/DELETE=module library
$ LIBRARY/DELETE=(module,module,...) library
```

where module may contain wildcards.

TEXT Libraries

Used for PL/I includes and GOOSY data type declarations compilers and GOOSY.

```
$ LIB/REPLACE library file.TXT             ! replace file as module 'file'
$ LIB/REPLACE/MODULE=name library file.TXT ! replace file as module 'name'
$ LIB/EXTRACT=module/OUT=file.TXT library  ! copy module to file
```

The text in the file / module may have any format. File type `.TEX` is defaulted.

Usage of libraries

Create the following libraries on your master directory (= current after login):

```
$ LIBRARY/CREATE/TEXT PRIV.TLB
$ LIBRARY/CREATE/HELP PRIV.HLB
$ LIBRARY/CREATE/OBJ  PRIV.OLB
```

Add following logical name definitions in your LOGIN.COM:

```
$ DEFINE/JOB TPRIV SYS$LOGIN:PRIV.TLB
$ DEFINE/JOB PLI$LIBRARY TPRIV
$ DEFINE/JOB FOR$LIBRARY TPRIV$ ! optional for FORTRAN
$ DEFINE/JOB HPRIV SYS$LOGIN:PRIV.HLB
$ DEFINE/JOB HLP$LIBRARY HPRIV
$ DEFINE/JOB OPRIV SYS$LOGIN:PRIV.OLB
$ DEFINE/JOB LNK$LIBRARY OPRIV
```

With these definitions, the `HELP` command searches first through your private help library HPRIV, the `COMPILE` command searches include modules first in your TPRIV, and the `LINK` command searches modules first in your OPRIV.

Related Commands

LIBLIS library module
: outputs a list of the modules in the library. The module specification may contain wildcards.

LIBTYPE library module
: outputs the content of the specified modules to the terminal.

LIBSEARCH library module list="search_parameter"
: searches through the specified modules. Search parameters are internally passed to SEARCH command.

LIBCOPY sourcelib module destinationlib module
: copies modules from one library to another.

LIBEXTR sourcelib module
extracts modules from library. Wildcards for module are supported. Module names may be in a text file which must be specified as @file.

LIBDEL sourcelib module
deletes a module from library. Wildcards for module are supported. Module names may be in a text file which must be specified as @file.

7.7 Source Code Analysis SCA

Together with the Language Sensible Editor LSEDIT a Source Code Analyzer SCA is available. SCA is an interactive, multilanguage, source code cross-reference and source code analysis tool that aids developers in understanding large-scale software systems. Because SCA deals with an entire software system, instead of individual modules, it is an effective tool during implementation and maintenance phases of a project. SCA stores compiler-generated information about the set of build sources for querying in one unique location, an SCA library. Thus, SCA is a query tool that allows you to reference and query time-stamped source information that directly corresponds to source modules in your system. When these sources are no longer of value, you can modify or delete the SCA library. The library data generated by supporting OpenVMS compilers consists of names of all of the symbols, modules, and files contained in a specific snapshot of the source. Once SCA libraries are created, you can select a library and query its contents from within LSEDIT, at the DCL level, or via the SCA callable interface. You may do cross-referencing and analysis (locate symbols and their occurrences) and consistency checking (of symbols).

You create a SCA library in the specified user directory by the SCA command

```
SCA> CREATE LIBRARY directory
```

You produce analysis data by using the compile DCL command line of the form

```
$ compiler /ANALYSIS_DATA[=file]  source-file[,...]
e.g.
$ FORTRAN /ANALYSIS_DATA TEST     ! from TEST.FOR produce TEST.OBJ and TEST.ANA
```

For details see DECset or LSEDIT/SCA manuals and Help or the Bookreader or contact the OpenVMS Advisory Service (see names on page 5).

7.8 DEC Performance and Coverage Analyzer PCA

PCA helps you to produce efficient and reliable applications by analyzing your program's dynamic behavior. PCA also measures codepath coverage within your program so that you can devise tests that exercise all parts of your application.

PCA has two operational components:

1. the collector:
   It gathers performance or test coverage data on a running program and writes that data to a performance data file. You may select either the main image or one of the shareable images in the program's address space. It measures the dynamic behavior of the image you have selected. You may select one or more of the following data:

   - program counter (PC) sampling at fixed sample time
   - CPU sampling at virtual-process sample time
   - counters of the exact number of times that specified program locations are executed
   - coverage data indicating which portions of your program are, or are not, executed during each test run
   - page default data
   - system service data
   - input/output data

2. the analyzer:
   It reads the performance data file produced by the collector and processes the data to produce performance and coverage histograms and tables.

You can run the collector and the analyzer in batch as well as interactively.

To invoke the collector compile all source files you want to analyze with the `/DEBUG` qualifier, e.g.

```
$ FORTRAN /DEBUG TEST
```

Then link the whole program with the `/DEBUG=SYS$LIBRARY:PCA$OBJ` library, e.g.

```
$ LINK /DEBUG=SYS$LIBRARY:PCA$OBJ   TEST
```

Finally run the program, e.g.

```
$ RUN TEST
```

The collector will come up with the prompt

```
PCAC>
```

Now enter PCA collector commands like

```
PCAC> SET DATAFILE TEST
PCAC> SET PC_SAMPLING
PCAC> GO
```

These commands will write the collected data to the file TEST.PCA, will set PC sampling mode and starts the program.

To invoke the analyzer use the DCL command `PCA`

```
$ PCA /COMMAND="command1; command2; ..." data-file
e.g.
$ PCA /COMMAND="SHOW DATAFILE; SHOW LANGUAGE" TEST
```

The analyzer will come up with the prompt

```
PCAA>
```

Now enter PCA analyzer commands like

```
PCAA> NEXT      ! produces a source plot with PC sampling data
```

For details see DEC Performance and Coverage Analyzer PCA manuals and Help or the Bookreader or contact the OpenVMS Advisory Service (see names on page 5).

7.9 Module and Code Management

A software system can have many program files, object libraries, include files, compilers, and compilation and linking options. The more complex the system, the more difficult it is to reproduce the same program image for each build.

The Module Management System MMS automates and simplifies the building of software systems. It can build simple programs consisting of one or more source files, or complex programs consisting of many source files, message files, and documentation files. It is similar to the `make` command of UNIX.

With MMS, you can specify exactly how a software system is to be built and rebuilt. You do this by using a description file in which you describe the components of the system and the file dependencies used to build and rebuild the system.

Each time you run MMS, it follows the description file you have created, reads the components and dependencies, and builds the same system.

During software development, programmers continually make changes to project files. The Code Management System CMS stores and monitors these files.

CMS allows you to store project files in a central library where they are available to all project members. Some of the tasks you can perform on these files are:

- store files (called elements) in a library
- fetch elements, modify them, and test them in your own directory
- control concurrent modifications to the same element
- Merge concurrent modifications to an element
- create successive versions (called generations) of elements
- compare two generations of an element within a library
- organize related library elements into groups
- define a set of generations of elements as a class to make up a base level or release version of a project
- track which users are working on which elements from the library
- maintain a historical account of element and library transactions

For details see DEC Module Management System manuals and DEC Code Management System manuals or the Bookreader or contact the OpenVMS Advisory Service (see names on page 5).

7.10 OpenVMS Routines

7.10.1 System Services

OpenVMS provides a large number of system routines for

- Event flag handling.
- Asynchronous system trap handling (AST).
- Logical name handling.
- Input/output
- Process control.
- Time handling.
- Condition (error) handling.
- Memory management.
- Lock manager.

You get a full list and short description by `HELP SYSTEM`. Note, that all routines begin with SYS$... In the HELP, however, they are listed beginning with $... Again, the call of these routines may be inserted by the `LSEDIT` F8 key.

7.10.2 Run Time Library Routines

OpenVMS provides a large number of runtime library (RTL) routines for various purposes.

- General library routines (LIB$...).
- Screen handling routines (SMG$...).
- Mathematical routines (MTH$...).
- String handling routines (STR$...).

You get a full list and short description by `HELP RTL`. Again, the call of these routines may be inserted by the `LSEDIT`

7.10.3 Utility Routines

OpenVMS provides program interfaces to several utilities.

- Command language routines (CLI$...).
- File definition routines (FDL$...).
- Library access routines (LBR$...).
- Sort/Merge routines (SOR$...).
- Text processing routines (TPU$...).

For these routines there is still no interactive HELP, but only a manual.

Chapter 8

DCL Procedures

DCL procedures are files containing DCL commands. A DCL procedure can be executed in batch job, or in the current process, or at a remote node.

```
$ @file                     ! Execute interactively.
$ @file /OUTPUT=outfile     ! Execute interactively and write all output
                            !  into the file named 'outfile'
$ SUBMIT/NOPRINT file       ! Submit for batch execution and write all output
                            !  into the log-file named 'file.LOG' under SYS$LOGIN
```

8.1 Command Procedure Format

- Each command line must begin with a dollar sign. Lines without a dollar sign are data lines. To span a command line over several text lines, end with an hyphen and continue next line **without** dollar sign.
- Comments begin with exclamation point, e.g.

```
$! This is a comment
$ DIRECTORY [USER.TEXT]     ! This is a comment after a DCL command
```

- DCL commands should be written in full length to provide clarity.
- Labels are terminated by a colon, e.g..

```
$ GOTO ALPHA
$ ALPHA:
```

- Besides the DCL commands used interactively, there are some used only in procedures:

```
   $ GOTO                ! Go to label
   $ CALL label          ! Subroutine call
   $ label: SUBROUTINE   ! Subroutine
   $ IF - THEN
     ELSE - ENDIF        ! Conditional execution
   $ INQUIRE             ! Prompt for a string
   $ ON - THEN           ! Control conditions
   $ ON ERROR THEN       ! Control error conditions
   $ ON CONTROL_Y THEN   ! Control Ctrl Y conditions
   $ SET [NO]ON          ! Enable (default) or disable error conditions
   $ SET VERIFY          ! Output commands lines during execution
   $ SET NOVERIFY        ! Do not output commands lines (default)
   $ SYNCHRONIZE         ! Synchronize batch jobs

! open an existing file for input and create logical name
   $ OPEN logname file.type

! create a new file for output and create logical name
   $ OPEN/WRITE logname file.type

! open an existing file for input and output at the start of the file
! and create logical name
   $ OPEN/READ/WRITE logname file.type

! open an existing file for output at the end of the file
! and create logical name
   $ OPEN/APPEND/WRITE logname file.type

! close a file, this is NOT done automatically when you leave
! the command procedure!!
! You cannot open the same file again if you have not closed it before
   $ CLOSE logname

   $ READ logname        ! Read from file
   $ WRITE logname       ! Write to file
   $ EXIT                ! leave procedure
```

8.2 Command Procedure Variables (Symbols)

You should be aware that DCL is a text interpreter. Therefore DCL does not distinguish between string and numeric variables. Both are represented as symbols. Expressions may be mixed up

with symbols having a numeric value and others having a string value, but the result will be unpredictable. String values are enclosed in quotation marks (""). If a symbol is enclosed in apostrophes, it is replaced by its value before the line is processed further. This is important because one can use symbols for very tricky operations. In expressions, or as arguments in lexical functions (see below) the apostrophes are not allowed.

There are **global** and **local** symbols. Local symbols are known only at the procedure level where they are created, whereas global symbols are known to all levels. Global symbols are deleted only by `DELETE/SYMBOL/GLOBAL` or by logoff. You should use only local symbols in DCL procedures.

We try to show with some examples the usage of symbols:

```
$ x1 = "ABC"            ! assign a string to a local symbol
$ x1 = "A""BC"          ! x1 has now the value A"BC
$ x1 = "ABC" + "XYZ"    ! x1 has now the value ABCXYZ
$ x1 = x1 - "A"         ! x1 has now the value BCXYZ
$ x1 = x1 - "X"         ! x1 has now the value BCYZ (one X removed)
$ x1 = 1 + 4            ! x1 has the value 5
$ x1 = "1" + "4"        ! x1 has the value 14
$ y = "ABC" + x1        ! y has the value ABC14
$ 'y' = "X"             ! symbol ABC14 has the value X
$ COPY 'y'.TXT 'y'.TEXT ! copy file ABC14.TXT to ABC14.TEXT
$ x = 5
$ y = 7
$ z = x * y             ! y has the value 35
$ IF x .EQ. y THEN EXIT ! false, do not exit
$ IF x .LT. y           ! nested IF clause
$  THEN
$  ELSE
$ ENDIF                 ! end of IF clause
$ CALL ADAM    ! call the subroutine ADAM
$ ADAM: SUBROUTINE      ! begin of subroutine ADAM
$ ENDSUBROUTINE         ! end of subroutine ADAM
$ r = x .NE. y          ! r has the value 1 (true)
$ IF r THEN GOTO hell   ! true, go to hell
$ IF "A" .LTS. "B" THEN ! true, because A is lower than B
$ x = "r = ''r'"        ! x has the value r = 1 (Note the two '')
$ x = "r = ""''r'"""    ! x has the value r = "1"
$ y[0,8] = 27        ! the first 8 bits are set to 27, the escape character
```

8.3 Control Statements

Besides the error control statements (see below) there are only four other control statements:

```
$ IF expression THEN command ! execute command only if expression true

$ IF                         ! Conditional execution
$ THEN
$ ELSE
$ ENDIF

$ GOTO label                 ! Proceed at label
$ label:                     ! label

$ GOSUB ROUT                 ! Proceed at label ROUT, but return
$ ROUT:                      ! label
$ RETURN                     ! return to statement behind GOSUB

$ CALL label                 ! Subroutine call
$ label: SUBROUTINE          ! Subroutine
```

8.4 Terminal I/O

After login, there are the logical names defined already

```
SYS$INPUT          ! data input device (terminal)
                   ! In batch or procedures the file
SYS$OUTPUT         ! output device (terminal)
SYS$ERROR          ! output device (terminal)
                   ! both are in batch jobs the log file
SYS$COMMAND        ! command input device (terminal)
                   ! in batch defined as disk
```

To write a line on the terminal use

```
$ WRITE SYS$OUTPUT "any text"
$ WRITE SYS$OUTPUT symbol
$ WRITE SYS$OUTPUT "any text and ''symbol'"
```

To read a line from the terminal into any symbol use

```
$ INQUIRE symbol "any prompt text"
```

The string defined with the INQUIRE statement will be typed on the terminal first (prompt string) followed by a colon ":". If you want to suppress the colon use INQUIRE/NOPUNCTATION.

DCL reads commands from SYS$COMMAND and writes output to SYS$OUTPUT or SYS$ERROR. Any prompts from programs are read from SYS$INPUT which is the DCL procedure file. You must provide the input as data lines behind the line calling the program. If you want the program to get the input from your terminal, add the following line **before** the line calling the program:

```
$ RUN program       ! program reads input from following lines
1234567
any text or numbers
$ DEFINE/USER SYS$INPUT SYS$COMMAND
$ RUN program       ! program reads input from terminal
$ EXIT
```

Of cause, it cannot make sense to execute such procedures in batch jobs, because the program cannot get an input.

8.5 Command Procedure Parameters

Calling a DCL procedure one may specify parameters. These are separated by spaces or (Tab) characters. They are converted to uppercase, unless they are enclosed in quotation marks. Inside the procedure these parameters are assigned to local symbols P1 through P8. They may be used as other symbols.

Because this is not a very comfortable parameter passing mechanism, we provide an interface which allows similar parameter lists as for DCL commands. In addition it generates a menu on request. The example shows how to use this interface. A more detailed description is found by HELP MDCLLIST.

```
$ descriptor = "PREF A=? B=DEV C "
$ qualifier  = "/SWI*TCH/VAL*UE=/DEF*AULT=XXX"
$ MDCLLIST_help = "Short description of the procedure."
$ MDCLLIST "''descriptor'" + "''qualifier'"
$ IF .NOT. $STATUS THEN EXIT
$! Now the following symbols are created
$! PREF_A has value of first positional parameter or is prompted
$! PREF_B has value of second positional parameter or DEV
$! PREF_C has value of third positional parameter or null string
$! PREF_SWITCH has value "/SWITCH" or null string
$! PREF_VALUE has value specified in argument list or null string
$! PREF_DEFAULT has value specified in argument list or XXX
$!
```

```
$! This routine may be called as
$! @file ?                              ! Enter menu
$! @file X                              ! 1 positional parameter
$! @file X Y/SWI                        ! 2 pos. par. and a qualifier
$! @file X /VAL=1
$! @file X "" Z /VAL=1/SWI /DEF="A B"! second pos.par is null string
```

It is strongly recommended to create a symbol for the DCL procedure call:

```
symbol == "@device:[directory]procedure"
```

The device and directory specification is required, because the procedure might be called from a different device/directory than the current. Called by a symbol the above procedure calls would look like DCL commands! **Before you create global symbols, check, if they are already in use** (command `SSYM symbol`).

8.6 Lexical Functions

The real strength of DCL are the **lexical functions**. These are functions beginning with F$. They are treated like symbols (but you cannot assign a value to them). We give a short overview and some examples here. You get information by `HELP LEXICAL`.

- **The following functions return information:**

```
F$CONTEXT         ! specifies selection criteria for F$PID use
F$CSID            ! returns cluster identification number
F$DEVICE          ! all specific devices
F$DIRECTORY()     ! current default directory
F$ENVIRONMENT     ! information about current environment
F$FILE_ATTRIBUTES ! file attributes
F$GETDVI          ! device information
F$GETJPI          ! process information
F$GETQUI          ! queue information
F$GETSYI          ! system parameters
F$MESSAGE         ! Text associated with message code
F$MODE()          ! execution mode (batch, interactive etc.)
F$PID             ! process ID's
F$PRIVILEGE       ! privileges of current process
F$PROCESS()       ! name of current process
F$SETPRV          ! returns state of queried privilege
F$TIME()          ! time as dd-mmm-yyyy hh:mm:ss.cc
F$TRNLNM          ! translate logical name equivalent
F$TYPE            ! type of a symbol, INTEGER or STRING
```

```
F$USER()          ! user identification code (UIC)
F$VERIFY          ! verification status
```

Examples:

```
$ old_default = F$DIRECTORY()  ! save directory
$ SET DEFAULT [X.Y]            ! set directory
$ SET DEFAULT 'old_default'    ! restore directory

$ IF F$MODE() .EQS. "BATCH" THEN SET VERIFY
                               ! set verification ON in batch only
$ IF F$MODE() .EQS. "BATCH" THEN -
  WRITE SYS$OUTPUT "Job executed at ''F$TIME()'"
                               ! write message to log file
$ WRITE SYS$OUTPUT "You are on terminal ''F$TRNLNM("SYS$COMMAND")'"
                               ! output terminal device
$ WRITE SYS$OUTPUT F$GETSYI("HW_NAME") ! type computer model name, e.g.
Digital 2100 Server Model A500MP
$ WRITE SYS$OUTPUT F$GETSYI("ARCH_NAME") ! type architecture name, e.g.
Alpha
$ IF F$GETSYI("ARCH_TYPE") .EQS. "2" THEN GOTO ALPHA_CODE
```

- **The following functions handle strings:**

```
F$EDIT            ! edit a string
F$ELEMENT         ! get elements out of a string separated by a delimiter
F$EXTRACT         ! extract a substring
F$FAO             ! format a string with arguments
F$LENGTH          ! length of a string
F$LOCATE          ! locate a substring
```

Examples:

```
$ x = "ABCDEFG"            ! assign value
$ i = F$LOCATE("C",x)      ! i is 2 (offset!)
$ l = F$LENGTH(x) - i      ! length minus offset
$ y = F$EXTRACT(i,l,x)     ! y is CDEFG

$ x = "A,B,C"
$ y = F$ELEMENT(0,",")     ! y is A, delimiter is comma
$ y = F$ELEMENT(3,",")     ! y is comma (no element of this number)
```

```
$ x = "   a    "
$ y = F$EDIT(x,"TRIM")     ! y is a
$ y = F$EDIT(y,"UPCASE")   ! y is A
$ x = "a   a   a "         ! several spaces
$ y = F$EDIT(x,"COMPRESS") ! y is a a a (one space)

$ x = "This is !AS"        ! The !AS is a dummy for a string
$ y = F$FAO(x,"A")         ! y is This is A
```

- **The following functions convert data types:**

```
F$CVSI            ! extract bit fields and convert to integer
F$CVTIME          ! convert time to yyyy-mmm-dd hh:mm:ss.cc
                  ! this time format can be compared to another
F$CVUI            ! extract bit fields and convert to integer
F$IDENTIFIER      ! converts an UIC identifier
F$INTEGER         ! converts expression to integer
F$STRING          ! converts expression to string
```

- **The following functions handle file names:**

```
F$PARSE           ! parse file names, replace defaults
F$SEARCH          ! search a file, return full name
```

Examples:

```
$ file = "LOGIN"
$ def  = F$PARSE(file)            ! e.g. EE$ROOT:[GOOFY]LOGIN.;
$ def  = F$PARSE(file,".COM")     ! e.g. EE$ROOT:[GOOFY]LOGIN.COM;
$ dir  = F$PARSE(file,,,"DEVICE") ! e.g. EE$ROOT:
$ full = F$SEARCH(file)           ! e.g. null string (LOGIN is not found)
$ full = F$SEARCH("LOGIN.COM")    ! e.g. EE$ROOT:[GOOFY]LOGIN.COM;53
```

As you can see the file type and the version number are not defaulted.

8.7 Command Procedure Debugging

DCL itself allows only to set verification on (`SET VERIFY`) and to place `SHOW SYMBOL` statements in the procedure to find out what happens. At GSI there is a tool to generate a debug version of a procedure without modifying the run version:

```
$ MDCLANAL DEBUG procedure debugfile    ! generates debug version
```

After that, debugfile contains a version of the procedure showing all symbols. You may specify, if verification is switched off during procedure calls or not. This is very convenient, if you call other, already tested procedures.

8.8 Command Procedure Error Handling

You should provide generally one or more labels where errors are handled and one where breaks (Ctrl Y) are handled. You may close files, delete temporary files, give error explanations etc...

```
$ ON ERROR THEN     GOTO error_label
$ ON CONTROL_Y THEN GOTO break_label
$ SET NOON              ! disable error checking
$ SET ON                ! enable error checking
```

After execution of programs you may check the symbol $STATUS for a successful completion ($STATUS = 1), e.g.

```
$ IF $STATUS .NE. 1 THEN GOTO ERRORHANDLING
```

8.9 Read/Write Files in Command Procedures

It is very easy to create, read and write text files. The following examples show that. The parameter 'logname' is an automatically defined logical name used to connect read/write statements to open/close statements.

```
$ OPEN/WRITE/ERROR=label logname file  ! create new file
$ OPEN/APPEND/ERROR=label logname file ! open existing file for write
$ OPEN/READ/ERROR=label logname file   ! open existing file for read
$ READ/ERROR=label1/END_OF_FILE=label2 logname symbol
                                       ! read record to symbol
$ WRITE logname expression             ! write expression to record of file

! if file is NOT closed automatically when you leave the command procedure!!
!  you cannot open the same file again if you have not closed it before
$ CLOSE logname                        ! close that file
```

Please, specify generally labels to handle errors or end of files. Otherwise files would stay OPEN and cannot be deleted!

8.10 LSEDIT Support for Command Procedures

Some unhandy DCL constructs are generated by the LSEDIT editor. These constructs are inserted in the text typing one of the following names terminated by Ctrl E

```
IF         ! IF statement
IF_ELSE    ! IF ELSE construct
```

```
IF_END      ! IF END construct
LOOP        ! Loop setup
TMP         ! unique temporary file name
```

Remember, that you have to replace 'placeholders' and 'tokens' marked by [] or {}, respectively. To do that, just type. Go to the next one by Ctrl N and replace it and so on.

Chapter 9

GSI Utilities

9.1 Miscellanea

There are some useful utilities written at GSI. Use the HELP command to get complete descriptions.

CBATCH executes a DCL command line as a batch job.

CREPEAT repeats a DCL command line till Ctrl Y.

CREPLACE replace one string by another in several files.

CDIFFER Compares sets of files and outputs which files are different.

ECLINE repeat a DCL command line using a set of values for a placeholder. Note that the command line must be enclosed in quotation marks. There **must** be a space between the command and the first . Examples:

```
$ ECL "DIR $$1" /$$1=*.PPL /1DIR
        ! replace the $$1 by all filenames matching *.PPL (full name)
$ ECL "DIR $$1" /$$1=*.PPL /1DIR/1SH
        ! replace the $$1 by all filenames matching *.PPL (short name)
$ ECL "DIR $$1" /$$1=A,B,C
        ! replace $$1 by A, then B, then C
$ ECL "LIB/EXTR=$$1/OUT=$$1.TXT TPRIV" /$$1=* /1LIB=TPRIV
        ! extract all modules from TPRIV to different files.
$ ECL "DIR $$1" /$$1=@file
        ! replace $$1 by the lines of the file (indirect list).
$ ECL "DIR $$1" /$$1=@file /LIST
        ! Do NOT execute, but list the command lines only
```

It is recommended to use always first the /LIST qualifier to check if the ECLINE command works correctly. The command is very powerful and may therefore produce a lot of junk, if an error was in the command line.

MLOCKS shows all defined resource locks. The program will prompt you with:

```
SUC: MLOCKS>
```

To enter the command menu type in

```
SUC: MLOCKS> $ MENU
```

NWDCL executes a DCL command line at a remote node.

SSEC shows Global Sections of your node. If you just type in `SSEC` you will see all Global Sections (e.g. GOOSY Data Bases) of your own user group on this Alpha AXP or VAX. You may give as an argument any string. This string will be searched in the total list of all Global Sections defined on your node, e.g.:

```
$ SSEC GRP                ! Shows all Group Global Sections (no system)
```

The command `HELP @UTILITY DIRECTORY` outputs a list of all GSI specific command procedures.

9.2 Documentation

The GOOSY documentation system is easy to use and produces a maximum of information by a minimum of effort. It has two components, a **generator** and a **formatter**.

- The generator prompts for information and formats a documentation header as language dependent comment block. It supports C, PL/I, FORTRAN, PASCAL, DCL, TSO and Alpha AXP or VAX assembler.
- The formatter extracts this documentation and produces optional output for SCRIPT, TEX, Alpha AXP or VAX printer, or OpenVMS HELP.

Related commands are:

CEDIT generates the documentation.

GLDOCUMENT generates the output for TEX, SCRIPT, or HELP.

TDOCUMENT processes the TEX files and outputs DVI files.

DVIPRI outputs DVI files to printer.

XDVI outputs DVI files to a DECwindows/Motif window.

LIBRARY/REPLACE inserts HLP files to a help library.

The formats of the documentation headers is described in `HELP DOCUMENT`.

9.3 GOOSY Program Library

Many general purpose routines have been written for GOOSY. They may be of interest for other programmers as well. All these routines begin with `U$...`. Get on overview by `HELP @MODULE DIRECTORY` or in the GOOSY manual 'Overview'. A detailed description can be obtained by `HELP @MODULE routine`. In an `LSEDIT` session you can include the complete routine call by pressing the F8 key and entering the routine name. The PL/I declaration is done by the statement:

```
@INCLUDE $MACRO(routine);
```

The modules are automatically linked with your program.

9.4 Preparing and Printing Documents

9.4.1 TeX and LaTeX

To prepare documents, the use of LaTeX is recommended for all types of printed material. One has to use LaTeX in conjunction with an editor, in your case it should be the Language Sensitive Editor (`LSEDIT`) (see chapter 7 on page 65).

Four steps are required to bring a document to paper.

1. Create the desired text and codes on the Language Sensitive Editor.
2. Check the text for spelling mistakes using `SPELL` command after Do.
3. "Compile" the text using the LaTeX compiler. For German text use the LaTeX D compiler to get the correct hyphenation.
4. Print the text on the laser printer.

You will now be shown how to proceed through all steps to create a small document named `FIRSTTRY.TEX`:

You first have to invoke `LSEDIT` and then enter your text:

```
$ LSEDIT FIRSTTRY.TEX
or just
$ LSE FIRSTTRY.TEX
```

The screen is cleared and you receive a prompt in the upper left corner. You are now in `LSE` mode and can enter your text:

```
\documentstyle{article}
\begin{document}
This is my first try at \LaTeX~.
\end{document}
```

You have now created your document and now you have to "check" the spelling. Hit the Do key and give the LSE command SPELL.

For the SPELL utility all LaTeX control string are unknown, so ignore them once with the I key. Do not use the A key because it would enter wrong spelling into the general directory of SPELL. After your text was check and corrected leave LSEDIT with Do Exit command or hit just GOLD Remove. Then call LaTeX by typing

```
$ LATEX FIRSTTRY.TEX
```

After the compilation is done you will again see your DCL Prompt ($) and now can proceed to the printing.

```
$ DVIPRI FIRSTTRY
```

Now you can go to the laser printer and pick up your document. (For directions to the laser printers see appendix D on page 109.)

You may preview the TeX output on your DECwindows/Motif screen using the DCL command

```
$ XDVI FIRSTTRY
```

If you wish to expand or change the document you may do so by returning back to LSE with the command LSE and change or edit the file. After you are done you must compile the file again by the above mentioned procedure.

If you do not want to save the created text you can enter QUIT when the LSE Command> prompt is given. This brings you into the DCL level without saving the file.

To see the files created you can enter the DCL command DIR. You will see all your files on your current directory. When you enter LSE you just have to enter the name and the extension of the file. The version number can be omitted since the highest version number will be taken automatically.

9.4.2 Expanding on LaTeX

Now that you have gained an insight to LaTeX , your horizons can be expanded, by giving you more detailed information about the preparation of documents.

The following pages are the output of the SMALL.TEX file. Compare the pages to the printed document to see how it was done. You can print the document by first "compiling" it with LATEX TEX$INPUTS:SMALL.TEX and then printing it by entering DVIPRI SMALL.

```
% THIS IS A COMMENT
% SMALL.TEX -- Released 5 July 1985
% USE THIS FILE AS A MODEL FOR MAKING YOUR OWN LaTeX INPUT FILE.
% EVERYTHING TO THE RIGHT OF A  %  IS A REMARK TO YOU AND IS IGNORED BY LaTeX.

% THE FILE  /usr/local/lib/tex82/local.gid  TELLS HOW TO RUN LaTeX.

% WARNING!  DO NOT TYPE ANY OF THE FOLLOWING 10 CHARACTERS EXCEPT AS DIRECTED:
%                &   $   #   %   _   {   }   ^   ~   \

\documentstyle{article}        % YOUR INPUT FILE MUST CONTAIN THESE TWO LINES
\begin{document}               % PLUS THE \end COMMAND AT THE END.


\section{Simple Text}          % THIS COMMAND MAKES A SECTION TITLE.

Words are separated by one or more spaces.  Paragraphs are separated by
one or more blank lines.  The output is not affected by adding extra
spaces or extra blank lines to the input file.

Double quotes are typed like this: ``quoted text''.
Single quotes are typed like this: `single-quoted text'.

Long dashes are typed as three dash characters---like this.

Italic text is typed like this: {\em this is italic text}.
Bold   text is typed like this: {\bf this is  bold  text}.

\subsection{A Warning or Two}        % THIS COMMAND MAKES A SUBSECTION TITLE.

If you get too much space after a mid-sentence period---abbreviations
like etc.\ are the common culprits)---then type a backslash followed by
a space after the period, as in this sentence.

Remember, don't type the 10 special characters (such as dollar sign and
backslash) except as directed!  The following seven are printed by
typing a backslash in front of them:  \$  \&  \#  \%  \_  \{  and  \}.
The manual tells how to make other symbols.

\end{document}                    % THE INPUT FILE ENDS WITH THIS COMMAND.
```

For more information about LaTeX turn to:

- *LaTeX User's Guide & Reference Manual* by L. Lamport
- or call Volker Schaa, tel. 340, room 1.158

You can print a document example by entering the command lines:

```
$ LATEX TEX$INPUTS:SAMPLE
$ DVIPRI SAMPLE
```

9.4.3 DECwrite

(Please do not use DECpresent any longer. It is retired and replaced by DECwrite). DECwrite is only running under DECwindows/Motif. If you have connected your session to another node via `SET HOST` or `TELNET` you must connect the display you are working on to the other node with the DCL command

```
$ SET DISPLAY /CREATE /NODE=node-name /TRANSPORT=transport
.e.g.
$ SET DISPLAY /CREATE /NODE=VSCN
```

where `node-name` is the name of the node to which the display belongs to and `transport` is either DECNET (the default), or TCPIP or LAT.

DECwrite is a very flexible documentation tool. This WYSIWYG (what you see is what you get) editor and composer runs in an English and a German version (select the language within the Session Manager Customize menu Languages... option). DECwrite allows the creation, editing and composing of text, graphics, and images on all levels of documentation. To include image data you may scan drawings with a scanner (e.g. the black/white scanner `SCAN01` located in the lab of the DVEE department room 2.249; for other scanners see appendix D on page 109; select the scanner device using first from the Commands menu the Scan.. option and then from the new window's Commands menu the Selects Scanner... option and set the Scanner: field to e.g. `SCAN01`, then from the Options menu the Save Settings option to store the scanner device for later use) using the DCL command `"$ IMAGE INTER"` from a DECwindows/Motif session. The data are sent automatically via Ethernet to your host computer. DECwrite includes color handling for characters and graphic.

For the first usage of DECwrite set the configuration of the systems selecting within the Options menu (in German Anpassen) the Preferences... option (in German Voreinstellung...). Specifically set Editing Keyboard (in German Tastatur) to EDT-like by selecting the corresponding widget with the cursor and press the right mouse button MB1, select EDT and then release MB1. You should also set Menu Type Full instead of Short (in German Menütyp Vollständig). Save the settings with the Customize menu Save Current Settings option.

The documentation on DECwrite and DECimage Scan exist as manuals and is available with HELP and the Bookreader utility.

Appendix A

Using a Terminal and Editing Command Lines

On the Alpha AXP or VAX terminal all input can be typed ahead, even if the CPU does not echo. The type ahead buffer allows 80 characters to be typed ahead.

The following keys allow you to edit the current DCL command line (and the command lines of most utilities). For some keys to work, the `SET TERMINAL/LINE_EDITING` command must be in effect, which is the start default for all terminals at GSI. (Enter the `SHOW TERMINAL` command to display your terminal's attributes.)

F12 **or** Ctrl H **or** BACKSPACE **or** BS — Moves the cursor to the beginning of the line.

F14 **or** Ctrl A — Changes between `SET TERMINAL/OVERSTRIKE` and `SET TERMINAL/INSERT`. The start default for all terminals at GSI is the insert mode.

Ctrl E — Moves the cursor to the end of the line.

Ctrl R — Reshows the current command line.

Ctrl X — Clears the typeahead buffer.

Ctrl U — Deletes all characters to the left of the cursor.

<X — Deletes one character to the left of the cursor, moving the cursor one space to the left.

F13 **or** Line Feed **or** LF **or** Ctrl J — Deletes the word to the left of the cursor.

⇒ **or** Ctrl F — Moves the cursor one character right.

⇐ **or** Ctrl D — Moves the cursor one character left.

⇑ **or** Ctrl B — Recalls the previously given DCL command (the last 20 commands can be recalled in this way).

`RECALL/ALL` — Show the last 20 commands.

`RECALL n` — Gets the n^{th} command.

`RECALL xyz` — Gets the last command starting with the characters `xyz`.

⇓ — Recalls the DCL command entered after the current command.

Ctrl O — Stops and starts the screen output. The execution of a program or command procedure will *not* be stop. Any error output or the DCL prompt will automatically switch on the output again.

F1 **or** HOLD — Stops and starts the screen output. The execution of a program or command procedure will be stopped waiting to continue with the screen output. If your terminal will not respond of any key stroke, try the F1 key on your terminal. This key is active for each window under DECwindows/Motif individually.

Ctrl S — Stops the screen output. The execution of a program or command procedure will be stop waiting to continue with the screen output. Restart output by typing Ctrl Q. This key is active for each window under DECwindows/Motif individually.

Ctrl Q — Starts the screen output previously stopped by Ctrl S. The execution of a program or command procedure will be stop waiting to continue with the screen output. This key is active for each window under DECwindows/Motif individually.

Ctrl W — Rewrites (refreshes) the whole screen within several utilities using screen mode, like `LSE` or `DEBUG`.

To write special characters like an umlaut you must compose a sequence. Depending on the keyboard and the terminal press and the release the Compose key only or simultaneously press and then release the Compose key and the space bar, and then type the two characters separately which are needed for the special character. In the following is a brief list of composed characters:

ä = "a, ö = "o, ü = "u, Ä = "A, ß = ss, ÿ = "y, á = a', à = a‘, â = a^ , ç = c, å = a*, é = e', è = e‘, ï = i", ø= o/, Ø= O/, æ= a, Æ= AE, ñ = n , ©= co, $\underline{a}$ = a_, £= L-, §= s!, ± = +-, ≪ = <<, ≫ = >>, $\frac{1}{2}$ = 12, $\frac{1}{4}$ = 14, μ = /u, ^{0} = 0^ , ^{1} = 1^ , ^{2} = 2^ , ^{3} = 3^ .

There are complete tables of available characters. In DECwrite you have much more special characters including Greek in the 'Text → Special' menu.

While using the terminal, you can have different sessions of the Terminal Server. Once you have started the first session (by logging in), you can "break" out of your session by pressing F5 (or Ctrl F5 and Return on a PECAD). After the `S200xx Local>` prompt connect to any Alpha AXP or VAX or to the IBM by typing `CONNECT service` and then simply log in to the chosen service. The number of simultaneous sessions is limited by default to 4. You can have only

two sessions if you use the multisession option together with a VT330, VT340, or VT420 terminal.

You can move through your established sessions by pressing Ctrl \, or by breaking out to the Terminal Server local mode with key F5 (or Ctrl F5 and Return on a PECAD) and then using the following Local> commands: FORWARD or BACKWARD.

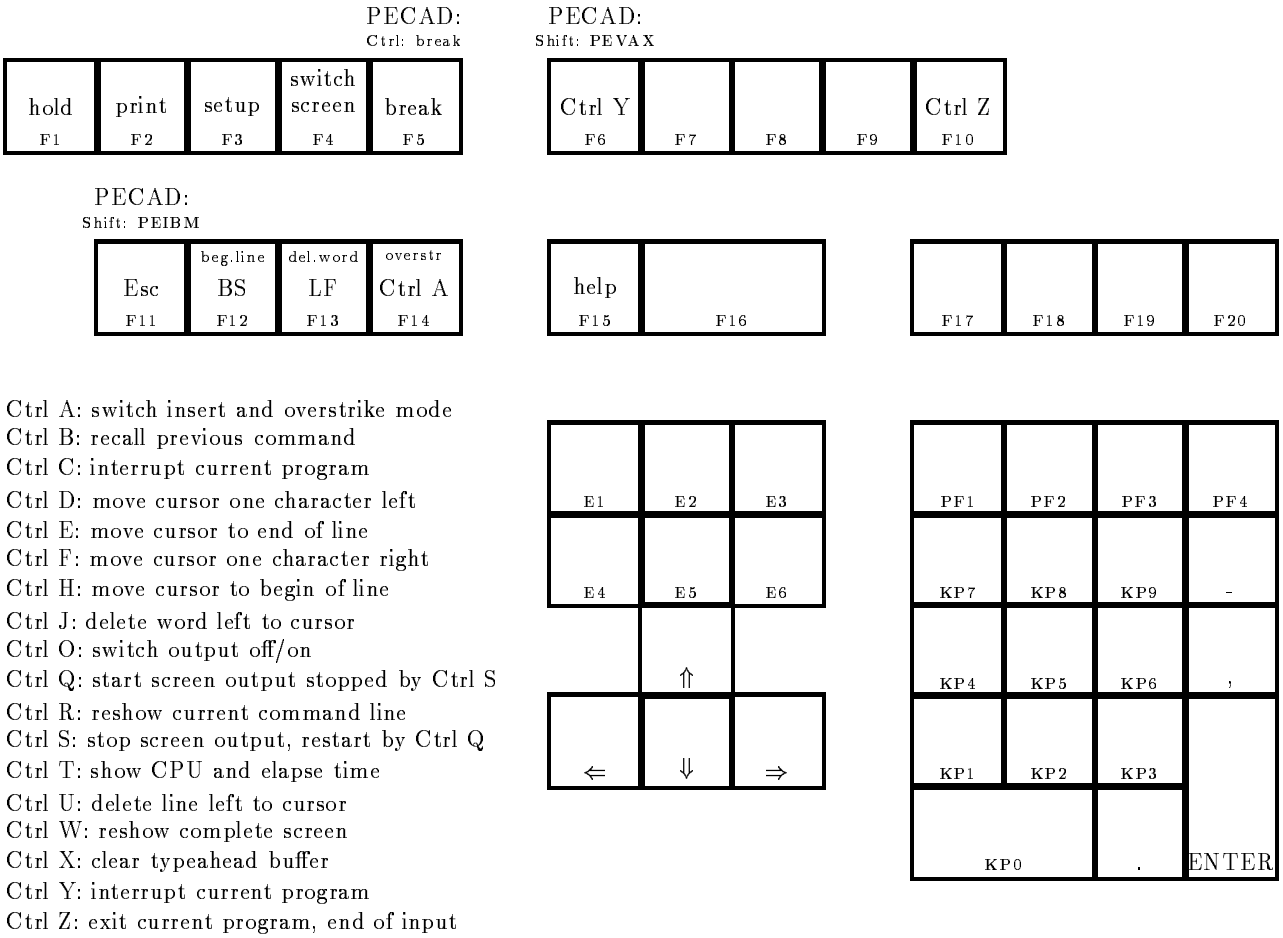
If you want to use a terminal connected to a terminal server as a graphics output device under OpenVMS (device separation) you have to get its terminal line number. To do so login at the graphics terminal to the S200xx Local> terminal server mode by hitting the Return key. Then type in the server commands

```
S200xx Local> SHOW SERVER
S200xx Local> SHOW PORT
S200xx Local> LOGOUT
```

The server information shows you the device number range LTAxxx-LTAyyy of that server. The SHOW PORT gives you the port number. Now combine them in selecting the last number of LTAxxx to be the port number. E.g. port number 5 and range LTA231-LTA238 defines the terminal line number of your graphics terminal to be LTA235:. The LOGOUT command frees the port for remote access from the Alpha AXP or VAX. Now you can access this terminal from another Alpha AXP or VAX terminal using this line number, e.g.

```
$ ALLOCATE LTA235:            ! reserve terminal line for your own usage
$ COPY file.type LTA235:      ! write something on the terminal
$ DEALLOCATE LTA235:          ! free the terminal line later
```

In the figure A.1 on page 102 you see the OpenVMS DCL keypad layout for line editing.



Ctrl A: switch insert and overstrike mode
Ctrl B: recall previous command
Ctrl C: interrupt current program
Ctrl D: move cursor one character left
Ctrl E: move cursor to end of line
Ctrl F: move cursor one character right
Ctrl H: move cursor to begin of line
Ctrl J: delete word left to cursor
Ctrl O: switch output off/on
Ctrl Q: start screen output stopped by Ctrl S
Ctrl R: reshow current command line
Ctrl S: stop screen output, restart by Ctrl Q
Ctrl T: show CPU and elapse time
Ctrl U: delete line left to cursor
Ctrl W: reshow complete screen
Ctrl X: clear typeahead buffer
Ctrl Y: interrupt current program
Ctrl Z: exit current program, end of input

Figure A.1: The Special Keypad Layout for DCL Level (Line Editing).

Appendix B

Login Command Procedure

A user's login command procedure will be executed at the login time of the user. The user might define logical names or DCL symbols in this login procedure. The file must have the name `LOGIN.COM` and it must be located in the so called login directory of the user where the default is set to after the login. Normally this directory has the same name as the username. You can get this name by the translation of the logical name `SYS$LOGIN`, using the command:

```
$ SHOW LOGICAL SYS$LOGIN
   or just
$ SLOG SYS$LOGIN
```

The following listing is an example of such a login command procedure file. It is located under `GOO$EXE:USER_LOGIN.COM`. A new user might copy this file to his own login directory and edit it to his own needs.

```
$ SET NOVERIFY
$ SET NOON
$! Define names for text library:
$ DEFINE/JOB PLI$LIBRARY SYS$LOGIN:privlib.TLB
$ DEFINE/JOB tpriv SYS$LOGIN:privlib.TLB
$ DEFINE/JOB LNK$LIBRARY SYS$LOGIN:privlib.OLB
$ DEFINE/JOB opriv SYS$LOGIN:privlib.OLB
$ DEFINE/JOB HLP$LIBRARY SYS$LOGIN:privlib.HLB
$ DEFINE/JOB hpriv SYS$LOGIN:privlib.HLB
$!
$! Set DCL function keys (Press PF2 for help):
$ PFKEY
$!
$! Set system prompt to "node:user$ ":
$ @GOO$EXE:SETPROMPT.COM
$!
```

```
$!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
$! The following lines are for all Utilities writen at GSI !!!!!
$!
$ TOOLLOGIN
$!
$!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
$! The following lines are for Public Software, like GNU and WWW !!!!!
$!
$ PUBLICLOGIN
$!
$!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
$! The following lines are for Software written by CERN, e.g. PAW  !!!!!
$! the parameter might be PRO, NEW or OLD depending on the version !!!!!
$!
$ CERNLOGIN NEW
$!
$!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
$! The following lines are for the TeX and LaTeX software !!!!!
$!
$ NEWTEX
$ NEWLATEX
$!
$!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
$! The following lines are for GOOSY users, only !!!!!
$!
$!  Define your global goosytable >LNM$GOOSY<
$!  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
$   set message/nofac/nosev/notext/noident
$   create/name_table LNM$GOOSY
$   set message/fac/sev/text/ident
$!  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
$!
$! Define names for profiles:
$ DEFINE/JOB GOO$PROFILE GOO$EXE:PROFILE.PROF
$ DEFINE/JOB GOO$INI_ALL GOO$EXE:INI_ALL.COM
$ DEFINE/JOB GOO$INI_TPO GOO$EXE:INI_TPO.COM
$!
$! Define all GOOSY stuff:
$ @GOO$EXE:GOOLOG.COM
$! Establish a data base for analysis control:
$ GOOCONTROL
$! Set GOOSY message output to readable format:
```

```
$ SETMES GOOSY /NOHEAD/NOPREF
$!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
$!
$! Add here user specific statements:
$!
$ IF F$MODE() .NES. "INTERACTIVE" THEN GOTO G_BATCH
$!
$! Add here statements to execute interactively only:
$!
$ GOTO G_FINISH
$!
$ G_BATCH:
$ IF F$MODE() .NES. "BATCH" THEN EXIT
$ WRITE SYS$OUTPUT "*********** Starting user batch procedure *****************"
$!
$! Add here statements to execute in batch only:
$!
$ SET VERIFY
$ G_FINISH:
$ EXIT
```

Appendix C

Standard File Types

Standard file types used at GSI are:

ANA SCA analysis file

ASM M68020 assembler source

C C language source

CP C language source for M68020

CXX C++ language source

COM DCL command procedures

DAT Data file

DIR Directory (NEVER use this type)

DOC DECwrite document file

EX20 M68020 exe files

EXE Executable image file (program)

FOR FORTRAN program source file

HLP Text file of a member of a help library

HLB Help library

IMG Image (scanned) document file

LIS Listing created by compilers (default for PRINT and TYPE)

LOG Output from batch jobs or GOOSY protocol file.

OBJ Object code file, output from compilers (input for linker).

OLB Object library

PCA PCA analysis file

PLI PL/I program source file (without preprocessor)

PPL PL/I program source file (for preprocessor)

PS PostScript file

TXT Text file written by user (input for text library)

TLB Text library

TMP Temporary file. Should be deleted each day.

ULB M68020 user libraries

SCR Script source file

SREC M68020 srec files

TEX TEX source file

Others are DIF, DIS, DMP, EDT, FDL, INI, JNL, JOU, MAI, MAP, MAR, MEM, MLB, MSG, OPT, PAR, RNO, STB, SYS, TEC, TJL, TPU, UPD, U, O, OA.

Appendix D

Printers and Scanners

D.1 Printers at GSI

To print a file on a laser printer there is only one command for OpenVMS and UNIX the `POP` command.

The `POP` command has the following structure:

```
POP   -<print queue selector> [-o <pass through options>] <file>

    Print the file <file> on the printer and with the style selected
    with <print queue selector>. <pass through options> are platform
    specific parameters passed through the print server.

POP   -C    <Job Id> [-<print queue selector>]

    Cancel the print job with the job id <Job Id> on the print queue
    selected by <print queue selector>.

POP   -Q -<print queue selector>

    Query the status of the print queue selected by
    <print queue selector>.

POP   -H

    Display the list of printers with their style options and locations
    and give a help for printing.

POP   -?     [-<print queue selector>]
```

```
    Display a syntax help of the print command.

Examples:
    $ POP -P01GPS file.PS
    prints file.PS on Printer P01 in PostScript format

    $ POP -Q -p08
    shows all entries in the default print queue of the printer p08
```

The print queue names (selector) define the printer device as well as the printing style and format. If the style or the printer selector are omitted default values are used. They have the following structure:

```
 print queue (name) selector :== p[nn[xxx]]

 p:      stand for print command
 nn :    two digits to select the printer device
 xxx:    three characters to select the print style
```

Depending on the platform and the printer device type you can add specific options to your print command. The available options are the qualifiers of the DCL PRINT command. Type HELP PRINT to get a list of qualifiers.

```
PostScript printers with DECprint Supervisor software will accept the

/PARAMETERS option.
  /PARAMETERS=INPUT_TRAY=tray-name
  Selects the input tray that provides paper for the job -
  valid tray names: top, middle, bottom.
  /PARAMETERS=PAGE_ORIENTATION=logical-orientation
  Specifies the orientation of printed output on the logical page.
  valid orientation: landscape, portrait.

If you want to apply a special form use the option /FORM=form.
Available forms:

  CPS$DEFAULT (stock=DEFAULT)          DCPS$DEFAULT (stock=DEFAULT)
  DEFAULT                              LAKP
  LN03_TEK (stock=DEFAULT)             LN03_TEXT_100 (stock=DEFAULT)
  LN03_TEXT_132 (stock=DEFAULT)        LN03_TEXT_80 (stock=DEFAULT)
```

```
  LN03_TEXT_80F (stock=DEFAULT)         LPS$$FORM (stock=DEFAULT)
  PCL (stock=DEFAULT)                   PS_PLAIN (stock=PLAIN_PAPER)

If you need additional fonts use the option /SETUP=(font,font,...)
Available additional fonts:

  BOOKMAN_DEMI          HELVETICA_NARROW              PALATINO_BOLD
  BOOKMAN_DEMIITALIC    HELVETICA_NARROW_BOLD         PALATINO_BOLDITALIC
  BOOKMAN_LIGHT         HELVETICA_NARROW_BOLDOBLIQUE  PALATINO_ITALIC
  BOOKMAN_LIGHTITALIC   HELVETICA_NARROW_OBLIQUE      PALATINO_ROMAN
  PRESENT_BULLETS       ZAPFCHANCERY_MEDIUMITALIC     ZAPFDINGBATS

other options are:

/COPIES=n               ! for n copies of the same file
/NOFLAG                 ! for deleting the flag page of the output (good for /copies=n)
/HEAD                   ! for deleting the flag page of the output (good for /copies=n)
/PARAMETER=(p1,p2,..)   ! for printer specific options, e.g. for the PrintServers 20 and 17
/PARAMETER=(INPUT=TOP) ! for getting paper from the top input tray (PS A to G)
/PARAMETER=(OUTPUT=FACE_UP) ! for putting paper to the lower output tray (PS B to G)
```

The print styles depend on the selected printer device. They have specific characters to be chosen in the printer (names) selectors:

```
  FCS = transparency, color postscript
  GAD = gray  paper, ASCII only, double sided
  GAQ = gray  paper, ASCII only, double sided and two-on-one page
  GAS = gray  paper, ASCII only, single sided
  GDS = gray  paper, DDIF, single sided
  GND = gray  paper, PostScript only, double sided
  GNQ = gray  paper, PostScript only, double sided and two-on-one page
  GNS = gray  paper, PostScript only, single sided
  GPD = gray  paper, ASCII/PostScript, double sided
  GPS = gray  paper, ASCII/PostScript, single sided
  GRS = gray  paper, REGIS, single sided
  GXS = gray  paper, Tektronix, single sided
  LNS = white paper, PostScript only, DIN A3, single sided
  PCS = white paper, PostScript only, DIN A0 Plotter, single sided
  WAD = white paper, ASCII only, double sided
  WAQ = white paper, ASCII only, double sided and two-on-one page
  WAS = white paper, ASCII only, single sided
  WCS = white paper, color PostScript, single sided
  WND = white paper, PostScript only, double sided
```

```
  WNS = white paper, PostScript only, single sided
  WNQ = white paper, PostScript only, double sided and two-on-one page
  WPS = white paper, ASCII/PostScript, single sided
  WPD = white paper, ASCII/PostScript, double sided
  WRS = white paper, REGIS, single sided
  WXS = white paper, Tektronix, single sided

Example:

 POP -P08WND -O /NOFLAG/COP=3 FILE.PS
   prints the PostScript file: files.ps on printer p08 on white paper without
   a flag page in 3 copies.
```

In the following the currently available printers are listed:

```
Printer Type Device          Server          Location
    Style
-------------------------------------------------------------------

P01  LaserJet 3Si                            Mittelspange Nordbau
    GPS,WPS,GPD,WPD

P02  LaserJet 4Si                            RZ Grafikraum
    GPS,WPS,GPD,WPD

P03  LaserJet 4Si                            Detektorlabor
    GPS,WPS,GPD,WPD

P04  LaserJet 4 Plus                         Lepton-Container C01 204
    GPS

P05  LaserJet 4Si                            BT Eingang A 2.Stock
    GPS,WPS,GPD,WPD

P06  Phaser III PXi                          I/O Room RZ 2.223
    WCS

P07  Color Jetprinter PS 4079                I/O Room RZ 2.223
    FCS

P08  PrintServer 20          PSAA::          I/O Room RZ 2.223
    WAS,WNS,WRS,WXS,WAD,WND,LNS,WAQ,WNQ
```

```
P09  PrintServer 17          PSAB::          1.222a
    GAS,GNS,WNS,GRS,GXS,GAD,GND,GAQ,GNQ,WAS

P10  PrintServer 17          PSAC::          Geschaeftsfuehrung
    GAS,GNS,WNS,GRS,GXS,GAD,GND,GAQ,GNQ,WAS

P11  PrintServer 17/600      PSAD::          Atomphysik vis-a-vis 2.294
    GAS,GNS,WNS,GRS,GXS,GAD,GND,GAQ,GNQ,WAS

P12  PrintServer 17/600      PSAE::          KPIII 4.174
    GAS,GNS,WNS,GRS,GXS,GAD,GND,GAQ,GNQ,WAS

P13  PrintServer 17/600      PSAF::          KPII vis-a-vis 4.141
    GAS,GNS,WNS,GRS,GXS,GAD,GND,GAQ,GNQ,WAS

P14  PrintServer 17/600      PSAG::          I/O Room RZ 2.223
    GAS,GNS,WNS,GRS,GXS,GAD,GND,GAQ,GNQ,WAS

P15  LN03R ScriptPrinter     LTA818:S200BS   DV/EE Lab 2.252
    GAS,GNS,GRS,GXS,GDS,GAQ,GNQ

P16  LN03R ScriptPrinter     LTA958:S200CG   KCI Lab 2.124
    GAS,GNS,GRS,GXS,GDS,GAQ,GNQ

P17  LN03R ScriptPrinter     LTA1368:S200DV  CAVEB Messhuette
    GAS,GNS,GRS,GXS,GDS,GAQ,GNQ

P18  LN03R ScriptPrinter     LTA1618:S200EU  CAVEA Messhuette
    GAS,GNS,GRS,GXS,GDS,GAQ,GNQ

P19  LN03R ScriptPrinter     LTA966:S200CH   HLI
    GAS,GNS,GRS,GXS,GDS,GAQ,GNQ

P20  LN03R ScriptPrinter     LTA1658:S200EY  Container Z2
    GAS,GNS,GRS,GXS,GDS,GAQ,GNQ

P21  LN03R ScriptPrinter     LTA598:S200AW   Prof. Metag 4.102
    GAS,GNS,GRS,GXS,GDS,GAQ,GNQ

P22  DEClaser 2250/plus      LTA508:S200AN   HKR 1.121
    GAS,GNS,GRS,GXS,GDS,GAQ,GNQ
```

```
P23  LN03R ScriptPrinter     LTA1748:S300FD  Cont.01 2046
    GAS,GNS,GRS,GXS,GDS,GAQ,GNQ

P24  LN03R ScriptPrinter     LTA1911:S700FN  Labor 2.103
    GAS,GNS,GRS,GXS,GDS,GAQ,GNQ

P25  LN03R ScriptPrinter     LTA1492:S200EI  Labor 3.001
    GAS,GNS,GRS,GXS,GDS,GAQ,GNQ

P26  LN03 / PLUS             LTA428:S200AF   DV/EE Lab 2.252
    GAS

P27  LN03 / PLUS             LTA408:S200AD   Mess-Station 1.124
    GAS

P28  LN03 / PLUS             LTA528:S200AP   KPII vis-a-vis 4.141
    GAS

P29  LN03 / PLUS             LTA438:S200AG   SHIP Villa Exp.-Halle
    GAS

P30  LN03 / PLUS             LTA998:S200CK   CAVEB Messhuette
    GAS

P31  LN03 / PLUS             LTA1028:S200CN  FRS Messhuette
    GAS

P32  LN03 / PLUS             LTA1058:S200CQ  ESR Messhuette
    GAS

P33  LN03 / PLUS             LTA1348:S200DU  KAOS Messhuette
    GAS

P34  LN03 / PLUS             LTA1468:S200EF  (Atomphysik+Bio) CAVEA Messhuette
    GAS

P35  LN03 / PLUS             LTA1568:S200EP  Sicherheit + Strahlenschutz
    GAS

P36  LN03 / PLUS             LTA1578:S200EQ  KPII 2. Stock hinter ESR
    GAS
```

```
P37  LN03 / PLUS             LTA457:S200AI   Prog.entw.Raum 2.117
     GAS

P38  LN03 / PLUS             LTA1458:S200EE  VAX-Rechnerraum 1.222a
     GAS

P39  InkJet LJ250            LTA488:S200AL   DV/EE Lab 2.252
     GAS

P40  Designjet 650C A0       LTA361:S200BD
     LNS

P41  Magicolor                               I/O Room RZ 2.223
     WCS,FCS

P42  LinePrinter             LTA668:S200BD   Room 1.107L
     GAS

P43  LN03 / PLUS             TTA3:           directly connected to VSAA Cave B
     GAS

P44  LN03 / PLUS             TTA3:           directly connected to VSBQ
     GAS

P45  LaserJet 4Si                            RZ I/O Room RZ 2.223
     GPS,WPS,GPD,WPD

P45  LaserJet 4Si                            RZ I/O Room RZ 2.223
     GPS,WPS,GPD,WPD

P46  LN03R ScriptPrinter     LTA658:S200BC   Ship Villa Exp.-Halle
     GAS,GNS,GRS,GXS,GDS,GAQ,GNQ

P47  LaserJet 4Si                            KPIII 4.174
     GPS,WPS,GPD,WPD

P48  Laser 1152              LTA643:S200BB   LSB #1, Rack 12
     GAS,GNS,GRS,GXS,GDS,GAQ,GNQ

P49  DesignJet 650C                          Pauserei 2.210
     PCS
```

You get help with `$ HELP PRINTER` and with `$ HELP POP`. The printer queue status can be obtained by command

```
$ SPRI
```

Each department is responsible for filling toner and new paper on their local LN03+ and LN03R printers. Toner and paper are available at the GSI stock. In case of problems call the the operators console tel. 2515 or the OpenVMS Advisory Service (see names on page 5).

D.2 Image Scanners at GSI

Pictures stored pixel-wise are called image data. You can produce and manipulate image data with an image *scanner* hardware and software and with the application program `PAINT`. A scanner scans a picture pixel-wise with a laser beam producing pixel image data.

To scan images call the DCL command `"$ IMAGE INTERACTIVE"` from a DECwindows/Motif session. The data are sent automatically via Ethernet to your host computer. Select the scanner device using first from the Commands menu the Scan.. option and then from the new window's Commands menu the Selects Scanner... option and set the Scanner: field to e.g. `SCAN01`, then from the Options menu the Save Settings option to store the scanner device for later use.

In the following the currently available printers are listed:

```
SCAN01       black/white, no half-tone, 300 dpi, DIN A4, El/Ex Lab, 2.252
SCAN02       color, half-tone, 300 dpi, DIN A4, 2.222a, tel: 2394
```

Appendix E

Batch Queues

At the Alpha AXP and the VAX VMSclusters (Nodes AXP601, AXP602 or V6000A, VSCN or all satellites AXP6xx or VSxx) there are among other queues several batch queues (this list is obtained by command `SHOW QUEUE /BATCH`):

```
Batch queue SYS$BATCH_node, on node::
e.g. Batch queue SYS$BATCH_AXP601, on AXP601::
or  Batch queue SYS$BATCH_VSCN, on VSCN::
similar for
Batch queue SYS$FAST_node, on node::
Batch queue SYS$TERM_node, on node::
Generic batch queue SYS$BATCH
Generic batch queue SYS$FAST
Generic batch queue SYS$TERM
```

The SYS$BATCH queues are limited to the low process priority 2. They should be used for compute intensive programs. Do not run such programs interactively!

The SYS$FAST queues are limited to the low process priority 3 and in addition to a limit of one hour CPU time.

The SYS$TERM queues run on interactive process priority 4 but with a limit of 20 minutes of CPU time.

Appendix F

Installed Software on OpenVMS

F.1 Software installed on OpenVMS VAX

The following software products are currently installed on the main OpenVMS VAX Cluster at GSI and is available on most VAXstations connected to this Cluster. A `(*)` marks all software not available on other VAXs. The DCL command to start the application is given in `[ ]`.

To all products documents and on-line Help are available. To most products a Bookreader documentation is available, too. For questions contact the OpenVMS Advisory Service (for names see on page 5).

DECwindows/Motif is available on most of the VAXstations. Some of the VAXstations are running the older VWS software which will not be described in this manual. For details about DECwindows/Motif refer to section 2.4.3 on page 19.

- Bookreader `[$ BOOK]`:

  If your are working directly on a VAXstation or via a Xwindow terminal on a VAXstation running DECwindows/Motif and only then a general utility is available to get all OpenVMS manuals on-line in windows on your screen.
  The manuals are stored on several CD-disks mounted on a centralized DEC InfoServer connected to Ethernet. This allows read access from all VAXstations at GSI to the same CD-disk drives.
  For details refer to section 3.2 on page 30.

- DECwrite `(*) [$ DECWRITE]`:

  DECwrite is a WYSIWYG (What You See Is What You Get) editor under DECwindows/Motif to compose complex documentations including text, graphics, and images previously scanned with DECimage Scan software. An English and German version is available

selected by the DECwindows/Motif Session Manager Language option. (Runs only under DECwindows/Motif. Bookreader documentation available)

- DECpresent `(*) [$ DECPRESENT]`: DECpresent is replaced by DECwrite.
- DECimage Scan `(*) [$ IMAGE INTER]`:

  Scans images in black/white or color with 300 dpi on one of the scanners installed at GSI, e.g. SCAN01 located in the Elektronik/Experimente laboratory room 2.249. For scanner locations see appendix D on page 109. The application can be started on any VAXstation running DECwindows/Motif. The data are transferred via Ethernet from the scanner to your VAXstation. The images can be manipulated like rotate, retouch, or cut. They can be used in DECwrite and DECpaint. (Runs only under DECwindows/Motif. Bookreader documentation available)

- DECimage Character Recognition Services `(*) [$ DCRS]`:

  Allows the interpretation of printed characters found on an image. The text image might by scanned before. (Runs only under DECwindows/Motif)

- LaTeX `(*) [$ LATEX or $ LATEXD and $ DVIPRI or $ NXDVI]`:

  Text written in TeX or LaTeX can be translated with the LaTeX compiler using the DCL command `$ LATEX file` for English text or `$ LATEXD file` for German text. The result can be transformed to laser printer format and printed with the `$ DVIPRI` command or to the DECwindows/Motif format to be previewed with the `$ NXDVI` command.

- SPELL `(*) [$ SPELL]`:

  There is a standalone spell checker utility for English written text files. Although, you should always use the `SPELL` command within `LSEDIT` for English text or the spell checker within `DECwrite` for English or German text.

- Topdrawer `(*) [$ TDRAW]`:

  Formatted data (as ASCII text tables) can be visualized on a graphics terminal or printed on a laser printer. For details see HELP TOPDRAWER.

- LSEDIT `[$ LSE]`:

  This DECset Language Sensitive text-editor supports various languages by specific functions, like creation of DO-loops, IF-THEN-ELSE constructs, availability of all language dependent items. At GSI many special functions have been added to LSEDIT performing global searches through all buffers, including of modules from include libraries, system calls, compilation of programs, etc. (Bookreader documentation available)

- SCA [$ SCA]:

  By the DEC Source Code Analyzer (SCA) together with the LSEDIT programs can be analyzed for cross-references and call consistencies. It produces detailed analysis libraries from program sources. (Bookreader documentation available)

- CMS and MMS (*) [$ CMS or $ MMS]:

  For large software project the DEC Code Management Software CMS and the DEC Module Management Software MMS traces the development and history of all components including source code changes, compilation and linking descriptions and library usage. It keeps track of consistency during the development phase. (Bookreader documentation available)

- PCA (*) [$ PCA]:

  The DEC Performance and Coverage Analyzer PCA measures the run-time performance of programs. In addition, it analysis which part of the code will be accessed during run-time. The statistic results are available as histograms or tables. (Bookreader documentation available)

- Compilers:

  The following compilers are available (*) (all compilers, except Modula 2, support the full screen debugger and the inter-language mixture and the Bookreader):

  - DEC FORTRAN-77 [$ FOR]
  - VAX C [$ CC]
  - DEC PASCAL [$ PASCAL]
  - VAX PL/I [$ PLI]
  - MODULA 2 [$ MODULA]
  - VAX BASIC [$ BASIC]
  - DEC OPS5 (for expert systems) [$ OPS5]

- graphic systems:

  The following graphic systems and packages are available (*):

  - DECimage Application Software (image software package)
  - DECphigs (Programmer's Hierarchical Interactive Graphics System)
  - DECwindows/MOTIF (Xwindow interface)
  - DEC GKS (Graphics Kernel System)

  - GTS-GRAL GKS (Graphics Kernel System)

  (Bookreader documentation available)

- DEC Notes `[$ NOTES]:`

  DEC Notes is an interactive notes and conference tool for many users joining common notes conferences. Users can read or write notes depending on the conference type. (Bookreader documentation available)

- Relational Data Base, RDB `(*) [$ RDO or $ SQL]:`

  The run-time version of the relational data base RDB is available on all Alpha AXP and VAX systems. They can be manipulated interactively by the Relational Database Operator (RDO) utility or by the SQL utility. Programs can use SQL-Precompiler statements. (Bookreader documentation available)

- Common Data Dictionary, CDD `(*) [$ CDO]:`

  The general common data dictionary CDD can be used together with the relational data base RDB and application programs. CDD/Repository is an active, distributed repository system that allows users to organize, manage, control, and integrate tools and applications across an entire enterprise. A repository is a central location where elements can be defined, stored, and shared. Elements correspond to various entities involved in a project. These include data definitions, models, files, reports, databases, tools, and system configurations. CDD/Repository allows you to manage these entities throughout the system life cycle and across multiple applications. (Bookreader documentation available)

- Datatrieve `(*) [$ DTR]:`

  DEC DATATRIEVE for OpenVMS is a data management tool for defining, storing, updating, and displaying data. It provides interactive and program-callable access to data, a report writing facility, a graphics capability, screen formatting support, and distributed access on a network connected by DECnet. (DECwindows/Motif is supported. Bookreader documentation available)

- RALLY `(*) [$ RALLY]:`

  DEC RALLY is an object-based fourth-generation application development system that provides a menu- and forms-based environment for creating, modifying, and executing interactive database applications. RALLY consists of a Definition System for creating, modifying, and testing RALLY applications and a Run-Time System for executing RALLY applications. (Bookreader documentation available)

- DECdecision, DECchart, DECcalc `(*) [$ INFORM]:`

  DECdecision is an integrated information management and decision support environment composed of spreadsheet, database, and charting components, and a cross-application tape recorder. Use DECdecision to store data, access and query data, analyze complex data, and create graphs based on that data.
  DECdecision includes the following components:

  - DECquery, the database management component
  - DECdecision Calc, the spreadsheet component
  - DECchart, the graphing component
  - DECdecision Builder, the cross-application tape recorder

  To start DECdecision from DCL level, type `$ INFORM` at the DCL prompt. DECdecision displays the Control Panel from which you start each of the components and Builder by clicking on the button to start them. (Runs only under DECwindows/Motif. Bookreader documentation available)

- Mathematica `(*) [$ MATHEMATICA]:`

  This system allows mathematical and graphical handling of data. Most mathematical function used in physics are included. Currently only available on the VAXstation VSAN.

- DEC TCP/IP Services for OpenVMS (UCX) `(*) [$ TELNET and $ FTP]:`

  The DEC TCP/IP Services for OpenVMS promote sharing between Alpha AXP and VAX servers and UNIX clients. It supports networking, file access, and application development between Alpha AXP and VAX servers and UNIX clients. The DEC TCP/IP Services for OpenVMS software is comprised of several components, e.g.:

  - Communication subsystem (Internet)
  - Network File System (NFS)
  - File Transfer Protocol (FTP)
  - Remote Terminal Services (TELNET)
  - Line Printer Daemon Protocol (LPD)
  - Berkeley Internet Name Domain (BIND) Resolver
  - Simple Mail Transfer Protocol (SMTP)
  - Simple Network Management Protocol (SNMP)

  (Bookreader documentation available)

- DEC SoftPC (*) [$ SOFTPC]:

  The SOFTPC command allows an OpenVMS system to emulate an IBM PC AT 286. It can operate either in DECwindows mode or character cell mode. (Bookreader documentation available)

- PCDISK (*) [$ PCDISK]:

  PCDISK is a DOS file maintenance utility which is used to maintain DOS devices, e.g. 3 1/2 " floppies, accessible to OpenVMS. The interface is DOS like, having many emulated DOS commands. OpenVMS style prompting, line editing and HELP are available. In some cases a OpenVMS command syntax may be used to perform a DOS function.

- Pathworks for DOS and Apple (*):

  Pathworks are programs to integrate IBM compatible PCs and Apple Macintoshs via Ethernet with OpenVMS. Pathworks uses DECnet, LAT, or TCP/IP as communication protocol. It allows to copy files, to simulate DOS hard-disks on a VAX, to run DECwindows/Motif, and to use all OpenVMS printer queues.

- RSX [$ MCR]:

  This PDP-11 emulation software allows the compilation and linking of PDP-11 software on the VAX. It is still in use for the accelerator control system and the J11 based CAMAC single crate system of GOOSY.

- System and network management tools (*). These programs are not available for standard users.

  – DECmcc Station Management Software:
  Covered by the Enterprise Management Architecture EMA several system and network programs are bundled together, e.g. for Terminal Servers (TSM), bridges (ELMS), Ethernet traffic analysis (LTM), Ethernet control (ETHERnim), DECnet and TCP/IP SNMP monitor. The common user interface is DECwindows/Motif. It will be converted to become OSF DME compliant in the future.

  – ATEM:
  Using a relational data base (RDB) and artificial intelligent (AI) software ATEM manages the data of all network components inclusive detailed information about descriptions, locations, installation dates, and pictures (images).

  – DECperformance Solution DECps:
  DECps collects and analyses the performance of OpenVMS systems and peripherals in detail. Extensive analysis tools are available providing exact hints about malfunctions or system false parameter settings using a built-in expert system.

- Distributed Name Service, DNS:
  DNS manages network-wide logical names, e.g. for peripherals or computer node names. It will be converted to the OSI X-500 standard in the future.
- Remote System Manager, RMS:
  Allows system management operations like software installations and backups on all VAX nodes on the Ethernet, even on other clusters.
- X-25 Software, PSI:
  This management software running on the VAX VSCN only controls the X-25 (DATEX-P, WIN) PTT network connection.
- Volume Shadowing:
  It allows the shadowing (simultaneous on-line use) of to identical disk drives as one logical device. If the hardware of one drive should crash the second drive runs uninterrupted.

F.2 Software installed on OpenVMS Alpha AXP

The following software products are currently installed on the main OpenVMS Alpha AXP Cluster at GSI and is available on the Alpha AXP Workstations connected to this Cluster. A `(*)` marks all software not available on other Alpha AXP Workstations. The DCL command to start the application is given in `[ ]`.

To all products documents and on-line Help are available. To most products a Bookreader documentation is available, too. For questions contact the OpenVMS Advisory Service (for names see on page 5).

DECwindows/Motif is available on all Alpha AXP workstations. For details about DECwindows/Motif refer to section 2.4.3 on page 19.

- Bookreader `[$ BOOK]`:

  If your are working directly on an Alpha AXP workstation or via a Xwindow terminal on an Alpha AXP workstation running DECwindows/Motif and only then a general utility is available to get all OpenVMS manuals on-line in windows on your screen.
  The manuals are stored on several CD-disks mounted on a centralized DEC InfoServer connected to Ethernet. This allows read access from all Alpha AXP workstations at GSI to the same CD-disk drives.
  For details refer to section 3.2 on page 30.

- DECwrite `(*) [$ DECWRITE]`:

DECwrite is a WYSIWYG (What You See Is What You Get) editor under DECwindows/Motif to compose complex documentations including text, graphics, and images previously scanned with DECimage Scan software. (Runs only under DECwindows/Motif. Bookreader documentation available)

- LaTeX (*) [$ LATEX or $ LATEXD and $ DVIPRI or $ NXDVI]:

  Text written in TeX or LaTeX can be translated with the LaTeX compiler using the DCL command `$ LATEX file` for English text or `$ LATEXD file` for German text. The result can be transformed to laser printer format and printed with the `$ DVIPRI` command or to the DECwindows/Motif format to be previewed with the `$ NXDVI` command.

- SPELL (*) [$ SPELL]:

  There is a standalone spell checker utility for English written text files. Although, you should always use the `SPELL` command within `LSEDIT` for English text or the spell checker within `DECwrite` for English or German text.

- LSEDIT [$ LSE]:

  This Language Sensitive text-editor supports various languages by specific functions, like creation of DO-loops, IF-THEN-ELSE constructs, availability of all language dependent items. At GSI many special functions have been added to LSEDIT performing global searches through all buffers, including of modules from include libraries, system calls, compilation of programs, etc. (Bookreader documentation available)

- SCA [$ SCA]:

  By the DEC Source Code Analyzer (SCA) together with the LSEDIT programs can be analyzed for cross-references and call consistencies. It produces detailed analysis libraries from program sources. (Bookreader documentation available)

- CMS and MMS (*) [$ CMS or $ MMS]:

  For large software project the DEC Code Management Software CMS and the DEC Module Management Software MMS traces the development and history of all components including source code changes, compilation and linking descriptions and library usage. It keeps track of consistency during the development phase. (Bookreader documentation available)

- PCA (*) [$ PCA]:

  The DEC Performance and Coverage Analyzer PCA measures the run-time performance of programs. In addition, it analysis which part of the code will be accessed during run-time.

The statistic results are available as histograms or tables. (Bookreader documentation available)

- Compilers:

  The following compilers are available (*) (all compilers, except Modula 2, support the full screen debugger and the inter-language mixture and the Bookreader):

  - DEC FORTRAN-77 [$ FOR]
  - DEC C [$ CC]
  - DEC C++ [$ CXX]
  - DEC PASCAL [$ PASCAL]
  - DEC PL/I [$ PLI]
  - DEC BASIC [$ BASIC]
  - DEC OPS5 (for expert systems) [$ OPS5]
  - DEC VEST [$ VEST]
  - DEC MACRO32 [$ MACRO/MIGRATION]
  - DEC MACRO64 [$ MACRO/ALPHA]

- graphic systems:

  The following graphic systems and packages are available (*):

  - DECphigs (Programmer's Hierarchical Interactive Graphics System)
  - DECwindows/MOTIF (Xwindow interface)
  - DEC GKS (Graphics Kernel System)

  (Bookreader documentation available)

- Relational Data Base, RDB (*) [$ RDO or $ SQL]:

  The run-time version of the relational data base RDB is available on all Alpha AXP and VAX systems. They can be manipulated interactively by the Relational Database Operator (RDO) utility or by the SQL utility. Programs can use SQL-Precompiler statements. (Bookreader documentation available)

- Mathematica (*) [$ MATHEMATICA]:

  This system allows mathematical and graphical handling of data. Most mathematical function used in physics are included.

- DEC TCP/IP Services for OpenVMS (UCX) `(*) [$ TELNET and $ FTP]:`

  The DEC TCP/IP Services for OpenVMS promote sharing between Alpha AXP and VAX servers and UNIX clients. It supports networking, file access, and application development between Alpha AXP and VAX servers and UNIX clients. The DEC TCP/IP Services for OpenVMS software is comprised of several components, e.g.:

  - Communication subsystem (Internet)
  - Network File System (NFS)
  - File Transfer Protocol (FTP)
  - Remote Terminal Services (TELNET)
  - Line Printer Daemon Protocol (LPD)
  - Berkeley Internet Name Domain (BIND) Resolver
  - Simple Mail Transfer Protocol (SMTP)
  - Simple Network Management Protocol (SNMP)

  (Bookreader documentation available)

- DEC SoftWindows `(*) [$ SOFTWINDOWS]:`

  The SOFTWINDOWS command allows an OpenVMS system to emulate an IBM PC AT 286. It can operate either in DECwindows mode or character cell mode. (Bookreader documentation available)

- Pathworks for DOS and Apple `(*)`:

  Pathworks are programs to integrate IBM compatible PCs and Apple Macintoshs via Ethernet with OpenVMS. Pathworks uses DECnet, LAT, or TCP/IP as communication protocol. It allows to copy files, to simulate DOS hard-disks on an Alpha AXP workstation or a VAX, to run DECwindows/Motif, and to use all OpenVMS printer queues.

- System and network management tools `(*)`. These programs are not available for standard users.

  - DECperformance Solution DECps:
    DECps collects and analyses the performance of OpenVMS systems and peripherals in detail. Extensive analysis tools are available providing exact hints about malfunctions or system false parameter settings using a built-in expert system.
  - Polycenter Disk File Optimizer for OpenVMS:
    It allows the defragmentation of OpenVMS file systems.

– Volume Shadowing:
It allows the shadowing (simultaneous on-line use) of to identical disk drives as one logical device. If the hardware of one drive should crash the second drive runs uninterrupted.

Appendix G

HELP File Format

The format of a help file must be (Note that there are in the first column only level numbers; any text must begin with a <SPACE>):

```
1 key
<space> text line
<space> text line
  ...
2 subkey
<space> text line
  ...
3 subsubkey
<space> text line
  ...
2 subkey
<space> text line
  ...
3 subsubkey
<space> text line
  ...
1 key
<space> text line
  ...
2 subkey
<space> text line
  ...
```

The module name(s) is (are) determined from the main keys in the file. Several modules may be in one file.

Appendix H

System Error Messages

System error messages are displayed in the format:

```
%FACILITY-L-IDENT, text

FACILITY = the mnemonic for the program issuing the message
L        = the first letter of the severity code
         (F = fatal error, E = error, W = warning, I = information, S = success)
IDENT    = an abbreviation of the text
text     = an explanation of the error
```

Example:

```
$ DELETE ADAM.TXT
%DELETE-E-DELVER, explicit version number or wild card required
```

The facility DELETE shows an error (-E-) with the ident DELVER. The reason was the missing file version number for the `DELETE` command. A correct command could be

```
$ DELETE ADAM.TXT;*
```

which would delete all versions of the file.

For a detailed explanation of an error, see the *OpenVMS System Messages and Recovery Procedures Reference Manual*. You may also user the HELP facility by typing

```
$ HELP /MESSAGE error-code
e.g.
$ HELP /MESSAGE ACCVIO
```

to get an error code explanation.

OpenVMS Glossary

The glossary follows the following format:

- The glossary words are printed in bold.
- Words that can be cross-referenced are *emphasized*.
- Words that can be entered at the terminal are in `typewriter style`.

account: Enables access to the system software (*command* interpreters, *compilers*, *utilities*, and so on) including the ability to perform work of general nature (program development, text editing, and so on). There is always an username, a password and a disk file directory associated with an account.

Alpha AXP: The name of computer systems manufactured by DIGITAL Equipment Corporation. The *Alpha AXP* computer systems are based on a high performance RISC (Reduced Instruction Set Computer) Computer *CPU*. There is a number of different *Alpha AXP* systems available from PC-like Workstations (like DEC 3000-300LX) to computer center servers (like Digital 2100 Server Model A500MP, e.g. AXP601 and AXP602 at GSI). These *Alpha AXP* computer systems may run three different operating systems: *OpenVMS*, *Digital UNIX* (former OSF/1) or MicroSoft Windows/NT.

ASCII: American Standard Code for Information Interchange. ASCII is the standard format for sending readable text. It is a code used by many computers to translate letters, numbers, and symbols from a keyboard into machine code, and vice versa. Thus, an ASCII *file* is a *file* that can be read both by people and by computers.

assembler: Language processor that translates a source program containing assembly language directives and machine instructions into an object module.

batch: Mode of non-interactive processing in which all *commands* to be executed by the *operating system* and, optionally, *data* to be used as input to the *commands* are placed in a *command procedure file* and submitted to the computer system for execution in a so called *batch queue* (corresponding *OpenVMS DCL command* `$ SUBMIT file /QUEUE=queuename`).

batch job: A non-interactive *process*.

batch queue: Execution queue for *batch jobs* submitted to the computer system. A *batch queue* executes *batch jobs* one after the other depending on their order in the *queue*.

BitNet: U.S. computer *network* for academic and research institutes, *EARN*, for example. The *IBM mainframe* has the *EARN* address `DDAGSI3`. This computer is the only one accessible at GSI via *EARN* (not available on *VAX* nor on *Alpha AXP*).

Bookreader: If your are working directly on an *Alpha AXP*, a *VAXstation*, or via a *Xwindow terminal* on an *Alpha AXP* or a *VAXstation* running *DECwindows/Motif*, and only then, this general *utility* is available to get all *OpenVMS* manuals on-line in windows on your screen. The manuals are stored on several CD-disks mounted on a centralized *DEC InfoServer* connected to Ethernet only. This allows read access from all *Alpha AXP* or *VAXstations* at GSI to the same CD-disk drives.

bridge: A bridge separates the traffic of two *Ethernet* or *FDDI* segments. It keeps address tables of nodes of each side of the bridge and allows transfer of packets according to these tables. Traffic between nodes both on one side of the bridge does not affect the traffic on the other side. There are bridges for Ethernet-Ethernet, Ethernet-FDDI, and FDDI-FDDI (*GIGAswitch*).

buffer: A temporary *data* storage area in the computer's main memory.

cluster: *see VMScluster*

command: An instruction or request for the system to perform a particular action. An entire *command string* consists of the *command* name with any *parameters* and/or *qualifiers*.

command format: see *syntax*

command procedure: *File* containing a sequence of *commands* to be executed by the *command language processor DCL*. The command procedure can be submitted for execution at the *terminal* (e.g. `$ @XYZ.COM`) or as a *batch job* (e.g. `$ SUB XYZ.COM`).

command string: A *command* with any *parameters* and/or *qualifiers*.

compiler: Language processor that translates a source program containing high–level language statements (for example C, FORTRAN, or PL/I) into an *object* module, the internal machine code format file. The object module can be linked by a *linker* to an executable *image*.

Computer Interconnect CI: A high-speed, fault-tolerant, dual-path serial bus, which has a bandwidth of 70 MBits per second. With the *CI*, any combination of *VAX* processor nodes and intelligent I/O subsystem (*Hierarchical Storage Controller, HSC*) nodes -up to 16 in number - can be loosely coupled in a computer room environment running the *VMScluster* software.

concatenate: To link together in a series.

CPU: Central Processing Unit. It is the hardware that handles all calculating and routing of input and output(I/O), as well as executing images. The CPU is the part of the computer that actually computes.

cursor: A flashing indicator used on video *terminals* to point to the screen position where the next character will appear. It is called a "cursor" because it shows the "course" or direction the printed or typed line will follow. On graphics workstations like *Alpha AXP* or *VAXstations* the cursor can be moved across the screen by a device called *mouse*.

data: A general term used for any representation of facts, concepts, or instructions in a form suitable for communication, interpretation, or processing. When *commands prompt* you for *command* elements, they are asking you for *data* to process.

DATEX-P: The computer telecommunication network service offered by the German PTT, Telekom. It uses the *X-25* packet-switched communication protocol on telephone lines with modems. The corresponding *VAX* (no *Alpha AXP*) software is called *PSI*, the Packetnet System Interface of *DEC*. The *Wissenschafts-Netz WI* is a subset of *DATEX-P* for all German Universities and research labs.

DCL: Digital Command Language. It provides a means of communication between the user and the *operating system*. DCL is designed for ease of use. *Commands* are English words, and if necessary elements are not typed in, DCL will prompt for them.

DEC: The *Digital Equipment Corporation* is the computer manufactory of *Alpha AXP*, *VAX*, *DECsystems*, and PDP hardware and software, namely *OpenVMS*, *Digital UNIX* (former (OSF/1), *ULTRIX*, and RSX-11 operating systems. It also produces high performance *Ethernet/FDDI* network equipment like *terminal servers*, *bridges*, and *GIGAswitch*.

DECnet: The standard *network* software for all *DEC* computers. This software uses the following hardware: *Ethernet*, *FDDI*, PTT lines with *X-25* protocol (*DATEX-P*, *WIN*), or terminal lines. GSI has a large internal DECnet and is in addition connected to an international DECnet for the High Energy Physics community, called *HEPnet*.

DECpresent: DECpresent is replaced by *DECwrite*.

DECwindows/Motif: The standard graphical user interface available on *Alpha AXP*, *VAXstations*, and *DECstations* is *DECwindows/Motif*. It is compatible to *OSF Motif* and the *Xwindow* standards.

DECwrite: DECwrite is a WYSIWYG (what you see is what you get) editor under *DECwindows/Motif* to compose complex documentations including text, graphics, and *image data*. It runs under *OpenVMS* on *Alpha AXP* and *VAXstation* and under MicroSoft Windows on IBM compatible PCs.

DECserver: A terminal interface to *OpenVMS* and *Digital UNIX* (former OSF/1) on *Alpha AXP* and *VAX*, *ULTRIX* on *DECstations*, OS/9 and *LynxOS* on *VME-boards*, and MVS on *IBM mainframe* computers using the *Ethernet* hardware.

DECstation: A *DECstation* is a *RISC* workstation running the DEC *UNIX* version called *ULTRIX*. It will be replaced by *Alpha AXP* systems running *Digital UNIX* (former OSF/1) in the future. The terminal interface is *DECwindows/Motif*.

default: Value supplied by the system when a user does not specify a required *command parameter* or *qualifier*.

delimiter: A character that separates, terminates, or organizes elements of a character string or statement. For example, in the *file specification*, `STORIES.DAT`, the period (.) is the delimiter that enables the system to tell the difference between the *file name* `STORIES` and the *file type* `DAT`.

device: Any peripheral hardware connected to the processor and capable of receiving, storing, or transmitting *data*. *Laser printers* and *terminals* are examples of record-oriented devices. Magnetic tapes and disks are examples of mass storage devices. Terminal line interfaces and interprocessor links are examples of communications devices. All devices have names either in the form `ddnn:`, where `dd` is a two letter mnemonic, `nn` is an octal number, and the colon (`:`) is a required terminator or as a logical name.

device name: Identification of a physical *device* (for example, LTA401) or a logical name (for example, SYS$OUTPUT) that is equated to a physical device name. The first two characters of a physical device name defines the device hardware type, the following character and numbers define the controller and the running device number, e.g. LTA5230 means a *DECserver* terminal on controller A with the number 5230. Disks are defined as logical names including a specific directory, the so called *roots*. A user normally handles *roots* only, like `KP2$ROOT`.

device separation: Separation of alphanumeric *terminal* for *commands* and a graphic *terminal* on an *Alpha AXP* or a *VAX* or on the *IBM mainframe*.

Digital UNIX: *Digital UNIX* (former OSF/1) is based on a *UNIX* operating system developed by the Open Software Foundation *OSF*. It is available for RISC computers of *DEC*, the *Alpha AXP*.

directory: A *file* that briefly catalogs a set of *files* stored on disk or tape. The directory includes the name, type, *version number*, creation date, modification date, and other information of each *file* in the set.

editor: see *LSEDIT*

EARN: The European Academic & Research *Network* which is the European version of the U.S. *BitNet*. The *IBM mainframe* has the EARN address `DDAGSI3`. This computer is the only one accessible at GSI via EARN (not available on *VAX* nor on *Alpha AXP*).

error message: Sent by the system when some action you have requested fails. Each error message identifies the particular part that detected the error. The great majority of error messages result from typing mistakes or mistakes in syntax. Often, you can correct the error by retyping the *command*. The severity of an error is given by the following levels: F = fatal error, E = error, W = warning, I = information, S = success.

Ethernet: Computer *Network* hardware. A coaxcable transmitting messages and *data* serially at 10 Mbits/sec between computers, *terminal* servers, and other devices.

extension: see file type.

FDDI: Fiber Distributed Data Interface. Computer *Network* hardware. A fiber cable ring and in the future also coaxcable transmitting messages and *data* serial at 100 Mbits/sec between computers, *terminal* servers, and other devices. The FDDI cabling can be connected transparently by a bridge to *Ethernet* cables, i.e. any station on FDDI can talk to any station on *Ethernet* as long as both are talking the same network software (e.g. *TCP/IP* or *DECnet*). Up to 32 FDDI-rings can be switched by a *GIGAswitch* hardware which is already in use at GSI.

file: Collection of *data* treated as a unit; generally used to refer to *data* stored on magnetic tapes or disks.

file name: The name component of a *file specification*.

file specification: Unique identification of a *file*.
`node::device:[directory]filename.type;versionnumber`

file type: The type of a *file specification*. A *file* type generally describes the nature of a *file*, or how it is used. For example, `TEX` indicates a LaTeX program or `FOR` indicates a FORTRAN program source file.

file version number: Numeric component of a *file specification*. When a *file* is edited, its version number is increased by one.

FTP File Transfer Protocol: FTP is the user interface to the *Internet TCP/IP* standard *file* transfer protocol. FTP allows users to log into remote systems, transfer *files* to and from remote hosts, obtain listings of directories on remote hosts, and other common file operations. Most FTP commands require one or more file specifications. Local file specifications must be valid *OpenVMS file specifications*. Remote *file specifications* will be interpreted by the remote host and therefore must be valid to that host. Keep in mind that *UNIX* node, user, and *file names* are mostly case sensitive. Any *file specification* which is not a valid *OpenVMS file specification* must be quoted, specifically if it contains lower case characters as usual in *UNIX* environments. The GSI *Alpha AXP601* has the *Internet* address `axp601.gsi.de`, the GSI *VAX* VSCN has the *Internet* address `vscn.gsi.de`, the *IBM mainframe* has the *Internet* address `mvs.gsi.de`.

GIGAswitch: Computer *Network* hardware. Up to 32 *FDDI* rings can be switched by a GIGAswitch hardware which is already in use at GSI. A *FDDI* fiber cable ring and in the future also coaxcable transmitting messages and *data* serially at 100 Mbits/sec between computers, *terminal* servers, and other devices. The *FDDI* data are switch on packet base between two rings. Several ring pairs can communicate at the same time (cross bar switch). If an *Alpha AXP* or another GIGAswitch is connected directly to a GIGAswitch port they will communicate in full duplex mode, i.e. with 200 Mbits/sec. In the future cards for ATM (Asynchronous Transfer Mode, a new PTT standard) will be available for GIGAswitches. A GIGAswitch has an internal bandwidth of over 3 Gbits/sec.

Global Section: A Global Section is a *file* whose contents can be loaded into the program address space of several users simultaneously. The contents will be only once in the physical memory, i.e. the data are shared by several programs. The Global Section must be created as a file and then it must be defined as a Global Section by *OpenVMS* system service routines from a program, e.g. by the mount of a *GOOSY* Data Base. There are system wide Global Sections, e.g. compiler code, and group Global Sections, e.g. a *GOOSY* Data Base. The Global Section will be defined as long as the system is running or another program uses a *OpenVMS* system routine to delete the Global Section attribute, e.g. by the dismount of a *GOOSY* Data Base.

GOOSY: The GSI Online/Offline System. It is a PL/I based nuclear data acquisition and analysis system implemented on *Alpha AXP* and *VAX OpenVMS* computers.

help file: A text *file* in a format for use with the `HELP` *command*. Help *files* can include simply organized information and can provide up to nine levels of search.

HEPnet: GSI is connected to an international *DECnet* for the High Energy Physics community, called *HEPnet*.

Hierarchical Storage Controller, HSC: A self-contained, intelligent, mass-storage (magnetic disk and tape drives) controller that communicates with *VAX* processors in a *VMScluster* environment via a *Computer Interconnect, CI* bus.

IBM mainframe: The GSI computer center is equipped with an IBM mainframe type 9121-621 running MVS/ESA/TSO under VM/ESA.

IBM terminal: One can connect any VT220 type terminal through the *DECserver* to the IBM system.

image data: Pictures stored pixel-wise are called image data. You can produce and manipulate image data with an image *scanner* hardware and software and with the application program `PAINT`. You can include image files in *DECwrite* documents. For the locations and the use of scanners see appendix D.2 on page 116.

image, executable: Programs are written and stored in plain text files, so called *source code* files. These *source code* files are translated by a *compiler* to a computer internal binary format the *object code*. Several of these *object code* files are *linked* together producing an image file which finally can be executed on the computer.

InfoServer: The *DEC InfoServer* is a standalone computer connected to *Ethernet*. It has several CD-disk drives attached locally. The *InfoServer* offers CD-disks mounted on these drives network-wide to all *Alpha AXP*, *VAX* and PCs. The CD-disks are used mainly to store *Bookreader* manuals available on several CD-disks. This allows read access from all *Alpha AXP* and *VAXstations* at GSI to the same Bookreader documentation.

interactive: Mode of communication with the *operating system* in which a user enters a *command*, and the system executes it and responds.

Internet: Computer *network* originally founded by the U.S. DoD. It is worldwide used by research and industry and is based upon the *TCP/IP* protocol. Keep in mind that *UNIX* node and user names are mostly case sensitive. The GSI *Alpha AXP601* has the *Internet* address `axp601.gsi.de`, the GSI *VAX* VSCN has the *Internet* address `vscn.gsi.de`, the *IBM mainframe* has the *Internet* address `mvs.gsi.de`.

job: (1) The *accounting* unit equivalent to a *process*; jobs are classified as *batch* or *interactive*. (2) A print job.

K: A unit for measuring the size of memory or similar resources. K is short for kilo and is used to mean 1000, although in computer slang K is equal to 1024.

keypad: The small set of keys to the far right of the main keyboard on a *terminal*. The key functions depend on the program using them, e.g. *LSEDIT*, DEBUG, *DCL*, or *IBM mainframe*.

Language Sensitive Editor See *LSEDIT*

Laser printer There are several laser printers LN03+, LN03R, PrintServer-20, PrintServer-17, HP, Tektronix, and Lexmark for black/white and color available at GSI (see appendix D on page 109.
They will print any text, all LaTeX , PostScript, and Tektronix 4014 output.

LAT: A communications protocol that the *OpenVMS* operating system uses within a local area network over *Ethernet/FDDI* to communicate with *terminal servers*.

LaTeX : A document preparation system used writing letters, books, manuals, articles, etc.

LAVC Local Area VAX Cluster: A type of *VMScluster* configuration in which cluster communication is carried over *Ethernet* or *FDDI* by software that emulated certain *Computer Interconnect (CI)* or *DSSI* functions. A *VMScluster* node can be any *Alpha AXP* or *VAX* processor.

linker: Programs are written and stored in plain text files, so called *source code* files. These *source code* files are translated by a *compiler* to a computer internal binary format the *object code*. Several of these *object code* files are *linked* together producing an image file which finally can be executed on the computer.

logging in: To perform a sequence of actions at a *terminal* that establishes a user's communication with the *operating system* and sets up *default* characteristics for the user's *terminal session*. `LOGIN.COM` is, as its name suggests, a log in command procedure. It is executed whenever you log in. You can add commands to this file. These commands are then executed whenever you log in.
To open an IBM MVS/TSO session use the `LOGON` command.

logging out: To terminate *interactive* communication with the *operating system*. The `LOGOUT` *command* executes the procedure and ends a *terminal session*.
If you are in an IBM MVS/TSO terminal session use the `LOGOFF` command to leave the session.

LSEDIT: The Language Sensitive Editor, which is used for writing any text file, e.g. *command procedures*, programs in various languages (FORTRAN, PL/I, etc.), or LaTeX text. There

are other text editors available, namely `EDT` and EVE. For more information on these text editors see the *LSEDIT* or DECset manuals or the *Bookreader* documentation.

LynxOS: LynxOS is a real-time operating system designed for use with Motorola based GSI CAMAC controller CVC, *VME* processor boards (like Eltec E6, E7), and Intel 80x86 based PCs. It provides fully *UNIX* compatible real-time options and POSIX standards. It is a multitasking operating system with predictable timing behavior and has user definable priorities and a fully pre-emptable kernel. It is used at GSI has the operating system for data acquisition with the single (CVC, SBS) and multiple branch system.

MAIL utility: A *utility* which allows you to send mail internally (within the GSI) and externally over *EARN/BitNet*, *HEPnet*, and *Internet*.

Messtation: The computer station where V6000A (FRITZ) is housed. There are also magtapes and a *laser printer* commonly available.

Motif: The standard graphical user interface available on *Alpha AXP*, *VAXstations*, and *DECstations* is *DECwindows/Motif*. It is compatible to the industry standard *OSF Motif* and the *Xwindow* standard.

mouse: To move a *cursor* across the terminal screen workstations are using a special pointer device connected to them, called *mouse*. When you move the *mouse* by hand the *cursor* moves correspondingly over the screen. There are three buttons on an *Alpha AXP*, *VAXstation*, and *DECstation* mouse called MB1 (leftmost), MB2 (middle), and MB3 (rightmost). Hitting these buttons (mouse click) or keeping them pressed during the mouse move initiates actions depending on the application you are currently running.

network: A collection of interconnected computer systems called *nodes*.

Network File System NFS: The Network File System (NFS) allows the access of disks across a network. It is based on the *TCP/IP Internet* protocol. A *NFS* server offers disks across the network to NFS clients which may mount these disks and use them as if they are locally connected.

node: An individual computer system in a *network* that can communicate with other computer systems in the *network*.

object code: Programs are written and stored in plain text files, so called *source code* files. These *source code* files are translated by a *compiler* to a computer internal binary format

the *object code*. Several of these *object code* files are *linked* together producing an image file which finally can be executed on the computer.

OpenVMS: VMS = Virtual Memory System. The name of the *Alpha AXP and VAX operating system*.

operating system: The system software that controls the operations of the computer. *Alpha AXP* and *VAX* computers are running the *OpenVMS* operating system. The *DECstations* are running the *ULTRIX* operating system.

OSF: The Open Software Foundation, *OSF*, is a non-profit software consortium founded by large computer and software companies, like *DEC*, IBM, HP, Siemens. OSF develops an *UNIX* based operating system, called *OSF/1*, and several basic software packages, like Distributed Computing Environment (DCE), Distributed Management Environment (DME), and a *Xwindow* based user interface (*Motif*).

OSF/1: *OSF/1* is a *UNIX* operating system developed by the Open Software Foundation *OSF*. See *Digital UNIX*.

parameter: Object of a *command*. A parameter can be a file specification, a symbol value passed to a *command procedure*, or a word defined by the *DCL*.

password: Protective word associated with the *username*. A user logging in to the system must supply the correct password before the system will permit access. A user may change his password by using the *DCL* command `SET PASSWORD`.

Pathworks for DOS and Mac: *Pathworks* are programs to integrate IBM compatible PCs and Apple Macintoshs via *Ethernet* with *OpenVMS* and *UNIX*. *Pathworks* uses *DECnet*, *LAT*, or *TCP/IP* as communication protocol. It allows to copy files, to simulate DOS hard-disks on a *Alpha AXP* and *VAX*, to run *DECwindows/Motif*, and to use all *Alpha AXP* or *VAX* printer *queues*.

PECAD: An alphanumeric and graphics *terminal* emulating *DEC* VT220 and Tektronix 4014 *terminals*.

PERICOM: An alphanumeric and graphics *terminal* emulating *DEC* VT220 and Tektronix 4014 *terminals*.

PostScript: A programming language designed by Adobe Systems, Inc. to convey a description of virtually any desired page to the printer. It can describe a page layout containing any combination of text, graphical shapes, and digitized *image data*. It is used by most computer companies and can be handled by printers (PostScript printer) able to translate PostScript text files sent to them for printing (all *laser printers* at GSI).

prompting: A symbol used by the system as a cue to signal that the system is ready to accept input from you.

preamble: Term from LaTeX . The commands before `\begin document`.

process: The environment in which you use the *Alpha AXP* or *VAX OpenVMS* system.

`PURGE`: The action of deleting older versions of *files*, but keeping the newest one (the one with the highest version number).

qualifier: *Command* modifier that describes the operation of a *command*. A qualifier is always preceded by a slash character (/).

queue: A line of items to be processed, e.g. a *batch* queue.

root: A root is a system wide defined logical name defining a physical disk *device name* and a specific *directory*. Such a root is used like a *device name* in all cases, e.g. in the *file specification* instead of the *device name*. A user will get a root instead of a physical disk as his default file device, e.g. AP$ROOT, KP1$ROOT, etc.

scanner: Pictures stored pixel-wise are called *image data*. You can produce and manipulate image data with an image *scanner* hardware and software and with the application program `PAINT`. A scanner scans a picture pixel-wise with a laser beam producing pixel image data. These data can be included in *DECwrite*. For scanner types and locations see appendix D on page 109.

service: A service is either an *Alpha AXP* or a *VAX node* or a *VME processor* console port or the *IBM mainframe* connected to a *DECserver* provided for users at GSI.

source code: Programs are written and stored in plain text files, so called *source code* files. These *source code* files are translated by a *compiler* to a computer internal binary format the *object code*. Several of these *object code* files are *linked* together producing an image file which finally can be executed on the computer.

subdirectory: *Directory file* cataloged in a higher-level *directory* that lists additional *files* belonging to the owner of the *directory*.

syntax: The form that a command must follow. Misspelled words are the most common syntax errors.

system manager: Person who makes resources available to users and sets up restrictions governing the use of such resources (see names on page 5).

TCP/IP: The Transmission Control Protocol/Internet Protocol, *TCP/IP*, was originally developed by the U.S. Department of Defense. Based on *TCP/IP*, the *Internet* network is a Computer *network* used worldwide by research and industry. Several protocol are based on *TCP/IP*, e.g. *Telnet* for remote login, the File Transfer Protocol (*FTP*) for remote file transfer, and the Network File System (*NFS*) for file serving. Keep in mind that *UNIX* node and user names are mostly case sensitive, i.e. under *OpenVMS* they must be included in double quotes, e.g. `"user@name.segment"`, `"M.Richter@gsi.de"`. The GSI *Alpha AXP601* has the *Internet* address `axp601.gsi.de`, the GSI *VAX* VSCN has the *Internet* address `vscn.gsi.de`, the *IBM mainframe* has the *Internet* address `mvs.gsi.de`.

Telnet: A terminal protocol for login on remote computers. *Telnet* is based upon the *Internet* network, which is a Computer *network* originally founded by the U.S. DoD. It is worldwide used by research and industry and is based upon the *TCP/IP* protocol. Keep in mind that *UNIX* node and user names are mostly case sensitive. The GSI *Alpha AXP601* has the *Internet* address `axp601.gsi.de`, the GSI *VAX* VSCN has the *Internet* address `vscn.gsi.de`, the *IBM mainframe* has the *Internet* address `mvs.gsi.de`.

terminal: Hardware communication device, with a typewriter–like keyboard that receives and transmits information between users and the system.

terminal server: A terminal interface to *Alpha AXP*, *VAX*, *IBM*, and *VME* computers using the *Ethernet* hardware. The *DECserver* is such a device.

type: see file type.

ULTRIX: A *UNIX* version from *DEC* for the MIPS RISC computers, like *DECstations*.

UNIX: A general purpose operating system for many different computer architectures developed by AT&T. *UNIX* flavors used at GSI are: IBM AIX, HP-UX, *ULTRIX*, *LynxOS*, and *Digital UNIX*.

username: Name by which the system identifies a particular user. To gain access to the system, a user specifies a username followed by a *password*.

utility: A general–purpose program that performs tasks included in an *operating system* to perform common functions, such as editing or *file* handling.

VAX: Virtual Address Extension. The name of a computer system manufactured by DIGITAL Equipment Corporation. The operating system for *VAX* computers is *OpenVMS*.

VMScluster: A highly integrated organization of *OpenVMS* systems that communicate over a high-speed communications path. *VMScluster* configurations have all the function of single node systems, plus the ability to share *CPU* resources, *queues*, and disk storage. Like a single-node system, the *VMScluster* configuration provides a single security and management environment. Member nodes can share the same *operating system* or serve specialized needs. There are three types of a *VMScluster* system configurations, depending on the medium used for interprocessor communications: *CI*-based, local area (*Ethernet/FDDI*, and mixed-interconnect (*CI* and *Ethernet/FDDI*).

VAX operating system: The *operating system* of the *VAX*, named *OpenVMS*.

VAXstation: A *VAX* workstation running the *OpenVMS*. The terminal interface is *DECwindows/Motif*.

version number: see file version number.

VMS: Now called *OpenVMS*. Virtual Memory System. The name of the *Alpha AXP and VAX operating system*.

VME: A bus standard developed by Motorola to interface process data to multiple micro processors connected to the VME bus. *GOOSY* is using *VME* with Motorola 680x0 processor boards to acquire and handle experiment data. The GSI accelerator control system uses VME processors for set-up and control of the equipment.

wildcard character: A symbol used with many *DCL commands* in place of all or part of a *file specification* to refer to several *files* rather that specifying them individually.

There are two symbols used: The * which can stand in place of unlimited characters, and the % which can stand only in place of one character. For example:

```
$ DIR *.TEX
$ COPY A*.FOR [TEST]*
$ DEL AD%M.TXT;*
```

The 1^{st} command lists all files with the extension '.TEX'. The 2^{nd} command copies all files with the extension '.FOR' and with their names starting with the character 'A' to the directory [TEST] without changing their names. The 3^{rd} command deletes all files with the extension '.TXT' and with names starting with 'AD' then any one character and ending with 'M'.

Wissenschafts-Netz WI: The computer telecommunication network service offered by the German PTT, Telekom to all German Universities and research labs. It uses the *X-25* packet-switched communication protocol on telephone lines with modems. The corresponding *VAX* software running at GSI on the *VAXstation* VSCN only is called *PSI*, the Packetnet System Interface of *DEC*.

WWW: The WorldWideWeb (WWW or W3) is the universe of network-accessible information, an embodiment of human knowledge. It is an initiative started at CERN, now with many participants. It has a body of software, and a set of protocols and conventions. W3 uses hypertext and multimedia techniques to make the web easy for anyone to roam, browse, and contribute to. The GSI WWW information is accessible via the URL "http://www.gsi.de/gsi.html". You will find a lot of information like this manual in the GSI WWW pages. Call the WWW on *OpenVMS* if your are running a *Xwindow* (*Motif*) session on a workstation or a *Xwindow terminal* by typing `$ XWWW`.

X-25: A packet-switched, synchronous communication protocol on telephone lines with modems. Based on *X-25 DATEX-P* is the computer telecommunication network service offered by the German PTT, Telekom. The *Wissenschafts-Netz WI* is a subset of *DATEX-P* for all German Universities and research labs. The corresponding *VAX* software running at GSI on the *VAXstation* VSCN only is called *PSI*, the Packetnet System Interface of *DEC*.

Xwindow: A software developed by the Massachusetts Institute of Technology MIT together with the computer companies *DEC*, IBM, and HP. This window software should unify the user's interface to workstations. Within the last years *Xwindow* has become a widely accepted defacto standard. Xwindow is also used as the base for *OSF/Motif*. The actual implementation for *Alpha AXP*, *VAXstations* and *DECstations* is *DECwindows/Motif*.

Xwindow terminal: Based on the *Xwindow* defacto standard developed by the Massachusetts Institute of Technology MIT several companies are offering graphics terminals connected directly to *Ethernet*. You can connect to any computer running the same network protocol (depending on the producer company mainly *TCP/IP*, *LAT*, or *DECnet*) and open terminal sessions like *DECwindows/Motif*. At GSI there are *Xwindow terminals* from *DEC*, IBM, and mainly Tektronix. Together with *Pathworks for DOS* or *Pathworks for Apple* an IBM compatible PC or an Apple Macintosh can simulate a *Xwindow terminal* by software.

Index

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Contents