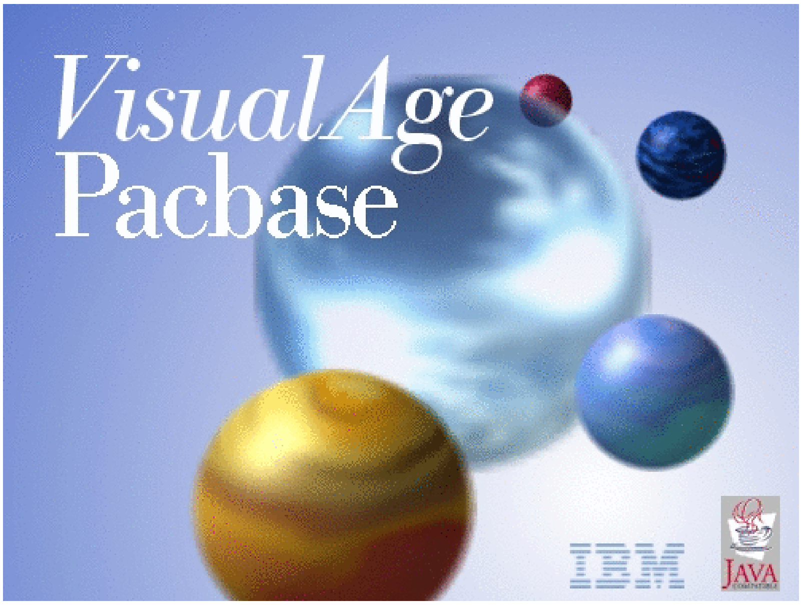
VisualAge Pacbase

IBM

Technical Support Information VisualAge Pacbase Additional Files ZAR980 sources for UNIX targets

Version 3.0



VisualAge Pacbase

Technical Support Information VisualAge Pacbase Additional Files ZAR980 sources for UNIX targets

Version 3.0

Note

Before using this document, read the general information under "Notices" on page v.

According to your licence agreement, you may consult or download the complete up-to-date collection of the VisualAge Pacbase documentation from the VisualAge Pacbase Support Center at:

http://www.ibm.com/software/awdtools/vapacbase/productinfo.htm

Consult the Catalog section in the Documentation home page to make sure you have the most recent edition of this document.

First Edition (April 2004)

This edition applies to the following licensed programs:
- VisualAge Pacbase Version 3.0

Comments on publications (including document reference number) should be sent electronically through the Support Center Web site at: http://www.ibm.com/software/awdtools/vapacbase/support.htm or to the following postal address:

IBM Paris Laboratory
1, place Jean–Baptiste Clément
93881 Noisy-le-Grand, France.

When you send information to IBM, you grant IBM a nonexclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

Contents

Notices

References in this publication to IBM products, programs, or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Subject to IBM's valid intellectual property or other legally protectable rights, any functionally equivalent product, program, or service may be used instead of the IBM product, program, or service. The evaluation and verification of operation in conjunction with other products, except those expressly designated by IBM, are the responsibility of the user.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to the IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk NY 10504–1785, U.S.A.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact IBM Paris Laboratory, SMC Department, 1 place J.B.Clément, 93881 Noisy-Le-Grand Cedex. Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

IBM may change this publication, the product described herein, or both.

Trademarks

IBM is a trademark of International Business Machines Corporation, Inc. AIX, AS/400, CICS, CICS/MVS, CICS/VSE, COBOL/2, DB2, IMS, MQSeries, OS/2, PACBASE, RACF, RS/6000, SQL/DS, TeamConnection, and VisualAge are trademarks of International Business Machines Corporation, Inc. in the United States and/or other countries.

Java and all Java-based trademarks and logos are trademarks of Sun Microsystems, Inc. in the United States and/or other countries.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States and/or other countries.

UNIX is a registered trademark in the United States and/or other countries licensed exclusively through X/Open Company Limited.

All other company, product, and service names may be trademarks of their respective owners.

Chapter 1. Presentation of the XP250UIX package

Object:

These utilities allow to use the Multi-screens variant, either for the Standard Dialog module, or for the Pacbase Web Connection module.

The ZAR980 program formats the message.

A set of tools is delivered for each generation target:

Bull-Gcos7, Bull-Gcos8, Compaq-VMS, HP3000, IBM-CICS, ICL, IBM-IMS, Microfocus-DOS, Unisys-A series, Microfocus-Unix.

Version:

Available for all versions.

Platforms:

These components are independent of the development platform as they concern the generated applications.

Reference documentation:

Dialog reference manual

Pacbase Web Connection Developer's Guide

PAW Developer's Guide

Components nature:

Text files, ISO compressed files

List of components:

readme_F.v02: French installation note

readme_E.v02: English installation note

zarbase.iso : Compressed components file zarinst.v02 : installation startup shell

<u>INSTALLATION :</u>

To install the package on a UNIX machine:

Unload the following files in binary mode

ZARINST.V02 (zarinst.v02)

ZARBASE.ISO (zarbase.iso)

Start the installation procedure

sh zarinst.v02 -E (English version)

or

sh zarinst.v02 -F (French version)

```
****************************************************
                    DIALOG MICRO FOCUS UNIX GENERATION
    ENVIRONNEMENT : UIX
                             REFERENCE : XP250V02UIX
                               VERSION : V02
                                  DATE : 02/04/2002
 ****************************************************
```

This technical package is composed of two utilities executable in a UNIX environment and for applications generated by the Dialog module in Cobol Micro Focus UNIX.

One of the utilities allows the communication of a dialog application revamped by PAW; the other utility manages the dialog and the keyword of the UNIX terminal.

The first utility is stored in the ZARPAW directory and the second one is located in the ZARTERM directory. A documentation is provided for the implementation of each utility:

- in the ZARPAW directory, the zarpaw.doc file,

- in the ZARTERM directory, the zarterm.doc file.

LIST OF DELIVERED FILES IN THE ZARPAW DIRECTORY:

Documentation : zarpaw.doc

zar980 program: servzar.c (source in C language) zar980.c (″ ″)

pactypes.h (header for zar980)

scrstruc.h (″ ″)

zar980.h (″ ″) comp_zar980 (compilation script)

LIST OF DELIVERED FILES IN THE ZARTERM DIRECTORY:

Documentation : zarterm.doc

zar980 program : zar980.c (source of display manager in C language)

pacparam.h (header for zar980)

pactypes.h (″ ″)

scrstruc.h (″ ″)

zar980.h (″ ″)

comp_zar980 (compilation script) (in normoal oprating mode)

Tests of the terminal:

touche.c (keyboard tests: C source)

video.c (screen tests: C source) Executable program and objects: touche (keyboard tests: executable program)

video (screen tests: executable program)

zar980.o (display manager: object)

Executable programs and objects are delivered for the following UNIX systems:

AIX with COBOL Micro-Focus Server Express 2.0.10 in the AIX/2010 sub-directory

HP-UX 10.20 with COBOL Micro-Focus HP 11.30 in the HP-UX/113 sub-directory

IRIX with COBOL Micro-Focus 3.1.39 in the IRIX sub-directory

OSF1 4.0 with COBOL Micro-Focus 4.1.10 in the OSF1/4110 sub-directory

OSF1 5.1 with COBOL Micro-Focus Server Express 2.0.11 in the OSF1/2011 sub-directory

SunOS with COBOL Micro-Focus 4.0.20 in the SunOS sub-directory

Chapter 2. Compilation modes

Implementation

Compilation of the Zar980 C program

Three compilation modes are suggested:

- the debug mode, using the ANIMATOR Cobol debugger,
- the standard operating mode,
- the degraded mode which does not take into account any function key.

NOTE: In the following descriptions, <ansi flags> corresponds to the options which allow the compilation in C-ANSI mode.

Debug Mode compilation

In debug mode, compile zarterm.c with the deletion of the link editer call (zarterm.o file keeping):

```
cc <ansi flags> -c -D_MF zarterm.c
```

and execute the following commands:

```
cob -x -e "" zarterm.o -lcurses
```

```
mv zarterm zar980
```

The result is the zar980 file which contains the Run Time System, the zarterm.o file, the curses library.

Then, to execute the COBOL program which calls zar980 in C, enter the following commands:

```
COBSW=+A-F
```

```
export COBSW
```

```
zar980 <program>
```

Operating Mode compilation

In normal operating mode, compile zarterm.c with the deletion of the link editer call (zarterm.o file keeping) :

```
cc <ansi flags> -c -D_CURSES zarterm.c
```

and execute the following commands:

```
cob -x -e ″″ zarterm.o -lcurses
```

```
mv zarterm zar980
```

the result is the zar980 file which contains the Run Time System, the zarterm.o file, the curses library.

Then, to execute the COBOL program which calls zar980 in C, enter the following command:

```
zar980 <program>
```

Degraded Mode compilation

In degraded mode, zarterm.c is compiled following the same principle as the other modes, but the command contains a compilation option (-D_NOKEYPAD) which inhibits the zar980 function keys:

```
cc <ansi flags> -c -D_CURSES -D_NOKEYPAD zarterm.c
```

and execute the commands:

```
cob -x -e ″″ zarterm.o -lcurses
```

```
mv zarterm zar980
```

The result is the zar980 file which contains the Run Time System, the zarterm.o file, the curses library.

Then, to execute the COBOL program which calls zar980 in C, enter the following command:

```
zar980 <program>
```

NOTE: On AIX, if the COBOL Run Time version is higher than 3.2.20, you must assign the LIBPATH environment variable before executing the ″cob″ command. ″cob″.

Example : LIBPATH=/lib:/usr/lib:$COBDIR/coblib

NOTE: On IRIX, SINIX, SunOS or OSF1, you must assign the LIBPATH variable before executing the ″cob″ command.

Example : LD_LIBRARY_PATH=$COBDIR/coblib

Chapter 3. Configuration of the workstation

WORKSTATION CONFIGURATION

For a correct using, you must configure the terminal keyboard, the function keys and the screen (colors and presentation attributes).

Terminal configuration

The zar980 program in, C language, has been developed by IBM to manage the screen and the keyboard in applications generated by the Dialog module in Cobol Micro Focus Unix.

The delivered C source must be compiled and link edited either with the generated Cobol screens or with the Run Time Systrem Micro Focus COBOL/2.

The zar980 program uses the curses Unix library to manage the keyboard and the screen, and conforms to the XPG3 standard (it uses only functions described in XPG3: X/OPEN PORTABILITY GUIDE 3).

The C source may also be compiled permitting the use of the Cobol Micro Focus Animator debbugger for the generated application.

In this case, the program uses the screen and keyboard routines supplied by Micro Focus and does not use the functions of the curses library.

Keyword configuration

The purpose of this chapter is to verify that all the function keys of your keyboard are correctly configured, and then to help you modify the configuration of your terminal if needed.

A PFkey is a key of the keyboard allowing the execution of a specific function, i.e. all the keys other than keys used to enter alphanumeric characters.

Function keys presentation

The zar980 program takes into account the function keys when they are described in a Unix database, named terminfo.

The terminfo database contains the configuration of different types of terminals.

The configuration of a terminal is composed of attributes.

Each attribute is characterized by a name and a value; the value can be:

- A boolean indicates that the type of terminal has or not the corresponing characteristic;
- An integer specifies a particular characteristic for the type of terminal;
- A string corresponds to the sequence of characters to send to the terminal to implement a particular function.

To work with the configurations described in terminfo, the zar980 program uses a functions library, curses.

The following schema summarizes the operations flow:

```
+----------------+
|  function key  |
+----------------+
  |

  |
  |     +--------------------+    +--------------------+
  |     |    characters      |    |      terminfo      |
  +---> |    sequence        |--->|      database      |<---+
        +--------------------+    +--------------------+    |
                                                            |
                                                            |
                                                            |
        +--------------------+    +--------------------+    |
        |    zar980          |    |       curses       |    |
        |    program         |<-->|       library      |<---+
        +--------------------+    +--------------------+
```

N.B. : For more information on terminfo, enter the ″man terminfo″ command (if your online manual contains information on terminfo).

The zar980 program uses the following function keys:

attribute name in curses(X).h	**attribute name in terminfo**	**description/interpretation in the zar980 program**	
KEY_UP	kcuu1	up arrow moves the cursor to the top of the screen	(1)
KEY_DOWN	kcud1	down arrow move the cursor to the bottom of the screen	(2)
KEY_LEFT	kcub1	left arrow moves the cursor to the left of the screen	(3)

attribute name in curses(X).h	attribute name in terminfo	description/interpretation in the zar980 program	
KEY_RIGHT	kcuf1	right arrow moves the cursor to the right of the screen	(4)
KEY_HOMET	khome	oblique arrow in general moves the cursor in the first input field of the screen	(5)
KEY_LL	kll	END key in general moves the cursor in the last input field of the screen	(6)
KEY_IC	kich1	INS key in general activates/daactivates the insert mode	(7)
KEY_DC	kdch1	DEL key in general deletes a characater	(8)
KEY_BACKSPACE	kbs	backspace key deletes a character	(9)
KEY_ENTER	cud1	ENTER key	(10)
KEY_F(0)	kf0	PFkey associated to the PF0 function of zar980	(11)
KEY_F(1)	kf1	PFkey associated to the PF1 function of zar980	(12)
KEY_F(2)	kf2	PFkey associated to the PF2 function of zar980	(13)
KEY_F(3)	kf3	PFkey associated to the PF3 function of zar980	(14)
KEY_F(4)	kf4	PFkey associated to the PF4 function of zar980	(15)
KEY_F(5)	kf5	PFkey associated to the PF5 function of zar980	(16)
KEY_F(6)	kf6	PFkey associated to the PF6 function of zar980	(17)
KEY_F(7)	kf7	PFkey associated to the PF7 function of zar980	(18)
KEY_F(8)	kf8	PFkey associated to the PF8 function of zar980	(19)
KEY_F(9)	kf9	PFkey associated to the PF9 function of zar980	(20)
KEY_F(10)	kf10	PFkey associated to the PF10 function of zar980	(21)

attribute name in curses(X).h	**attribute name in terminfo**	**description/interpretation in the zar980 program**	
KEY_F(11)	kf11	PFkey associated to the PF11 function of zar980	(22)
KEY_F(12)	kf12	PFkey associated to the PF12 function of zar980	(23)
KEY_F(13)	kf13	PFkey associated to the PF13 function of zar980	(24)
KEY_F(14)	kf14	PFkey associated to the PF14 function of zar980	(25)
KEY_F(15)	kf15	PFkey associated to the PF15 function of zar980	(26)
KEY_F(16)	kf16	PFkey associated to the PF16 function of zar980	(27)
KEY_F(17)	kf17	PFkey associated to the PF17 function of zar980	(28)
KEY_F(18)	kf18	PFkey associated to the PF18 function of zar980	(29)
KEY_F(19)	kf19	PFkey associated to the PF19 function of zar980	(30)
KEY_F(20)	kf20	PFkey associated to the PF20 function of zar980	(31)
KEY_F(21)	kf21	PFkey associated to the PF21 function of zar980	(32)
KEY_F(22)	kf22	PFkey associated to the PF22 function of zar980	(33)
KEY_F(23)	kf23	PFkey associated to the PF23 function of zar980	(34)
KEY_F(24)	kf24	PFkey associated to the PF24 function of zar980	(35)

**** TABLE 1 ****

NOTE: On COMPAQ TRUE 64 system, only 10 PFkeys with the type KEY_F(i) (i=0,...,24) can be assigned.

PFkeys assignment

The 'touche' program can be used in a large range of terminals. Its purpose is to help you in the configuration of your terminal (C source and executable programs are delivered).

Before modifying the terminfo database, you must :

- save the terminfo directory,
- know the processing of the production utility of the terminfo database (tic UNIX command, in general),
- ask the hardware vendor:
  - which terminal emulation is the most adapted to your terminal (content of the TERM Unix environment variable),
  - if your terminal can support all the attributes used by zar980 (see TABLE 1),
  - the procedure to modify the database,
  - the characters sequence sent by the PFkeys,

Note: the touche program may have an impredictable behaviour if the terminal, where it is executed, is not correctly configured.

Purpose of the touche program

The touche program allows you to:

- check that, for a given configuration of the terminal, the PFkeys are recognized in terminfo (choice 1: touche configuration in terminfo),
- know the characters sequence of PFkeys which do not contain the Ctrl sequence (choice 0: touche sequence)
- know the Ctrl sequence of particular PFkeys (choice 2: Ctrl-x sequence).

Implementation of touche program

Enter the command : touche -f (-f for the french version)

then press <RETURN>

The executable program displays:

Your terminal is configued in xxx

Touche sequence enter 0 <RETURN>

Touche configuration in terminfo enter 1 <RETURN>

CTRL-x sequence enter 2 <RETURN>

with xxx = VT, VT52, VT100, VT102, VT125, VT200, VT220, VT240, VT241, VT300, VT320, VT330, VT340, VT400, VT240, VT241, VT300, VT320, VT330, VT340, VT400, VT420, LK201, LK401, xterm ...

In the 1 and 2 choices, the PFkey activation directly sends back the requested information. In the 0 choice, the PFkey activation must be followed by <RETURN>.

Program exit: press the space bar, then press <RETURN>.

Way of using the touche program

First check that the PFkeys are recognized in terminfo for the active emulation on your terminal.

Choose th option 1 of the touche program, ther press each PFkey described in the column 3 of the TABLE 1:

- if the key returns the attribute of curses(X).h (column 1 of TABLE 1), the key has its matching in terminfo,
- if not, try several types of X terminal emulation (set the TERM variable to different values), then keep the emulation the most adapted to your terminal.

Then, if all PFkeys are not taken into account in terminfo or not correctly configured (no matching between the columns 1 and 3) or if the key described in column 3 does not exist on your keyboard (in that case replace this PFkey by another one which is not described in the TABLE 1), launch the touche executable program, select the option 0 or 2 and note the sequence of characters, sent by each PFkey, which will be taken into account in terminfo.

Recognition of PFkeys in terminfo

It is highly advisable to keep the existing configurations in terminf as they are, and to create another configuration using an existing terminal configuration.

Example :

You ahve tested your terminal with the touche program in vt300 configuration, and you have received the following message:

Your terminal is configured in vt300

Touche sequence enter 0 <RETURN>

Touche configuration in terminfo enter 1 <RETURN>

CTRL-x sequence enter 2 <RETURN>

You have tested the configuration of your PFkeys at terminfo level by using the option 1 of the menu above.

The result is as follows:

attribute name in curses(X).h	**description of the key used by zar980**	
KEY_UP	up arrow	(1)
KEY_DOWN	down arrow	(2)
KEY_LEFT	left arrow	(3)
KEY_RIGHT	right arrow	(4)
	oblique arrow in general	(5)
	END key in general	(6)
	INS key in general	(7)
	DEL key in general	(8)
KEY_BACKSPACE	backspace key	(9)
KEY_ENTER	ENTER key	(10)
KEY_F(0)	PFkey associated to the PF0 function of zar980	(11)
KEY_F(6)	PFkey associated to the PF6 function of zar980	(12)
KEY_F(2)	PFkey associated to the PF2 function of zar980	(13)
.	.	.
.	.	.
.	.	.
KEY_F(24)	PFkey associated to the PF24 function of zar980	(35)

In this table, we note that:

- the (1), (2), (3), (4), (9), (10), (11), (12), (13),..., (35) keys are declared in the terminfo database,
- the (12) key is not correctly defined,
- the (5), (6), (7), (8) keys are missing in terminfo for a terminal defined in vt300.

The following operation consists in the determination of the characters sequence sent back by each PF key (5), (6), (7), (8), (12).

After the touche program execution with the option 0, here is the result table:

sent sequence of characters by the PFkey	**description of the key used by zar980**	
\E[1~	oblique arrow in general	(5)
\E[4~	END key in general	(6)
\E[2~	INS key in genrarl	(7)
\E[3~	DEL key in general	(8)
\E[18~	PFkey associated to the PF1 function of zar980	(12)

After this operation, you have to create the following sequence in the source file of termninfo database:

vtcgi

khome=\E[1~,

kll=\E[4~,

kich1=\E[2~,

kdch1=\E[3~,

kf2=\E[18~,

use=vt300,

Your new configuration is named vtcgi and inherits the vt300 configuration.

You will have to declare at user login level, in the .login or .profile environment files, the TERM variable:

export TERM=vtcgi

Screen configuration

The verification of the screen consists in verifying that the management of intensity and presentation attributes is correctly performed by the terminal.

The zar980 program processes the following attributes:

```
INTENSITY                         PRESENTATION
.Bright (Highlighted)             .Reverse video
```

```
.Normal intensity                    .Blinking
.Underlined                          .Color
```

As for PFkeys, the terminfo database can contain their presentation:

attribute name in curses(X).h	attribute name in terminfo	description	
A_BLINK	blink	blinking	(1)
A_BOLD	bold	bright (highlighted)	(2)
A_DIM	dim	semi-intensity	(3)
A_REVERSE	rev	reverse video	(4)
A_UNDERLINE	smul	underlined (beginning)	(5)
	rmul	underlined (end)	

**** TABLE 2 ****

Restriction: on hpterm terminal, the blinking mode is not supported.

The 'video' program (C source and executable program are delivered) displays a list of labels with different intensity and presentation attributes. You can therefore test if your terminal supports all these attributes.

To the attributes listed above, zarterm manages the color attributes defined in the terminfo database:

attribute name in curses(X).h	attribute name in terminfo	description
COLOR_WHITE	colf0	blank character
COLOR_RED	colf1	red character
COLOR_GREEN	colf2	green character
COLOR_YELLOW	colf3	yellow (brown) character
COLOR_BLUE	colf4	blue character
COLOR_MAGENTA	colf5	magenta (pink) character
COLOR-CYAN	colf6	cyan (turquoise) character
COLOR_BLACK	colf7	black character
	colb0	black background
	colb1	red background
	colb2	green background
	colb3	yellow background

attribute name in curses(X).h	attribute name in terminfo	description
	colb4	blue background
	colb5	magenta background
	colb6	cyan background
	colb7	blank background

Chapter 4. Set of tests

Set of tests introduction

The MBUPDT.ZAR file contains an example pf dialog (MA-coded), in batch transactions format. To perform the suggested tests, the set of tests provided in MBUPDT.ZAR must be integrated in your specifications database by the batch update procedure (UPDT).

The purpose of the set of tests is to be familiar with the zar980 use and to test the keyboard-screen environment.

Compilation of the dialog

After the generation of the MA dialog and its screens:

MA00ME General

MA00M1 Menu 1 (screen branching via operation code)

MA00M2 Menu 2 (screen branching via PFkeys)

MA01CA Tape definition (menu 1)

MA01FI List of movies (menu 1)

MA02CA Tape definition (menu 2)

MA02FI List of movies (menu 2)

MAHELP Online help screen

Compile these programs the following comp_cobol shell script:

COBOPT='-C NOANIM

-C ASSIGN=EXTERNAL

-C NOBOUND

-C DEFAULTBYTE=32

-C NOLIST

```
-C NESTCALL

-C SEQUENTIAL=LINE

-C NOTRACE

-C NOTRUNC

-C NOWARNING

-N NOHPOPTIMIZE (on HP)

-N NOCHECK (on AIX/BOSX)

-N NOBOUNDOPT' (on AIX/BOSX)

export COBOPT

cob -v -u $COBOPT MA.cbl

cob -v -u $COBOPT MA00ME.cbl

cob -v -u $COBOPT MA00M1.cbl

cob -v -u $COBOPT MA00M2.cbl

cob -v -u $COBOPT MA01CA.cbl

cob -v -u $COBOPT MA01FI.cbl

cob -v -u $COBOPT MA02CA.cbl

cob -v -u $COBOPT MA02FI.cbl

cob -v -u $COBOPT MAHELP.cbl
```

Then copy the produced .gnt files in a directory referenced by the COBPATH variable.

NOTE: on AIX, if the COBOL run time version is higher than 3.2.20, you must assign the LIBPATH environment variable before the ″cob″ command startup.

Example : LIBPATH=/lib:/usr/lib:$COBDIR/coblib

On IRIX , SINIX , SunOS or OSF1, you must assign the LIBPATH variable before the ″cob″ command startup.

Example : LD_LIBRARY_PATH=$COBDIR/coblib

<u>Application startup</u>

Execute the testma commands file which assigns the necessary files, then start the MA dialog.

NOTE: On AIX, if the COBOL Run Time version is higher than 3.2.20, you must assign the LIBPATH environment variable before the testma commands file startup.

Example : LIBPATH=/lib:/usr/lib:$COBDIR/coblib

On IRIX , SINIX , SunOS or OSF1, you must assign the LIBPATH variable before the testma commds file startup.

Example : LD_LIBRARY_PATH=$COBDIR/coblib

The files necessary for the application are the following:
- fparam zar980 parameters file (provided)
- LE application error labels file (provided)
- HE backup file if call of the help (created by the application)
- CA video tapes file (created by the application)
- VI movies file (created by the application)

The dialog is very simple, you can consult or edit a video tapes file and a movies file. It is divided into two branches, one allowing a screen branching in the application via operation code input, the other via PFkeys.

Input the operation code and the action code is required and they must be entered in uppercase.

The operation code values are described in the screens; the action code values are:

C (creation), M (modification), A (deletion).

Online help call : PF6, data element: PF7

Tab: tabs key or Ctrl and i keys pressed simultaneously

In the list, the number in brackets, which follows the listed keys, refers to the TABLE 1 number.

Tests for keyboard configuration

The purpose of this section is to verify the behavior of your keyboard keys with the provided set of tests. For each key described in table 1, you must have the same behavior with the key of your keyboard.

These tests are to be performed in two steps: screen branching in the application via the 'OPERATION CODE' field, test of cursor moving keys, and then test of PFkeys.

Screen branching in the application via 'OPERATION CODE'

At application startup, the following menu appears:

```
                                  GENERAL MENU
                                  ============
                        M1 : MENU 1 (operation code)
                        M2 : MENU 2 (function key)
                        FT : END OF TRANSACTION
                                  OPERATION CODE:
```

The cursor is placed in the 'OPERATION CODE' field.

Enter M1 in this field to display the following menu:

```
                                  TAPE MENU1
                                  ==========
                     CA : TAPE DEFINITION
                     FI : LIST OF MOVIES
                     ME : BACK TO GENERAL MENU
                     M1 : BACK TO VIDEO TAPE MENU1
                     FT : END OF TRANSACTION
OPERATION CODE:              TAPE NUMBER:
```

Then in the screens:

- MA = Update
- SU = Display continuation

You will create a tape definition. The cursor is always placed in the 'OPERATION CODE'. Enter CA and 001 (tape number) :

```
                      TAPE DEFINITION

ACTION CODE.......:
TAPE NUMBER.......: 001           TAPE TYPE.........:
                                  TAPE TRADEMARK....:
                                  TAPE DURATION.....:
                                  REMAINING TIME....:
```

```
A PO MOVIE TITLE                  GE DUR CO V

ME=MENU, M1=MENU1, CA=TAPE DEF., FI=MOVIES LIST, MA=UPDATE
 END OF LIST

OPERATION CODE: MA          TAPE NUMBER:
```

Enter the following information:

```
ACTION CODE.......: C
TAPE NUMBER.......: 001           TAPE TYPE.........: VHS
TAPE TRADEMARK....: TDK
TAPE DURATION.....: 240
REMAINING TIME....:

A PO MOVIE TITLE                            GE DUR VE V
C 01 Robin des Bois                         AV 120 VF 4
C 02 Metropolis                             FI 110 VO 4
```

Place the cursor in the 'OPERATION CODE' field and enter FI (list of movies):

```
LIST OF MOVIES

A NUM PO MOVIE TITLE                       GE VE DURV
  001 01 Robin des Bois                    AV VF 1204
  001 02 Metropolis                        FI VO 1104
ME=MENU, M1=MENU1, CA=TAPE DEF., FI=MOVIES LIST, MA=UPDATE
END OF LIST

OPERATION CODE: MA           TAPE NUMBER:
```

Use this screen to test the 'arrow' keys (1 to 4) to move the cursor. You can continue your tests in the two screens above for the consultation and the update.

When the first set of tests is finished, go back to the general menu via the ME 'OPERATION CODE':

```
                                 GENERAL MENU
                                 ============
                       M1 : MENU 1 (operation code)
                       M2 : MENU 2 (PFkey)
                       FT : END OF TRANSACTION
                                 OPERATION CODE:
```

<u>Screen branching in the application via PFkeys</u>

Go to the Video menu2' :

```
                  VIDEO MENU2
                  ===========
PF-03 : TAPE DEFINITION
PF-04 : LIST OF MOVIES
PF-01 : BACK TO GENERAL MENU
PF-02 : BACK TO VIDEO MENU2
PF-05 : END OF TRANSACTION
```

```
Then, in the screens:  ENTER = Update
                       PF-08 = Display continuation
                       PF-06 = Screen documentation
                       PF-07 = Data element documentation
                       TAPE NUMBER:
```

From this menu, you can test the proposed PFkeys.

The input in the screens is the same than for the 'Video Menu1' screens.

Tests of screen configuration

the purpose of this section is to verify that the intensity and presentation attributes are correctly taken into account by the terminal.

The 'GENERAL MENU' screen has different intensity and presentation attributes supportéd by the zar980 program:

- the 'GENERAL MENU' title is highlighted and cyan- colored,
- the choice 1 (menu 1) is in normal intensity, underlined and yellow-colored,
- the choice 2 (menu 2) is in normal intensity, blinking and blue-colored,
- the choice 3 (end) is highlighted, underlined and red-colord,
- the 'OPERATION CODE' is in reverse video and green- colored,
- the first input field is not visible.

You can modify these attributes and the other screens ones in the specifications database, and then re-generate the Dialog for complementary tests. Then, if you are not satisfied with the display of these different attributes, you can modify the zar980 parametrs file (FPARAM).

List of UNIX systems where zar980 has been tested.

AIX

BOS

BOSX

HP-UX

IRIX

NCR

OSF1

SINIX

SunOS

ULTRIX

Part Number: DUSRC300021A - 6599

Printed in USA