1 Preface

Increasingly, mainframe computers are being used together with PCs or multiuser systems in heterogeneous environments. This configuration combines the fast processing speeds of central mainframes with the individual nature of personal computers, which are nowadays generally equipped with user-friendly graphical interfaces to facilitate working with applications.

1.1 Characteristic features of WIN-DOORS

WIN-DOORS is the graphical interface for all BS2000/OSD applications. A simple mouse click often replaces the tedious process of entering text. The number of input errors is reduced and the application is far easier to use. All applications have a uniform interface, so that users can concentrate on their work rather than on the mechanics of using the application.

WIN-DOORS supports all BS2000/OSD form systems, so existing applications need not be rewritten. The structure of the screen forms is automatically converted to match the graphical interface.

WIN-DOORS runs under Microsoft Windows V3.1, Windows 95 and Windows NT.

In order to work with WIN-DOORS, you will need to have a DDE-compatible emulation for BS2000 installed on your PC. This emulation is the communications basis via which forms are displayed as graphically converted panels on a PC. All communication protocols supported by the emulation are also supported by WIN-DOORS. The DOORS emulation is included in the delivery package.

1.2 Structure and contents of the WIN-DOORS documentation

WIN-DOORS is aimed at BS2000 users who want to take advantage of the benefits provided by graphical interfaces for BS2000 applications.

The documentation for WIN-DOORS comprises the following components:

– the present manual, “WIN-DOORS, Graphical Interface for BS2000/OSD Applications”

– the manual “DOORS Emulation, 9750 Basic Emulation for BS2000/OSD Connections”

– the manual “WIN-DOORS/FHS-DOORS, Optimizing Panels Using the DOORS Editor”

– online help for WIN-DOORS

– online help for the DOORS emulation

– online help for the DOORS editor

To work with an application under WIN-DOORS, you require basic knowledge of how to operate graphical interfaces. You will find an introduction to the Microsoft Windows graphical user interface and a description of how to work with it in the Microsoft Windows documentation.

WIN-DOORS, Graphical Interface for BS2000/OSD Applications

This manual describes the functionality provided by WIN-DOORS. It covers how to use WIN-DOORS, OLE automation, and the installation and configuration of WIN-DOORS.

If you are using WIN-DOORS for the first time, you should first read chapter “WIN-DOORS - graphical interface for BS2000/OSD applications” on page 7ff and chapter “Using WIN-DOORS” on page 13ff.

This manual is intended for end users who work with host applications under WIN-DOORS.

A detailed description of the WIN-DOORS interface can be found in the online help for WIN-DOORS.

DOORS Emulation, 9750 Basic Emulation for BS2000/OSD Connections

The DOORS emulation manual describes the functions of the DOORS emulation. It tells you how to work with the DOORS emulation, how to set up connections (connection names), how to use startup scripts to automate the process of establishing a connection and how to install and configure the DOORS emulation.

It is intended for users who want to use the DOORS emulation for connecting to a BS2000/OSD partner.

A detailed description of the DOORS emulation interface can be found in the online help for the DOORS emulation.

WIN-DOORS also works with other 9750 emulations which support the DOORS-DDE protocol (e.g. SNI’s MT9750 emulation). These can be used in place of the DOORS emulation.

WIN-DOORS/FHS-DOORS, Optimizing Panels Using the DOORS Editor

The DOORS editor manual describes the options available for optimizing the graphical interfaces (panels) generated automatically by WIN-DOORS using the DOORS editor. It tells you how to work with the DOORS editor, explains the user-specific extensions under Dialog Builder and describes how to install and configure the DOORS editor.

The manual is aimed solely at those responsible for optimizing panels.

A detailed description of the DOORS editor interface can be found in the online help for the DOORS editor.

README files

Details of any functional changes and additions to this manual and the online help system are included in README files named *readme.txt*. The file is located in the installation directory for the relevant products. You can view the files in an editor or print them out on a standard printer.

1.3 Changes since WIN-DOORS V3.0

- **Capture**
  The Capture mechanism allows you to define an identifier for each form. This enables WIN-DOORS to uniquely identify the form. The Capture mechanism can also be used to create subpanels for a form. If a form template is available for a form, WIN-DOORS will use it to display the panel.
- **Alphanumeric object**
  When using forms with several I/O fields, performance can be improved by combining multiple forms for a panel into a single alphanumeric object. This does not change how fields are displayed in the panel or processed by the user; however, less system resources are required to display the alphanumeric object.
- **Interfaces**
  Dynamic forms library
  A customized algorithm for forms recognition can be made available to WIN-DOORS by means of a dynamic library (DLL) for forms recognition. A skeleton program to write this DLL is supplied with the product. This program includes predefined functions that can evaluate the data of a form and process it as specified.

1.4 Notational conventions

The following conventions are used in this manual:

In syntax:

bold characters	Constants: these characters are to be entered exactly as printed.
regular characters	Variables: these characters stand for other characters that you select and enter.
␣	Mandatory blank.
[ ]	Optional: arguments in square brackets are optional. The brackets themselves must not be specified.
{ | }	Braces enclose alternatives separated by the logical OR character. The braces themselves and the OR character must not be specified.

In the body of the text:

“double quotes”	Names of chapters/sections and terms requiring special emphasis.
italics	File names and terms which appear on the screen (such as functions, menus, field names or pushbuttons).
[number]	Reference to a manual listed in the “Related publications” section.
►	An action that you have to take.
$\boxed{\text{Key}}$	Indicates a key on the keyboard.
$\boxed{\mathbf{i}}$	A note providing you with additional information.

It is a characteristic of a graphical interface that functions are no longer initiated by typing in commands but can be selected from menus. In order to make it easier for you to find a particular function in this manual, we have used a slash to separate the menu title from the menu item. For example, *Edit/Copy* means that the *Copy* function is to be selected from the *Edit* menu. An arrow (→) in a command indicates a cascade menu.

1.5 Terms used

The terms *form* and *panel* have special meanings in this manual. In the past, the terms “mask” and “graphical mask” or “graphical form” have been used for these two terms.

A *form* is an alphanumeric form from a host application (e.g. an FHS form or a FORMPLAG form etc.).

A *panel* is a graphical form on the PC and is stored in a semantics file (sdc file). The sdc file is generated from a form with WIN-DOORS.

The term *resource path* is used synonymously with *resource directory*.

2 WIN-DOORS - graphical interface for BS2000/OSD applications

WIN-DOORS allows BS2000 to support graphics. This applies to all BS2000/OSD applications.

WIN-DOORS allows you to make use of the advantages of a graphical interface without the need to rewrite your existing host applications. The structure of the forms is automatically converted to graphical windows (panels).

It is possible to migrate to WIN-DOORS step by step. It is thus possible to run PCs with graphical interfaces in parallel with alphanumeric terminals.

Forms are converted to the graphical interface automatically using default values (e.g. background colors, fonts and borders). The DOORS editor then allows you to optimize the converted forms (panels). Important items can be highlighted in color and related items can be clearly grouped. Customization allows the graphical interface to be tailored exactly to your own requirements.

WIN-DOORS features:

- **A graphical interface for all BS2000/OSD forms systems**
  There are no special requirements with regard to the form system used
- **System stability**
  No changes to the existing application are required
- **Templates**
  Automatic conversion of forms using freely-definable, user-specific templates
- **Macros**
  Automation of work processes
- **Ergonomics, efficiency and individuality**
  Optimization of the panels using the DOORS editor
- **Graphics where they make sense**
  Mixed use of graphical and alphanumeric screens
- **OLE automation**
  Control of work procedures using, for instance, a Visual Basic script or a C++ program

2.1 Areas of application for WIN-DOORS

WIN-DOORS can be used with any BS2000/OSD form system and with a connection to any UTM, TIAM or DCAM application or task.

When you establish a connection to a host application with WIN-DOORS, the system first checks whether the host application operates in full-screen mode (i.e. forms) or line mode. If it uses full-screen mode, the forms are displayed in the WIN-DOORS application window. If it uses line mode, output is sent to the emulation window of the emulation used to establish the connection or to the WIN-DOORS application window.

Automatic conversion of forms using templates

The forms are automatically converted to panels on the PC at runtime. Templates mean that this process can be controlled using freely-definable, user-specific templates. This means that all the forms can be linked to a "common" template, or each form can be linked to an "individual" template.

The templates are created directly with WIN-DOORS and saved in a file. Generation of the templates requires no programming since it is carried out in simple dialogs. Templates can be made available to all users once they have been saved.

Gradual optimization

After the panels have been converted automatically, each of them can be further optimized. This optimization process can be carried out at runtime (online) or offline. WIN-DOORS includes an object-oriented editor (DOORS editor) for this optimization process. The optimized panels are stored in files (sdc files) and can then be made available to all users. Optimization requires no programming since it is carried out in simple dialogs

Filling in fields automatically using macros

Macros allow you to automate individual steps, e.g. by automatically filling in certain fields and sending the panel to the host application. Macros thus relieve users of tiresome, repetitive tasks. WIN-DOORS provides an easy way of recording macros.

Macros can be started using shortcuts or menus or automatically as soon as the panel is displayed.

Controlling execution with OLE automation

Instead of controlling the host application from panels, entire work processes can be automated with a Visual Basic script or a C++ program using the OLE objects provided by WIN-DOORS.

It is thus possible, for instance, to collect data from various forms, from different host applications and from different hosts and show this data in a single Excel chart. The various forms called are no longer displayed. Instead of the underlying host application, the user then only sees the interface of the Visual Basic script or the C++ programs which control a particular process.

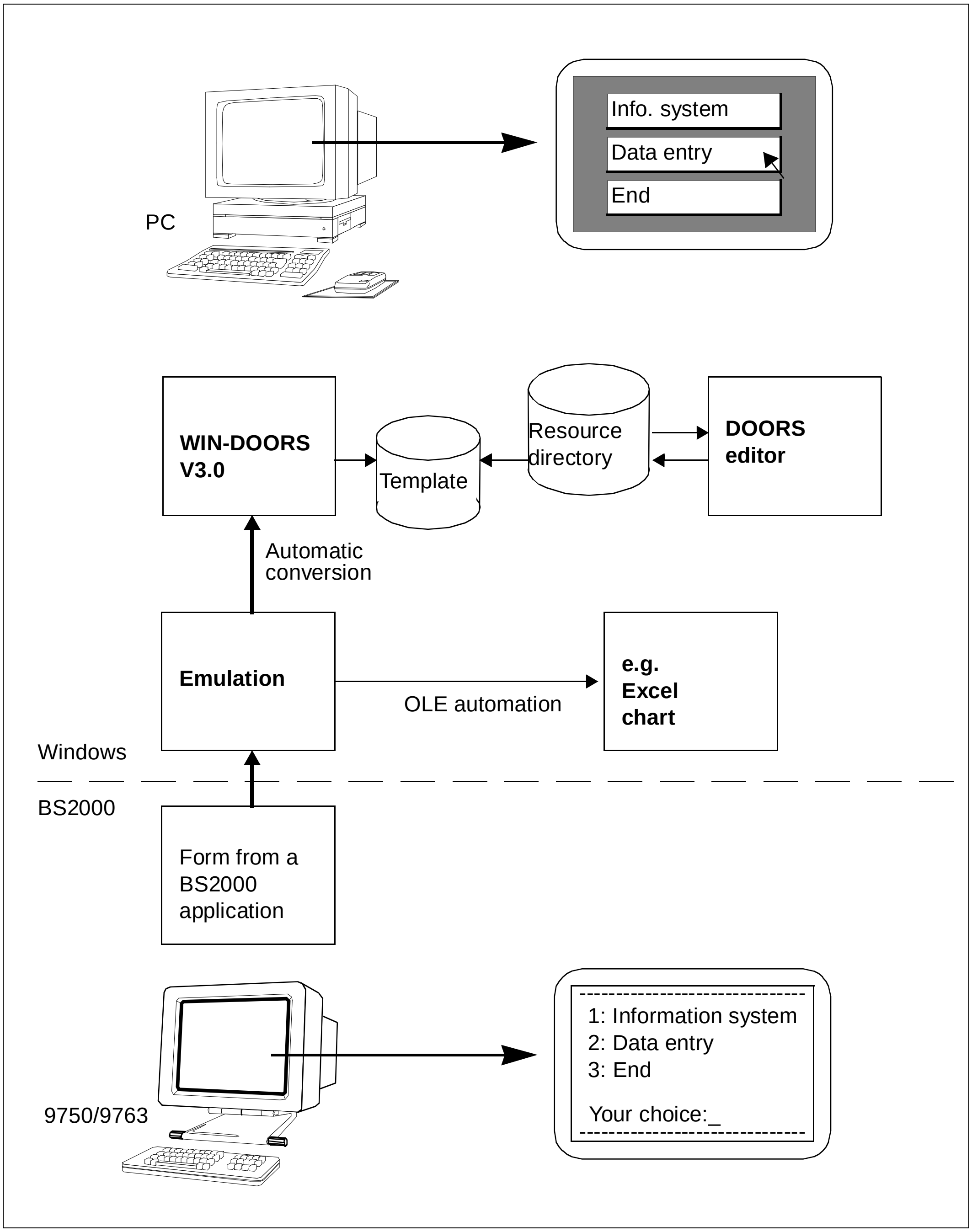


Figure 1: WIN-DOORS model

2.2 Optimizing panels with the DOORS editor

The DOORS editor allows you to optimize panels on the PC. You can, for instance, improve the appearance of a panel or emphasize important data. In short, you can adapt panels to suit the requirements of your organization. Refer to the manual "WIN-DOORS/FHS-DOORS, Optimizing Forms Using the DOORS Editor" [2] for a description of how to customize panels and the objects WIN-DOORS provides for this purpose.

Panels can be optimized step by step. If, for instance a host application provides 100 panels, but users generally only work with 10 of these, it may make sense to first optimize these 10 panels and to leave the remainder unchanged or only make minor changes. The basic conversion carried out automatically means that even panels which have not been optimized have the typical Windows appearance.

Testing panels

Once you have optimized panels with the DOORS editor, you should test the optimized panels to ensure that they work correctly with the host application under WIN-DOORS. You can activate debugging mode in WIN-DOORS to help you find any errors which may be present.

3 Using WIN-DOORS

3.1 Sessions and connections

WIN-DOORS uses so-called sessions for all connections to the host applications. A session contains certain parameters for the session which can be set with the *Setup* menu as well as the parameters indicating the host application and host computer to which a connection is to be established, the emulation to be used and the panels to be used (resource path). All these parameters can be configured as required for each session. The parameters can be stored in a “parameter file” so that the session does not need to be reconfigured every time you use it.

WIN-DOORS allows you to:

- open new sessions
- open saved sessions
- edit saved sessions
- close open sessions
- save open sessions

WIN-DOORS distinguishes three different statuses:

- “no session open”
- “session open, no connection established”
- “session open, connection established”

Generally, you will want to open a session and establish the connection immediately. Do this by clicking on *Start* in the relevant dialog boxes. WIN-DOORS then switches to the status “session open, connection established”.

If, however, you first want to edit/modify a saved session, you must open the session, but you must not establish the connection. Click on *OK* to open the session. WIN-DOORS then switches to the status “session open, no connection established”.

3.1.1 Parameter files for sessions

You can save the parameters of a session in parameter files in order to open that session at a later point in time or directly on starting WIN-DOORS (see “Starting a session directly” on page 28).

Parameters that are applicable to all sessions are stored in the *wind.ini* file in the Windows directory, while those that are only applicable to a special session are stored in the *.drs* file in the WIN-DOORS directory. The parameters in these two files can be edited to some extent either with menu commands or with an editor.

The parameter file for a session is subdivided into individual sections under various headers. Each of the parameters under a specific section is listed in a separate line together with its corresponding value. The following sections are included in the *wind.ini* file:

Section header	**Meaning**
misc	Contains parameters that affect WIN-DOORS settings
WindowState	The parameters in this section are managed by WIN-DOORS. They control the windows displayed by WIN-DOORS.
Recent File List	Contains the last four opened sessions; these parameters are managed by WIN-DOORS

Table 1: Structure of the *wind.ini* file

The following sections can be found in the *.drs* file:

Section header	**Meaning**
connection	Contains the connection parameters
misc	Contains parameters that determine how panels are displayed
Custom	Parameters for earlier DOORS versions
font	Parameters that define the font
debug	Contains just one parameter for debugging
display attributes	Parameters for converting BS2000 display attributes
alternate colors	Parameters for converting BS2000 display attributes with alternative colors
secondary windows	Contains parameters that determine the behavior of secondary windows (subpanels)
UsrDlls	Contains parameters for the dynamic library (DLL) that is read by WIN-DOORS when converting commands and special keys for buttons

Table 2: Structure of the *.drs* file

Section header	**Meaning**
Help	Parameters to call external Windows Help files
SpecialKeysSyntax	Parameters to convert special keys
popup	Contains parameters that can be used to control pop-up windows from “natural” applications

Table 2: Structure of the *.drs* file

If you want to edit or modify the parameters of a saved session before starting it, you will need to open the session, but without establishing the connection. To open a session, select the *OK* button. WIN-DOORS will then switch to the status “Session open; no connection established”.

The number of sessions that can be set up simultaneously depends on the used emulation, but only one connection can exist for each instance of WIN-DOORS.

A connection can be established only if:

– an emulation has already been started or

– the emulation is specified in the emulation parameter on opening a session directly and thus started automatically with it.

When WIN-DOORS is started (without specifying a parameter file), the initial WIN-DOORS status shows “no session opened”, i.e. only the primary window is displayed.

3.1.2 Parameters for all sessions (wind.ini)

The following parameters in the *misc* section of the *wind.ini* file can be modified by means of menu commands or a dialog box:

Parameter	**Menu command/Dialog box**
AutoTab	*Setup/Options->Miscellaneous*
DiagnosticsAlwaysOnTop	*Setup/Options->Miscellaneous*
DisableAutosend	*Tools/Macro*
Overwrite	*Setup/Options->Miscellaneous*
Statusbar	*Setup/Statusbar*
Toolbar	*Setup/Toolbar*

Table 3: Parameters of the *wind.ini* file that can be set via menu commands or a dialog box

A description of these commands can be found in the online help for WIN-DOORS. The two parameters listed below are also in the *misc* section of the *wind.ini* file, but can only be set with an editor such as Microsoft Notepad:

`OverwriteWhenSending={0|1}`
: Defines whether the Insert mode for the keyboard is to be turned off when sending data to the host application.
The default value is 0.

`Trimodal={0|1}`
: Defines whether or not WIN-DOORS uses a keyboard driver for trimodal keyboards.
The default value is 0.

Example

```
[Font]
FaceName=FHS DOORS
Style=Regular
Size=-13
Clip=2
Out=1
Quality=1
Family=49

[Debug]
Diagnosis=0
Semantics=0
; When this flag is set, a trace of the I/Os is taken in
; the file c:\windows\fhsddump.log
Dump=0

[Display Attributes]
Normal=0
Underlined=4
Bold=8
Flashing=17

[Alternate Colors]
Background=12632256
Foreground=255

[Misc]
AutoTab=1
Overwrite=1
StatusBar=1
3DLook=1
ShowFullMenus=0
Language=1
```

```
;
;Trimodal - When this flag is set, pressing the DEL key on the numeric keypad
; while holding the CTRL key down outputs a comma.
Trimodal=1
;
; CallMapScriptOnce - Diabuild „map“ script are called when a window is first
; displayed, but also when the BS2000 updates the screen. The following flag
; can be used to turn off this behavior.
CallMapScriptOnce=0
;
; Optimized810Protocol - This flag should be set when using some BS2000
; formatters (like FORMPLAG).
Optimized810Protocol=1
;
; ShowEmptyProtectedBoxes - Normally, an edit box, standing for a protected
; field which is either empty or holding only spaces or underscores, is
; automatically hidden. When this flag is set, these boxes stay visible.
ShowEmptyProtectedBoxes=0
ConstantLineHeight=0
ProcessLineMode=1
Mode=2107
ToolBar=1
DiagnosticAlwaysOnTop=1

[WindowState]
MainLeft=181
MainTop=36
MainWidth=638
Help=11 0 1 -1 -1 -1 -1 100 100 300 300

[Recent File List]
File1=C:\FHSDOORS\TUTORIAL\TUTORIAL.DRS
File2=C:\FHSDOORS\FHSD\PE.DRS
```

3.1.3 Parameters for different sessions (.drs file)

The following parameters in the *misc* section of the *.drs* file can be modified via menu commands or a dialog box:

Section header	**Parameter**	**Menu command/Dialog box**
connection	BufferAutoMngt	Setup/Session->Options
	BufferSize	Setup/Session->Options
	ConnectionId	Setup/Session or Session/New
	EmulationName	
	EmulationCmdLine	
	PromptBeforeDelete	Setup/Session->Options
	ResourcePath	*Setup/Session* or *Session/New*
misc	Alignment	Setup/Options->Graphical Tuning
	AutoResize	
	CharacterOverhead	
	ConstantLineHeight	
	HookDllPath	Tools/Capture
misc	LinesSpacing	Setup/Options->Graphical Tuning
	ObjectsSpacing	
	Optimized810Protocol	
	Overlapping	
	ProcessLineMode	Setup/Options->Miscellaneous
	RemoveTrailingSpaces	Setup/Options->Graphical Tuning
	3DLook	Setup/Options->Graphical Tuning
	ShwowEmptyProtectedBoxes	Setup/Options->Graphical Tuning
Custom	LookForWildLSPs	Setup/Options->Miscellaneous
font	Clip	Setup/Font
	FaceName	
	Family	
	Out	
	Quality	
	Size	

Table 4: Parameter of the *.drs* file that can be set via menu commands or dialog boxes

Section header	Parameter	Menu command/Dialog box
font	Style	Setup/Font
debug	Diagnosis	Setup/Options->Miscellaneous
display attributes	Bold	Setup/Options->Display Attributes
	Flashing	
	Normal	
	Underline	
alternate colors	Background	
	Foreground	

Table 4: Parameter of the *.drs* file that can be set via menu commands or dialog boxes

A description of these commands can be found in the online help for WIN-DOORS.

The parameters described below are also in the *.drs* file, but can only be set with an editor such as Microsoft Notepad.

The [connection] section

`AutomaticSwitching={0|1}`

If this parameter is set to a value of 1, every new panel of your application that is transferred will be displayed on the PC in the foreground.
The default value is 1.

`ConnectionTitle=`*string*

Title of the connection shown in the connection window. This parameter is evaluated only if the used emulation has an MDI interface.
If the used emulation has an MDI interface and you supply a value for this parameter as well as for *EmulationTitle*, the emulation window will not be displayed when running the application.
The default value is an empty string.

`DoorsTerminalType={9750|9755|9763}`

Defines the type of terminal on which the BS2000 application displays its output. WIN-DOORS can simulate various terminal types; however, if you are using FHS Version V8.1 or later or any version of Formplag as of Version V2.4, you must enter 9763 as the terminal type here.
The default value is the terminal type 9750.

`DataTransferRate=`integer `K`

Defines the buffer size for data transmission with the BS2000 host. The buffer size can be extended up to 28K. WIN-DOORS automatically informs the partner system which buffer size has been set.
The default value is 4K.

`EmulationTitle=`*string*

Title of the emulation shown in the emulation window. This parameter is evaluated only if the used emulation has an MDI interface.
If the used emulation has an MDI interface and you supply a value for this parameter as well as for *ConnectionTitle*, the emulation window will not be displayed when running the application.
The default value is an empty string.

The [misc] section

`AddDUEToEM={0|1}`

If this parameter is set, WIN-DOORS will also send the value of the DUE key to the host application whenever the EM key is pressed.
The default value is 0.

`CallMapScriptOnce={0|1}`

Defines whether a Dialog Builder script is executed by BS2000 when a panel is displayed for the first time or whenever a new screen is created.
The default value is 1.

`MaxEntryBoxes=`integer

Defines the maximum number of entry fields that may be included in a form so that the form can be automatically converted or the converted panel can still be displayed. Forms that contain more than the specified number of entry fields are displayed alphanumerically in the emulation window. You can define a maximum limit of 220 entry fields here.
The default value is 150.

`ShowMainMenu={0|1}`

Defines whether the primary window of WIN-DOORS is to be displayed when it is started directly. If the primary window is suppressed with the value 0, the individual commands will be available in a pop-up menu that can be opened with the right mouse button.
The default value is 1.

`ReplaceNilsBySpaces={0|1}`

Causes all null values in the form sent by the BS2000 application to be converted to spaces before being displayed in the panel.
The default value is 0.

The [Custom] section

`AppAdminCoordinates=`string

Defines a string that is referenced in error messages. This string should contain the name of the system administrator and the phone number of e-mail address (if required). The string may include any special characters and blanks.
The default value is an empty string.

Example

```
AppAdminCoordinates=Hans Müller, Tel: 0049/030/456 231
```

`PrimaryResPath=`string

Defines a directory, with the absolute pathname, from which resource files of type *.rbn* are loaded before starting the application.
The default value is an empty string.

`WindowTitle=`string

This string provides the title to be displayed in the first application window (not the primary window).
The default value is an empty string.

The [secondary window] section

`AutomaticClose={0|1}`

When data of a panel is sent to the host application, the subpanels usually remain open. This behavior can be modified with the value 1 in order to automatically close all subpanels on sending data.
The default value is 0.

`CenterOnFather={0|1}`

By default, subpanels are displayed centered on the parent panel. Setting this parameter to 0 causes WIN-DOORS to read the x and y values of the subpanel position from a resource file (*res* file) that was created on editing the panel. If a general protection fault occurs in the XTXM.DLL module when reading the resource file, you can edit the resource file with the Dialog Builder.
The default value is 1.

The [UsrDlls] section

`keys1=string`

Captures of pushbuttons and their associated actions are read from a dynamically linked library (DLL) that is assigned to forms with this input field. The specified library is used.

Starting with WIN-DOORS V3.0B, a library to support PF keys is included in the WIN-DOORS directory. This library is called *pfkeys.dll* and is used by the Natural programming language.
There is no default value.

`keys`_n=string_n
: You may specify any number of additional libraries for key conversions. This can be done by appending an integer in ascending order to the keyword “keys”.
There is no default value.

The [Help] section

`FileName=`string
: This string contains the name of the project file for Windows help.
The default value is an empty string.

identifier1=value1
: The identifier is that of the Help topic and can also be defined with the Help ID attribute in the DOORS editor. The value is the associated context number that was defined in the MAP section in the project file of the Windows help.

identifier_n=value_n
: You may specify any number of identifiers with their associated context numbers in this section.

The [SpecialKeysSyntax] section

`LeftPadded={0|1}`
: Defines whether the representation of keys is displayed by means of a two-digit integer. Leading blanks are replaced by zeros.
The default value is 1.

`Prefix=`string
: String with the prefix for PF keys.
The default value is P.

`Suffix=`string
: String with the suffix for PF keys.
The default value is an empty string.

`EMPrefix=`string
: String with the prefix for the EM key.
The default value is EM.

`EMSuffix=`string
: String with the suffix for the EM key.
The default value is an empty string.

`LAPrefix=`string
: String with the prefix for the LA key.
  The default value is LA.

`LASuffix=`string
: String with the suffix for the LA key.
  The default value is an empty string.

`LVDPrefix=`string
: String with the prefix for the LVD key.
  The default value is LVD.

`LVDSuffix=`string
: String with the suffix for the LVD key.
  The default value is an empty string.

Example

```
[SpecialKeysSyntax]
Prefix=<P
Suffix=>
LeftPadded=1
```

With this setting, the P15 key is displayed as <P15>; the P3 key as <P03>.

The [popup] section

`UsePopupRecognition={0|1}`
: Causes WIN-DOORS to recognize pop-up windows that are sent along with forms by the host application. This behavior is typically useful for applications written in Natural (see also the section “Dialog boxes” in the FHS-DOORS manual). The individual pop-up windows are managed in independent panels. You should generally set this parameter only if you really need it, since the complex algorithm involved has a negative impact on performance.
  The default value is 0.

`HStart=`string
: This string contains the characters with which the horizontal frame of a pop-up window begins.
  The default value is a colon followed by a period **:.** .

`VStart=`string
: This string contains the characters with which the vertical frame of a pop-up window begins.
  The default value is a colon followed by a period **:.** .

`HMiddle=`string
: This string contains the characters with which the horizontal frame of a pop-up window is displayed.
The default value is a period `.` .

`VMiddle=`string
: This string contains the characters with which the vertical frame of a pop-up window is displayed
The default value is a period `.` .

`HEnd=`string
: This string contains the characters with which the horizontal frame of a pop-up window ends.
The default value is a colon `:` .

`VEnd=`string
: This string contains the characters with which the vertical frame of a pop-up window ends.
The default value is a colon `:` .

`HOffset1={0|1}`
: Defines the number of spaces between the left margin and data in the pop-up window.
The default value is 0.

`HOffset2={0|1}`
: Defines the number of spaces between the right margin and data in the pop-up window.
The default value is 0.

`VOffset1={0|1}`
: Defines the number of spaces between the top margin and data in the pop-up window.
The default value is 0.

`VOffset2={0|1}`
: Defines the number of spaces between the bottom margin and data in the pop-up window.
The default value is 0.

Example

```
[connection]
BufferAutoMngt=0
PromptBeforeDelete=0
BufferSize=0
EmulationName=C:\DOORS_EM\DOORS_EM.EXE
EmulationCmdLine=C:\DOORS_EM\D016ZE07.DRE
```

```
ConnectionID=
DoorsTerminalType=
ResourcePath=c:\fhsdoors\tutorial

[misc]
ShowMainMenu=1
Optimized810Protocol=1
ShowEmptyProtectedBoxes=0
RemoveTrailingSpaces=32
MaxCacheEntry=10
ProcessLineMode=1
AutoResize=1
3DLook=1
HookDllPath=

[Custom]
LookForWildLSPs=0

[font]
FaceName=FHS DOORS
Style=Regular
Size=-13
Clip=2
Out=1
Quality=1
Family=49

[debug]
Diagnosis=0
[display attributes]
Normal=0
Underlined=4
Bold=8
Flashing=17

[alternate colors]
Background=12632256
Foreground=255

[secondary windows]
AutomaticClose=0
CenterOnFather=1
```

3.2 Starting and terminating WIN-DOORS

Start WIN-DOORS by double-clicking on the *WIN-DOORS* icon in the Windows 3.11 Program Manager or via the Start menu in Windows 95.

i You can also start a session directly. For details, refer to “Starting a session directly” on page 28.

After WIN-DOORS has started, the WIN-DOORS primary window appears.

From the primary window (main window), choose *Session/New*, select the resource path for the panels of the host application and specify the command used to call the emulation in the *Emulation parameters* fields. Choose *Session/Open* to open a session you saved previously with *Session/Save*.

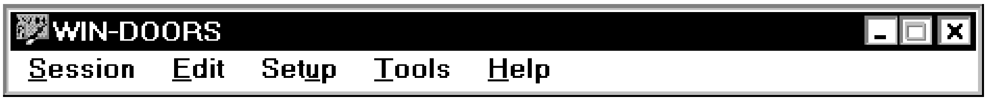


Figure 2: WIN-DOORS primary window

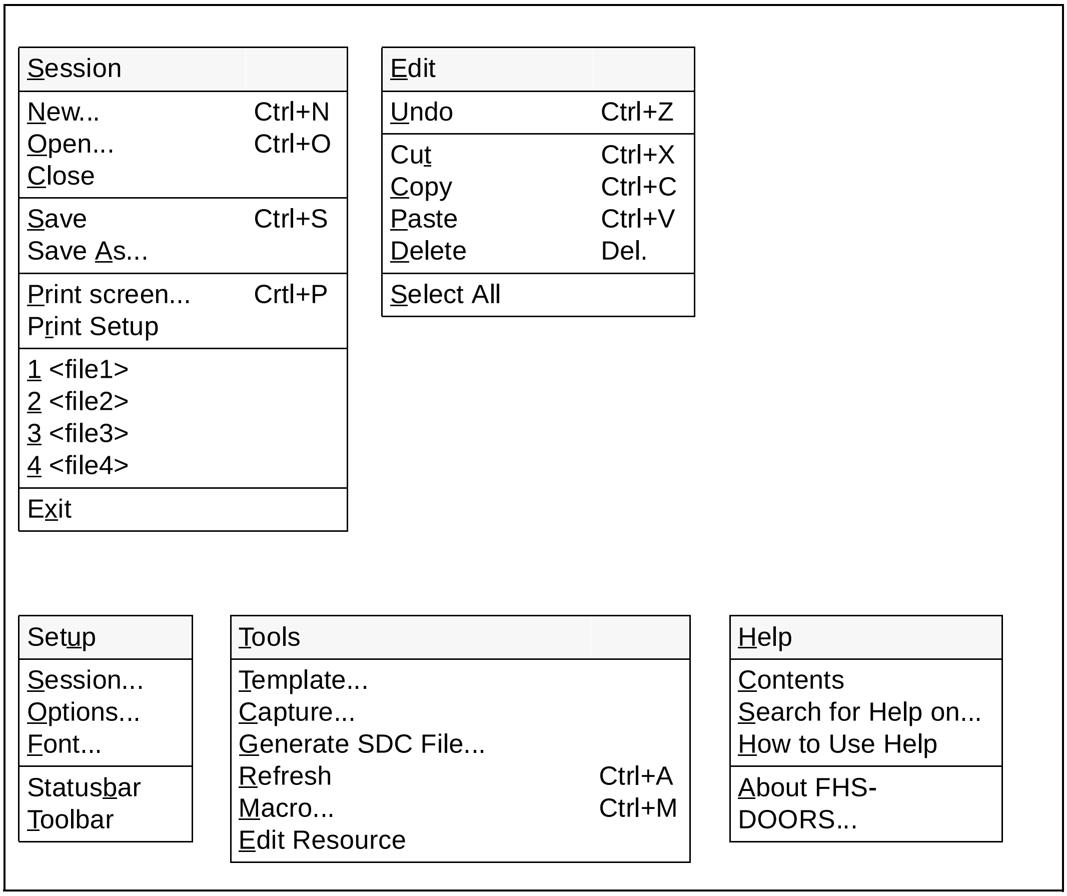


Figure 3: Menus in WIN-DOORS

Terminating WIN-DOORS

To terminate WIN-DOORS, choose *Session/Exit*.

Current connections are cleared as follows:

– by closing the current session in WIN-DOORS

– by terminating the host application on the host computer, e.g. by entering KDCOFF for UTM applications or LOGOFF for user programs.

If a session was started directly (see “Starting a session directly” on page 28), WIN-DOORS is also terminated when the session is closed.

Starting a session directly

WIN-DOORS offers an the option to start a session directly, i.e. to start WIN-DOORS, open the session, and establish a connection to the host computer.

windoors␣parameter-file

The parameter file must exist (see also section “Parameter files for sessions” on page 14).

When a session that was started directly is closed, WIN-DOORS is also terminated automatically.

If desired, you can create a separate icon for each session in the Windows 3.x Program Manager or the Windows 95 Start Menu and then open that session directly by simply double-clicking on the icon.

Windows 3.x

Copy the WIN-DOORS icon in the Program Manager (using the *File/Copy* command of the Program Manager) and change the program properties (using the *File/Properties* command) as follows:

Description	Enter the text that is to be appear under the icon
Command Line	windoors-installation-directory**\windoors**␣parameter-file

More information on the Program Manager can be found in the Microsoft Windows documentation.

Windows 95

Create a shortcut to the WIN-DOORS program by using the Windows 95 Explorer (with *Edit/Copy* or *Edit/Create Shortcut*). Then change the program properties in the *Properties* dialog box (with the *File/Properties* command) by entering the following via the *Shortcut* tab:

Target	windoors-installation-directory**\windoors**␣parameter-file

More information on the Explorer can be found in the Microsoft Windows 95 documentation.

3.3 Opening sessions and establishing connections

There are two ways of opening a session and establishing a connection with WIN-DOORS:

1. Open a new session (see the *Session/New* command in the WIN-DOORS online help system)
2. Open a saved session (see the *Session/Open* command in the WIN-DOORS online help system)

i For establishing connections, WIN-DOORS supports all connection methods/ transport access systems that are also supported by the emulation used. The DOORS emulation (supplied with WIN-DOORS) supports the connection methods LAN/CMX, BAM and HDLC/AFP.

3.3.1 Opening a new session

The *Session/New* command allows you to specify the resource path, the emulation parameters and mask buffer management for the new session.

In Emulation Parameters, specify the command for calling the emulation to be used to establish the connection. The command used to call the emulation depends on the emulation you are using. If the connection is to be established via the DOORS emulation, you must enter the following under *File Name*:

emul-installation-directory**\doors_em.exe**␣parameter-file

The parameter file must exist (see also section “Creating a parameter file for the emulation” on page 77)

or

emul-installation-directory**\doors_em.exe**␣**-C**␣connection-name

The connection name must exist.

Opening a session and establishing a connection

Click on *Start* to open the session and establish the connection.

Depending on the way in which the host application you wish to use was generated, you may first have to log on. The logon procedure with WIN-DOORS is the same as when WIN-DOORS is not used. Startup scripts allow you to automate this logon procedure if you are using the DOORS emulation for establishing the connection (see the manual “DOORS Emulation, 9750 Basic Emulation for BS2000/OSD Connections” [3]).

Once the session has been opened and the connection established, you can conduct a dialog with the host application in the WIN-DOORS application window.

Parts of the dialog with the application which are not handled with forms (e.g. KDCSIGN with UTM applications or SET-LOGON-PARAMETERS and START-PROGRAM for user programs) are handled by the emulation in the emulation window.

Opening a session without establishing a connection

Click *OK* to open the session without establishing a connection.

3.3.2 Editing the current session

The *Setup/Session* command enables you to view and change the resource path, emulation parameters and mask buffer management settings for the current session.

You can therefore select different panels in the current dialog by selecting a different resource path. This is useful, for example, when optimizing panels, as you can then display and test the newly optimized panels immediately.

If you want to change the emulation parameters or the mask buffer management, you must first clear down the connection (by choosing *Setup/Session* and clicking on *Stop*). Now you can, for example, select a different emulation or change the maximum buffer size. By clicking on *Start*, you can establish the connection with the new parameters.

Settings for the current session made with *Setup/Options* (Font, Display Attributes, Auto Tab, Overwrite, etc.) can be changed at any time. For example, to switch the status bar display on or off, choose *Setup/Status Bar*.

3.3.3 Saving the current session

The *Session/Save* or *Session/Save As*... command allows you to store the parameters of the current session in a parameter file so that you can open this session again later.

3.4 Designing user interfaces with subpanels

The output that is displayed by BS2000 application to the user is based on forms. A form is a contiguous line area with a specific field structure. It is not bound to a line position within a window. A specific form may appear at various positions in different windows.

Panels are developed with WIN-DOORS on the basis of underlying forms, i.e. WIN-DOORS composes the outputs in the window from the panels of the recognized forms. The representation of a field structure that appears in multiple forms can now be defined just once as a subpanel and need not be defined individually for each panel.

3.4.1 Recognizing forms and displaying the interface

WIN-DOORS recognizes forms on the basis of output attributes, which are maintained in a special database called the Capture database. Each capture in this database links an output attribute with one or more form assignments, and each form assignment assigns a form to a specific line position in the window. The records are arranged in order.

Whenever the BS2000 application modifies the output buffer, WIN-DOORS goes through the captures in the defined order and checks whether the respective attribute applies to the current form. If a match is found, the form assignments of the record are noted for the panel setup, provided that they do not coincide with the form assignments noted from an earlier match.

WIN-DOORS composes the final output from the (sub)panels of the noted forms in the defined order. If there are output areas for which no form assignment can be found on searching the database, the subpanels for these areas are generated dynamically.

The following rules apply to the assignment of subpanels:

– A specific form may be referenced in multiple captures.

– A specific form may be referenced in single capture more than once with different line positions.

– A specific output attribute can be used in multiple captures.

– The empty attribute is a special type of output attribute that always results in a match in every form. It can also be used in multiple captures.

3.4.2 Capture database

The Capture database can be processed with the *Tools/Capture* command. Every capture has a unique name (called the Capture name). The names of the capture used by WIN-DOORS for attribute recognition are listed in the corresponding order in the *Capture* dialog box. This order can be changed by using the two arrow buttons. For more information on how to create a capture, see section “Creating a capture” on page 43.

Two status flags on the left of the Capture name indicate the status of the capture in relation to the representation of the current output. The M flag (for Match) indicates a match; the A flag (for Active) means that the from assignment of the capture are used for the output. The status flags are followed by the number of resources defined in the form assignment.

The captures in the Capture database have the line position and the resource name for a form stored in them, but not the structure of the form. A description of the structure is stored together with the information for the graphical setup of the panel in the resource file. The creation of a new resource file is described in section “Generating a resource file” on page 44.

3.4.3 Tips on panel design

In order to complete the design of the graphical interface, the resource files are opened and processed individually in the editor. The following tips are useful in this context:

- The window menu graphical object: A window menu can be defined in every resource file. If the output is composed from the subpanels of multiple forms, the window menu for the output is built by combining the menu definitions of the individual panels.
- The alphanum graphical object: In order to have a correctly functioning panel, the field structure of the current output must usually match the corresponding structure stored in the resource. Deviations are possible only within very narrow limits, i.e. an I/O field in the resource may be matched by a pure output field in the current panel or several contiguous output fields in the resource may correspond to a single large output field in the panel.

  If an alphanumeric object is used in the panel, the situation changes, i.e. the field structure stored in the resource is not relevant for the form area of the alphanumeric object. This is because alphanumeric objects always work with the field structure that is current at runtime. A resource with an alphanumeric object thus defines the representation of an entire class of forms.

There are three typical cases in which alphanumeric objects are used:

- Screens with several (more than hundred) fields
  An alphanumeric object can be used to combine multiple fields into one graphical object. This reduces the number of graphical objects and thus the time required to set up the panel.
- Output information for the clipboard
  Under normal circumstances, only the information that can be overwritten in the screen of the BS2000 application can be marked in a panel and copied to the clipboard. In the case of an alphanumeric object, any text may be marked and copied to the clipboard, including the data in pure output fields.
- Dynamically structured screen areas
  Screen areas that frequently vary in structure, i.e. because the structure is defined by output data and only determined at runtime, can be easily incorporated in the graphical interface by means of an alphanumeric object.

The use of alphanumeric objects is a new feature in WIN-DOORS 3.1. One example of how such objects are used is the “built-in” LINEMODE resource for displaying screen outputs in line (Rollup) mode in the runtime system.

3.5 Online help for WIN-DOORS

The online help for WIN-DOORS contains a complete description of the WIN-DOORS interface.

There are a number of ways of calling the online help system:

- Choose *Help/Contents* to display a list of the topics in the online help system.
- Choose *Help/Search for Help on* to search for a particular topic or keyword.
- If you want to display help for a particular window or dialog box, press F1 or click on *Help*.
- Context-sensitive help for an object can be obtained by clicking on the icon for the “question-mark cursor” in the toolbar and then with the question-mark cursor on the object for which help is required. If no context-sensitive help for the selected object is available, help on the current window or on the current dialog box is displayed.
- The question-mark cursor cannot be clicked in a modal dialog box. If you want to request context-sensitive help for an object in this case, you will need to first select the corresponding object and then press the Ctrl+F1 key combination.

Choose *Help/How to Use Help* for more details on how to use the Microsoft Windows online help system.

4 Sample session

This chapter is intended to provide you with a quick introduction to WIN-DOORS. It is based on a single and progressive example that illustrates how you can graphically convert forms of BS2000 applications with WIN-DOORS and then edit them with the DOORS editor.

The sample session is supplied with WIN-DOORS and is stored in the *tutorial* directory during installation. When you run this session, you will be working without a connection to a host application, i.e. the behavior of the application is merely simulated. The emulation used when running the sample session is the DOORS emulation.

In order to correctly run this sample session, you will need to have a complete installation of WIN-DOORS, which includes the DOORS emulation and the example.

Escape routes in the case of errors

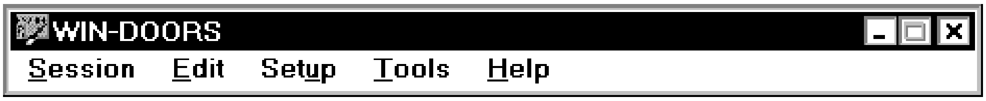
– If you have the feeling that you are stuck somewhere in the middle of the sample session, you can return to the previous state by pressing the *Cancel* button.

– The Alt + Tab key combination can be used to toggle between the individual (panel and emulation) windows of the sample session.

– A detailed description of the WIN-DOORS interface can be found by means of the Help function.

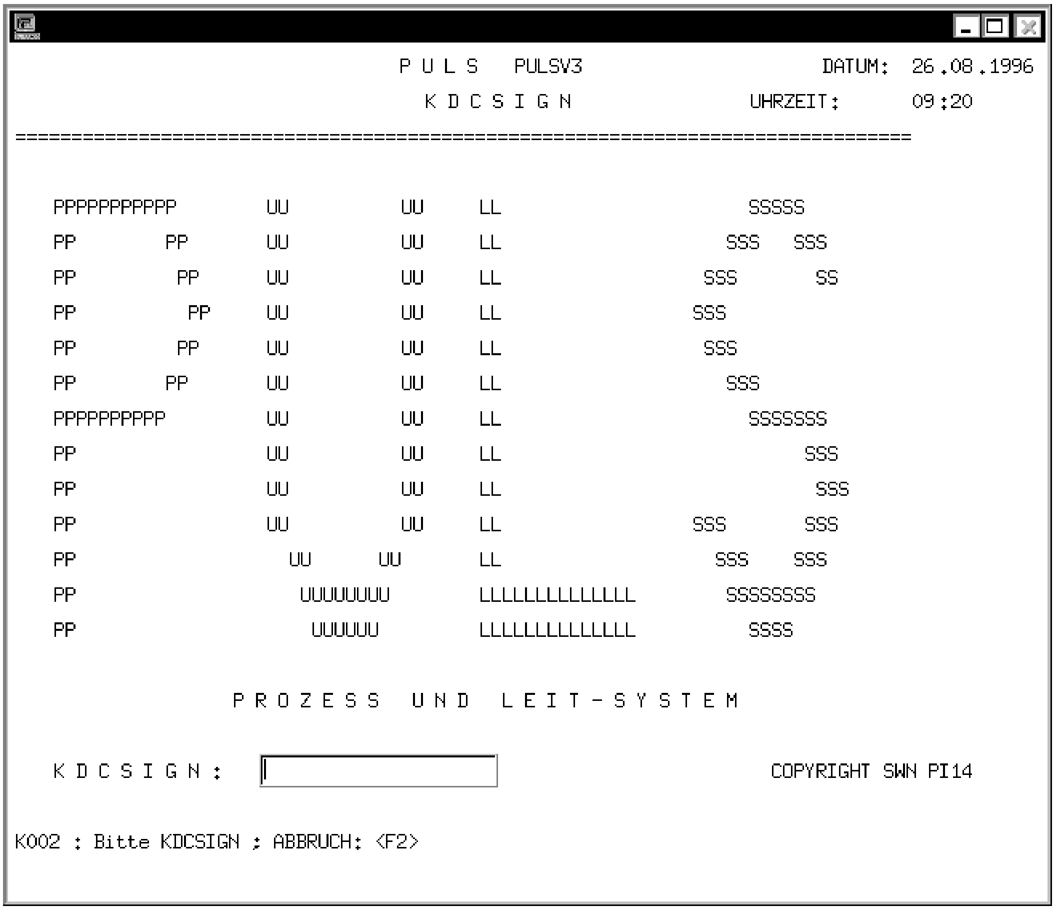
4.1 Starting the sample session

► Double-click the WIN program icon in the WIN-DOORS program group or select the program icon via the Start menu of Windows 95. The primary window of WIN-DOORS will appear.



► Select the *Session/Open* command. An *Open* dialog box will be displayed. Select the *Tutorial* subdirectory in the installation directory of WIN-DOORS and then the file *Tutorial.drs*.

► Click on *OK*. This will start the sample session.



```
                              P U L S   PULSV3                    DATUM:  26.08.1996
                               K D C S I G N                UHRZEIT:      09:20
================================================================================

   PPPPPPPPPPP        UU          UU     LL                      SSSSS
   PP        PP       UU          UU     LL                    SSS   SSS
   PP         PP      UU          UU     LL                  SSS       SS
   PP          PP     UU          UU     LL                 SSS
   PP         PP      UU          UU     LL                  SSS
   PP        PP       UU          UU     LL                    SSS
   PPPPPPPPPP         UU          UU     LL                      SSSSSSS
   PP                 UU          UU     LL                           SSS
   PP                 UU          UU     LL                            SSS
   PP                 UU          UU     LL                 SSS       SSS
   PP                   UU      UU       LL                   SSS   SSS
   PP                    UUUUUUUU        LLLLLLLLLLLLLL        SSSSSSSS
   PP                     UUUUUU         LLLLLLLLLLLLLL          SSSS

                    P R O Z E S S   U N D   L E I T - S Y S T E M

   K D C S I G N :   [                    ]                 COPYRIGHT SWN PI14

K002 : Bitte KDCSIGN : ABBRUCH: <F2>
```

The first form of the sample session is automatically converted to a panel and displayed on the screen. This form will not have any template assigned to it, as indicated by the entry *[none]* in the WIN-DOORS toolbar.

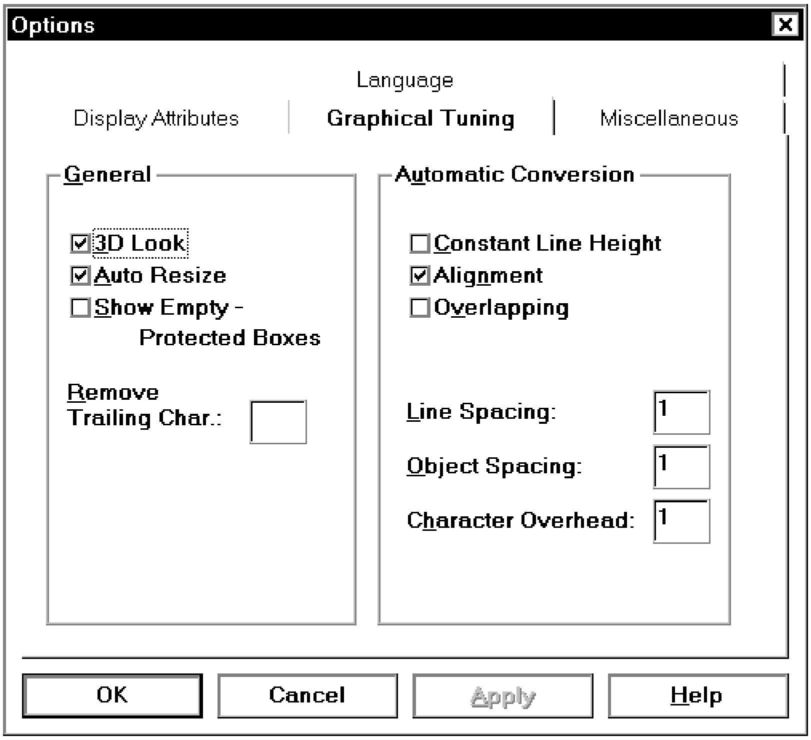
► Use the Alt + Tab key combination to bring the DOORS emulation window to the foreground. This key combination should be used throughout the sample session whenever you need to switch to individual windows of the application.

Configuring WIN-DOORS for the sample session

In order to obtain a better display of the panel, you should first set some options:

► This can be done by selecting the *Setup/Options* command. The *Options* dialog box will appear on the screen.

► Select the *Graphical Tuning* tab by clicking on it.



☑ 3D Look
☑ Auto Resize
☐ Show Empty - Protected Boxes
Remove Trailing Char.:

☐ Constant Line Height
☑ Alignment
☐ Overlapping

Line Spacing: 1
Object Spacing: 1
Character Overhead: 1

► Deactivate the *Constant Line Height* option in the *Automatic Conversion* group and confirm it with *OK*.

You can now customize the sample session for your work environment:

► Select the *Setup/Session* command. The *Edit Session* dialog box will be displayed.

► Verify that the directory containing the sample session has been set in the *Resource Path* field.

- Ensure that the program file of the DOORS emulation has been specified with the correct path in the *Name* field (*C:\DOORS_EM\DOORS_EM.EXE* for a default installation) and that the parameter file (*.dre* extension) for the emulation that is entered in the *Parameter* field is *TUTORIAL.DRE*.
- After you have verified that these settings are correct, click on *OK* to call the sample session.

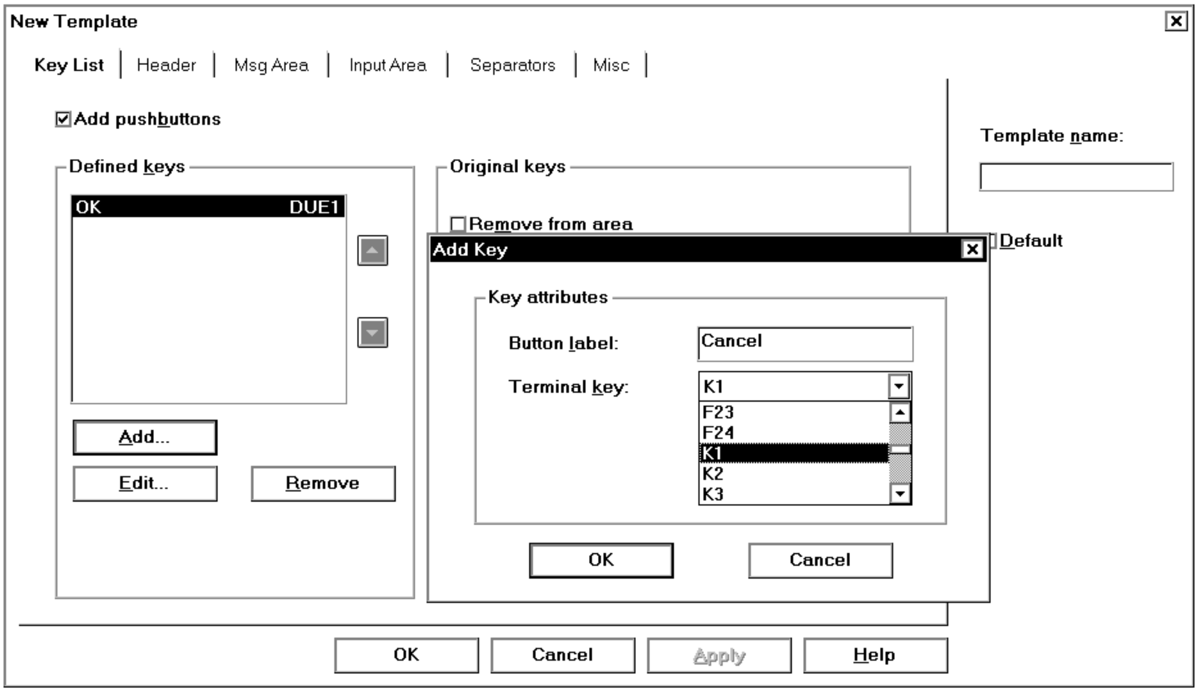
i The sample session will run as described if you accepted the default directories when installing WIN-DOORS.

4.2 Working with templates

Templates can be used to control how automatic conversion occurs. One example for the use of a template is the creation of *OK* and *Cancel* buttons in every panel.

4.2.1 Creating a template

- Select the *Tools/Template* command or click on the corresponding icon in the toolbar. The *Template* dialog box will appear. All templates already created would normally appear here in the *Template* list. At present, you will find only the entry *none*.
- To create a new template, click on the *New* button. The *New Template* dialog box appears. This dialog box comprises a number of tabs that can be selected to set various options for a template.
- In order to have buttons created during automatic conversion, you will need to click on the *Keys* tab to activate it.
- Select the *Add pushbuttons* option to activate the *Add* button.
- Click *Add*. The *Add keys* dialog box appears. This dialog box can be used to define the label for the button and the corresponding action.



☑ Add pushbuttons

Defined keys

OK DUE1

Original keys

☐ Remove from area

Template name:

☐ Default

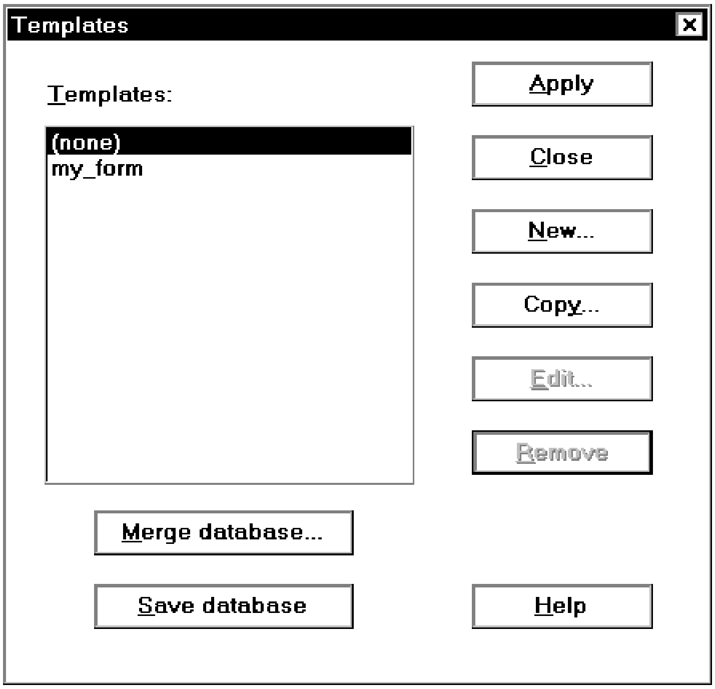
Key attributes

Button label: Cancel

Terminal key: K1

F23
F24
K1
K2
K3

- ► Enter **OK** in the *Button Label* entry field.
- ► Select the entry *DÜ1* from the *Terminal Key* list and confirm your selection with *OK*. The dialog box is closed, and your input appears in the *Defined Keys* list.
- ► Use the same method to define a **Cancel** button for the *K1* action.
- ► Now enter a name for your template in the *Template Name* field of the *New Template* dialog box, i.e. my_form. If you forget to specify a name for your template, you will receive an error message.
- ► Select the *Default* option so that WIN-DOORS assigns your template to all forms during automatic conversion.
- ► Confirm your settings with *OK* and do the same for the prompt that asks you whether you really want to change the default template. This closes the dialog box with the prompt. A new entry called *my_form* should now be visible in the *Templates* list in the *Template* dialog box.



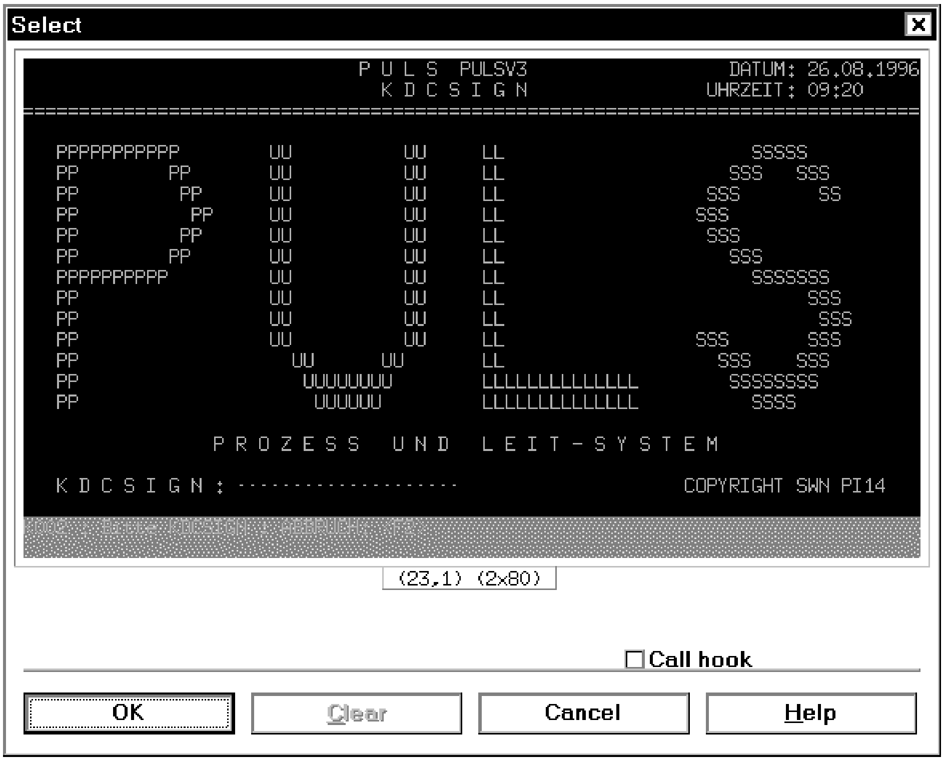
In order to have even the first panel displayed with this new template, you will now need to assign the template to it.

- This can be done by clicking on *Apply* in the *Template* dialog box.

4.2.2 Editing a template

The next step is to edit the template in order to define a message area and a separator for the forms of your application.

- In the *Template* dialog box, click on *Edit*. If you had already closed the dialog box, you will need to open it again with the *Tools/Template* command. The *Edit Template* dialog box appears on the screen.
- Select the *Msg Area* tab in the *Edit Template* dialog box.
- Now choose the *Create message box* option in the *Msg Area*. This activates the *Select* button.
- Click on *Select* to display the *Select* dialog box.



```
                              P U L S  PULSV3                DATUM: 26.08.1996
                                K D C S I G N               UHRZEIT: 09:20
================================================================================

   PPPPPPPPPPP        UU          UU      LL                        SSSSS
   PP        PP       UU          UU      LL                      SSS   SSS
   PP         PP      UU          UU      LL                    SSS       SS
   PP          PP     UU          UU      LL                   SSS
   PP         PP      UU          UU      LL                    SSS
   PP        PP       UU          UU      LL                      SSS
   PPPPPPPPPP         UU          UU      LL                        SSSSSSS
   PP                 UU          UU      LL                             SSS
   PP                 UU          UU      LL                              SSS
   PP                 UU          UU      LL                   SSS       SSS
   PP                   UU      UU        LL                     SSS    SSS
   PP                    UUUUUUUU         LLLLLLLLLLLLLL          SSSSSSSS
   PP                     UUUUUU          LLLLLLLLLLLLLL            SSSS

                P R O Z E S S   U N D   L E I T - S Y S T E M

   K D C S I G N : --------------------                  COPYRIGHT SWN PI14
K002 : Bitte KDCSIGN : ABBRUCH: <F2>

```

(23,1) (2x80)

- Use the left mouse button to mark the last two lines in the form where messages are usually output by the host application. If you have marked the lines correctly, the last two lines should be displayed in a different color (see Figure).
- Confirm the marked lines by clicking on *OK*.
- The line values for your marked lines are copied to the fields of the *Original Message* group. They should now have the following values:

Starting line	23
Height of area	2

  If you do not succeed at marking the lines for some reason, you can also enter the values directly into the entry fields.
- Deactivate the *Protected lines only* option.

This completes the definition of a message area in the form. Messages received from the host application will now be displayed in a separate message panel on the PC display terminal.

You should now replace the alphanumeric separators by graphical ones.

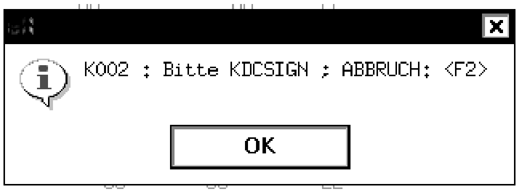
- ► To do this, select the *Separators* tab in the *Edit Template* dialog box.
- ► Select the *Replace by graphical separator* option in the *Horizontal Separator* group.
- ► In the *Alphanumeric Separator* input field, enter the characters **= _ *** without any delimiter between them. This defines the characters to be replaced by a graphical separator during automatic conversion.
- ► Enter the value **3** in the *Min. number of characters* entry field. This value indicates that at least 3 of the specified characters must be present at the start of a line in order for them to be replaced by a graphical separator.

In order to ensure that the editor can edit the representation of labels in the panel, the following setting must still be made:

- ► In the *Edit Template* dialog box, select the *Misc* tab.
- ► Select the *Split at blanks* option and accept the default value (2). Labels that are too long will now be wrapped during automatic conversion.
- ► Confirm your settings with *OK* in the *Edit Template* dialog box.

4.2.3 Assigning a template to a panel

- ► To make your changes effective for the current panel, you will need to click on *Apply* in the opened *Template* dialog box. You should now see the following message panel on the screen:


K002 : Bitte KDCSIGN ; ABBRUCH: <F2>

- ► Confirm the message window of the application with *OK*.
- ► Close the *Template* dialog box by clicking on *Close*.
- ► Enter your name in the *KDCSIGN* entry field in the first panel of the sample application and send this value to the host application by clicking on the new *OK* button.

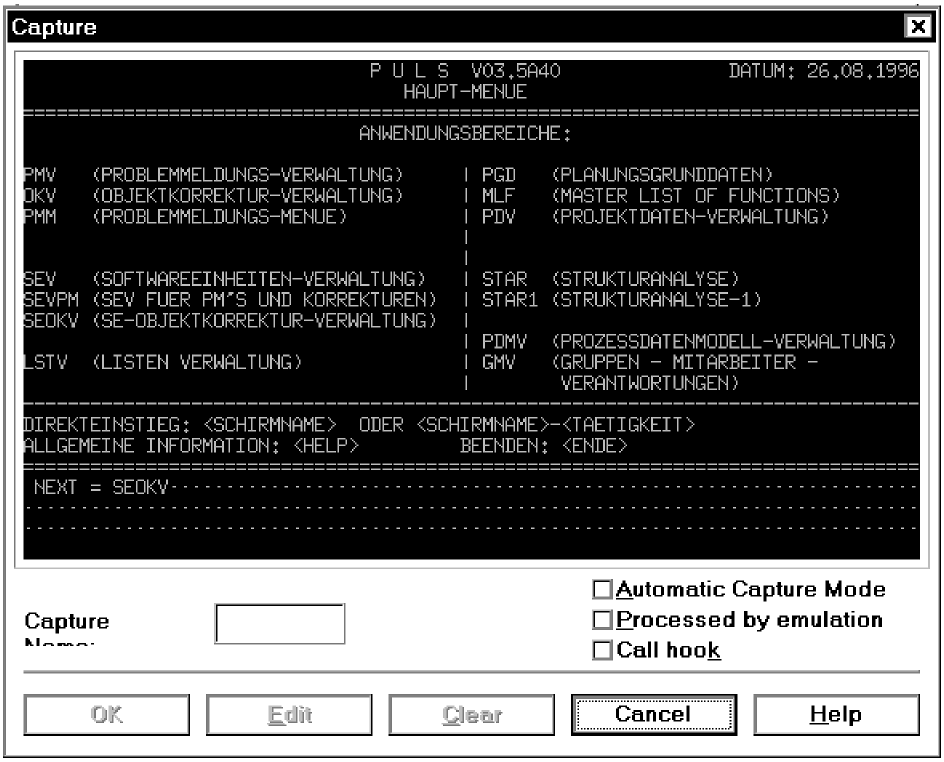
  The next form of the sample application is automatically converted in accordance with the template specifications and displayed as a panel on the screen.

4.3 Optimizing a panel with the DOORS editor

If you are not quite satisfied with the result of the automatic conversion, you can also customize the appearance of the displayed image by using the DOORS editor. In order to do this, the panel must be stored as a resource file. This resource file can then be edited with the editor.

4.3.1 Creating a capture

- Select the *Tools/Capture* command. The *Capture* and *Capture Management* dialog boxes appear on the screen. It contains the alphanumeric representation of the current panel.
- In order to create a capture for a form, you must first select a form identifier. Do this by marking the term *MAIN-MENU* in the displayed form with the left mouse button.

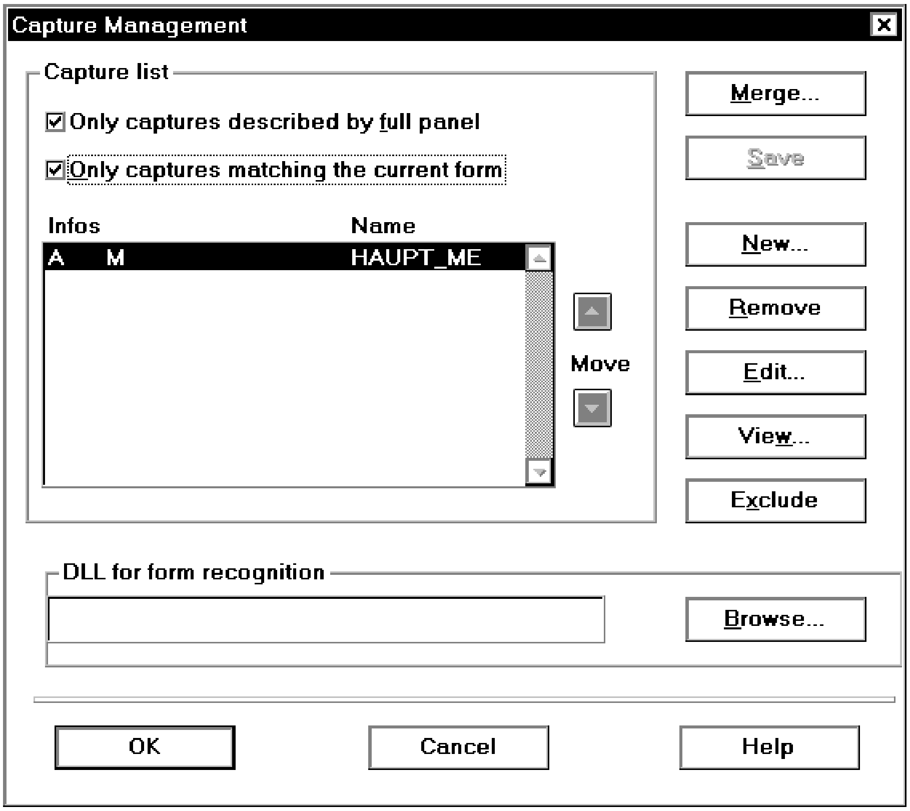


```
                              P U L S  V03.5A40            DATUM: 26.08.1996
                                 HAUPT-MENUE
================================================================================
                             ANWENDUNGSBEREICHE:

PMV   (PROBLEMMELDUNGS-VERWALTUNG)      | PGD   (PLANUNGSGRUNDDATEN)
OKV   (OBJEKTKORREKTUR-VERWALTUNG)      | MLF   (MASTER LIST OF FUNCTIONS)
PMM   (PROBLEMMELDUNGS-MENUE)           | PDV   (PROJEKTDATEN-VERWALTUNG)
                                        |
                                        |
SEV   (SOFTWAREEINHEITEN-VERWALTUNG)    | STAR  (STRUKTURANALYSE)
SEVPM (SEV FUER PM'S UND KORREKTUREN)   | STAR1 (STRUKTURANALYSE-1)
SEOKV (SE-OBJEKTKORREKTUR-VERWALTUNG)   |
                                        | PDMV  (PROZESSDATENMODELL-VERWALTUNG)
LSTV  (LISTEN VERWALTUNG)               | GMV   (GRUPPEN - MITARBEITER -
                                        |        VERANTWORTUNGEN)
--------------------------------------------------------------------------------
DIREKTEINSTIEG: <SCHIRMNAME>  ODER <SCHIRMNAME>-<TAETIGKEIT>
ALLGEMEINE INFORMATION: <HELP>         BEENDEN: <ENDE>
================================================================================
 NEXT = SEOKV-------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
```

The marked form identifier is copied into the *Capture Name* entry field.

► Confirm the name for the identifier with *OK*. This closes the *Capture* dialog box.

► You will find an overview of all existing captures in the *Capture Management* dialog box. The new capture will appear in the *Capture List*.


Capture Management

Capture list

☑ Only captures described by full panel

☑ Only captures matching the current form

Infos	Name
A M	HAUPT_ME

► Confirm the new capture with *OK*. This closes the *Capture Management* dialog box.

4.3.2 Generating a resource file

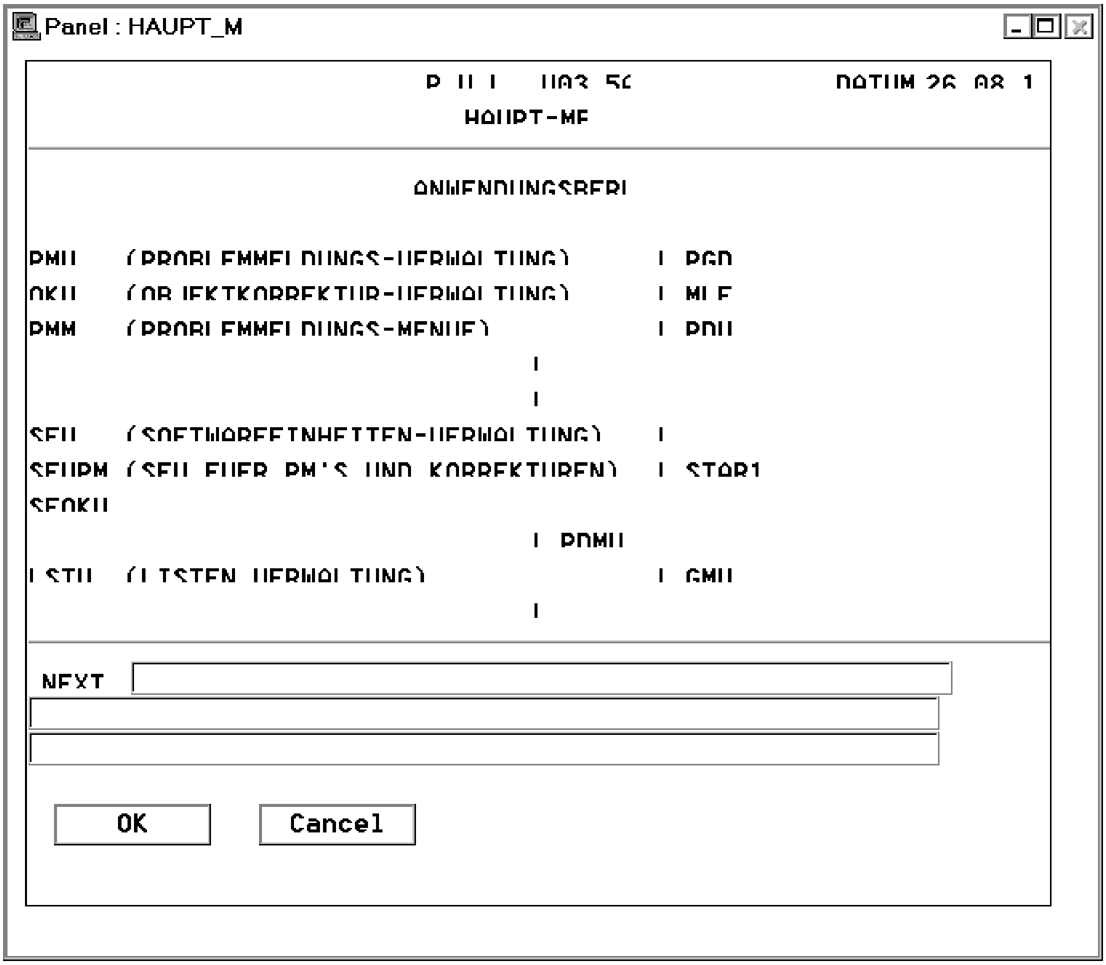
The capture created above can now be used to generate a resource file named MAIN-MEN.sdc.

► To do this, select the *Tools/Generate SDC File* command. The resource file is created and placed in the resource directory with no further message.

► Call up the DOORS editor with the *Tools/Edit Resource* command. This starts the editor with the current panel displayed in its primary window. The newly created resource file MAIN-MEN.sdc is used by the DOORS editor as the input file in which all changes are stored.

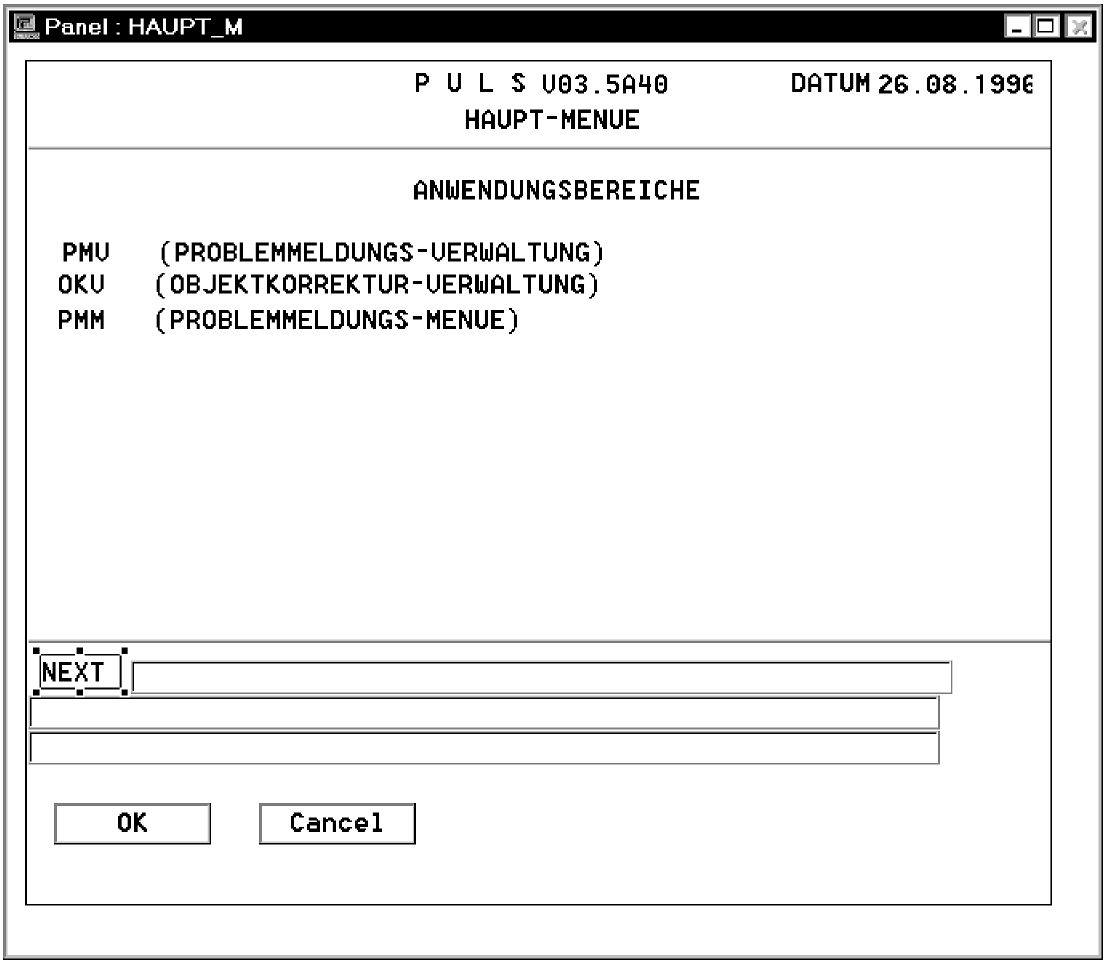
4.3.3 Editing a panel

In this example, the panel with the main menu is to be modified so that only the commands which are useful for an inexperienced user can be selected. To do this, you must first delete the unnecessary objects from the panel.



```
                                   P U L   VA3.5C              DATUM 26.08.1
                                      HAUPT-ME

                                   ANWENDUNGSBERI

PMU    (PROBLEMMELDUNGS-VERWALTUNG)       | PGD
OKU    (OBJEKTKORREKTUR-VERWALTUNG)       | MLE
PMM    (PROBLEMMELDUNGS-MENUE)            | PDU
                                   |
                                   |
SEU    (SOFTWAREEINHEITEN-VERWALTUNG)     |
SEUPM  (SEU FUER PM'S UND KORREKTUREN)    | STAR1
SEOKU
                                   | PDMU
LSTU   (LISTEN VERWALTUNG)                | GMU
                                   |

 NEXT
```

- ► To delete the objects, click on the selection tool in the toolbar and draw a frame around the objects that you wish to delete. The selected objects will be shown as marked.
  This can also be done by holding down the Shift key and clicking on the objects individually with the mouse.
- ► To delete the objects, press the DEL key or select the *Edit/Delete* command.
- ► Delete all redundant objects in the panel so that only the following objects are visible:



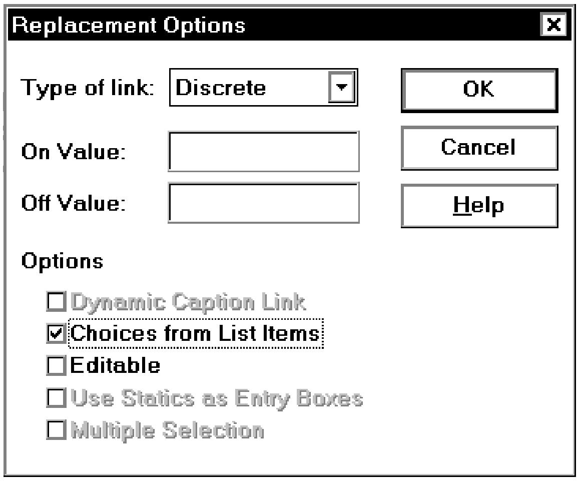
```
Panel : HAUPT_M

                    P U L S U03.5A40      DATUM 26.08.1996
                      HAUPT-MENUE

                    ANWENDUNGSBEREICHE

 PMU   (PROBLEMMELDUNGS-UERWALTUNG)
 OKU   (OBJEKTKORREKTUR-UERWALTUNG)
 PMM   (PROBLEMMELDUNGS-MENUE)

NEXT

   OK          Cancel
```

Note that some text fields may also include some hidden text. In this example, all help information is supplied with a variable value by the BS2000 application. These values are shown in angle brackets.

- Use the selection tool to mark the six text fields and move them with the mouse to the upper middle area of the panel. If you do not succeed at moving them on your first attempt, you have probably selected the panel as well (as in the diagram). Deselect the panel by pressing the CTRL key and clicking on the panel.
- Retain the marked text fields and select the *Graphical Tuning* tab in the properties window.
- Double-click on *Font* in the *Graphical Tuning* tab. This opens the *Font* dialog box.
- Select some other *Font* and *Size* for the text fields, e.g. the Arial font with a 10 point size.
- Confirm your selection with *OK*. The marked text fields will be displayed in the selected font. If the selected font is too wide or the font size is too large, the text frame may be too small, and the displayed text may be truncated. If this occurs, you will need to click on the text frames individually and extend them as required to make the entire text visible.
- Now arrange the fields of the first column so that they end at the same height. This can be done by marking the fields and selecting the *Layout/Align Objects>Right* command. All objects will be aligned on the last selected object.
- Arrange the fields of the second column so that they all begin at the same height as well. This can be done by marking the fields and selecting the *Layout/Align Objects>Left* command. All objects are aligned on the last selected object.

So far, you have only processed the visual aspects of the panel in the sample session. You can, however, also use the DOORS editor to create graphical objects in the panel and associate them with actions. Existing objects can be converted to graphical objects by using the “Replace” tool. In the next step, the multi-line entry field *NEXT* at the end of the panel is to be replaced by a list.

- To do this, mark the first line of the multi-line entry field.
- Press the right-hand mouse button to display the context menu.
- Select the *Replace By>Combo Box* command. This opens the *Replacement Options* dialog box.



Replacement Options

Type of link: Discrete

On Value:

Off Value:

Options

- [ ] Dynamic Caption Link
- [✓] Choices from List Items
- [ ] Editable
- [ ] Use Statics as Entry Boxes
- [ ] Multiple Selection

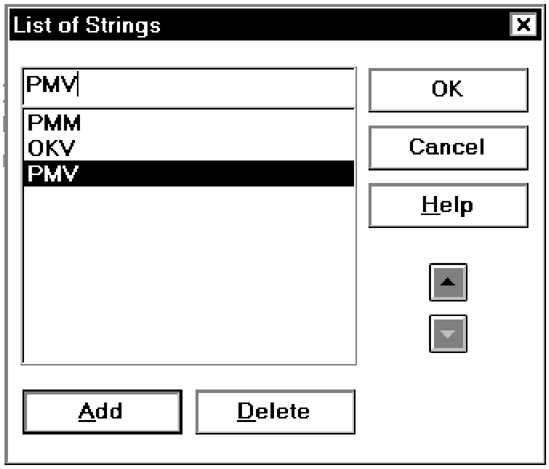
► Select the *Choices from List Items* option and confirm it with *OK*. The dialog box is closed, and the entry field is replaced by an empty list.

► Change the size of the frame for the choicelist by extending it downward to define the size of the list in a drop-down state.

► Check whether the list is still marked or mark it if required and click the *Graphical Tuning* tab in the properties window.

► In the *Graphical Tuning* tab, double-click on the *List Items* property. The *List of Strings* dialog box appears.

► Enter the names of the individual elements in the entry fields of the *List of Strings* dialog box and confirm each entry with *Add*.

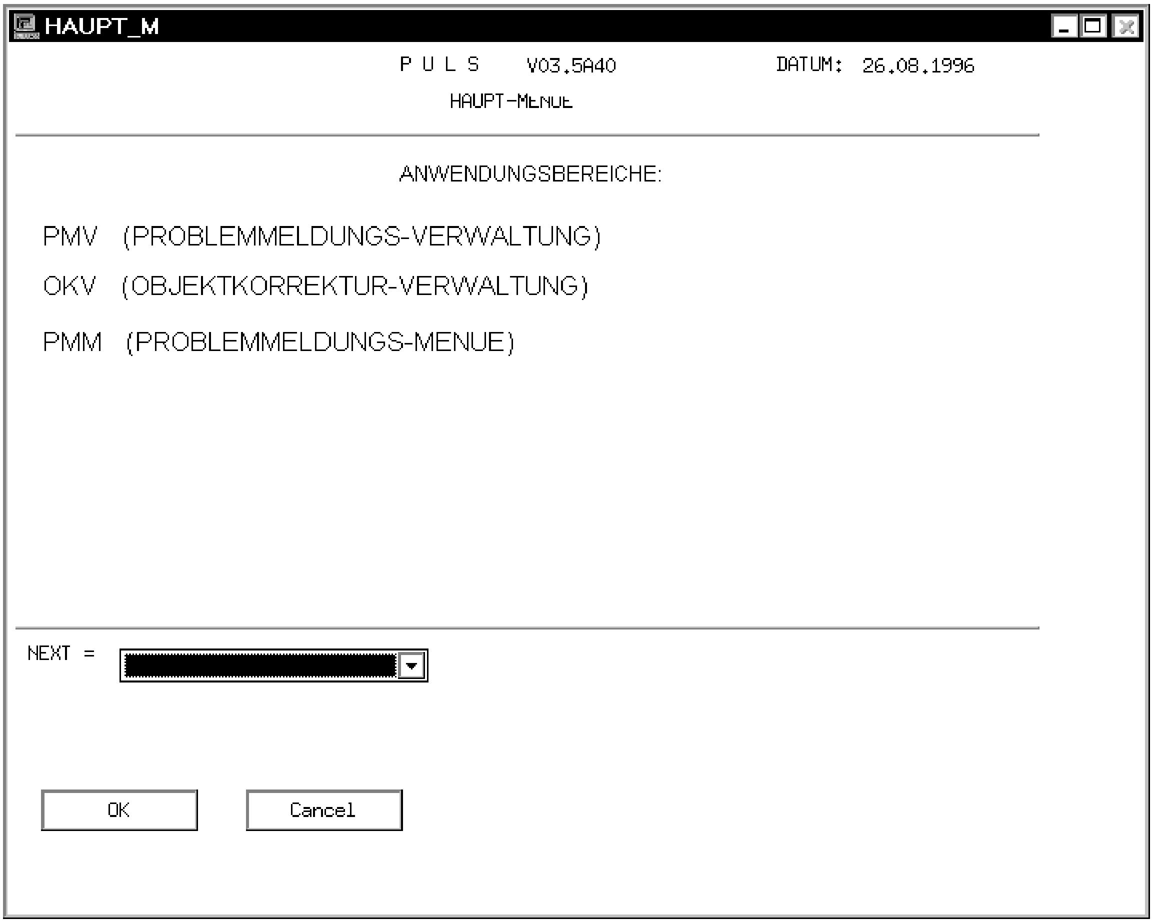


- Close the dialog box with *OK*. You have now finished defining list items, but these will not yet appear in the list. The items will be presented to the user as a choicelist only at runtime.
- Save the edited panel in the resource file using the *File/Save* command.
- Exit the DOORS editor with *File/Exit*. WIN-DOORS is still active with the sample application and will now become visible on the screen.

4.4 Testing an edited panel

The panel displayed on the screen will still reflect its state prior to editing.

► To display the edited panel, select the *Tools/Refresh* command. WIN-DOORS will directly read the resource file and display the edited panel.



HAUPT_M

P U L S V03.5A40 DATUM: 26.08.1996

HAUPT-MENUE

ANWENDUNGSBEREICHE:

PMV (PROBLEMMELDUNGS-VERWALTUNG)

OKV (OBJEKTKORREKTUR-VERWALTUNG)

PMM (PROBLEMMELDUNGS-MENUE)

NEXT =

The entries in the new list are now also shown.

► Make sure that only the entered items in the list are selectable.

► Now click on *OK* in the panel to display the third panel of the sample application.

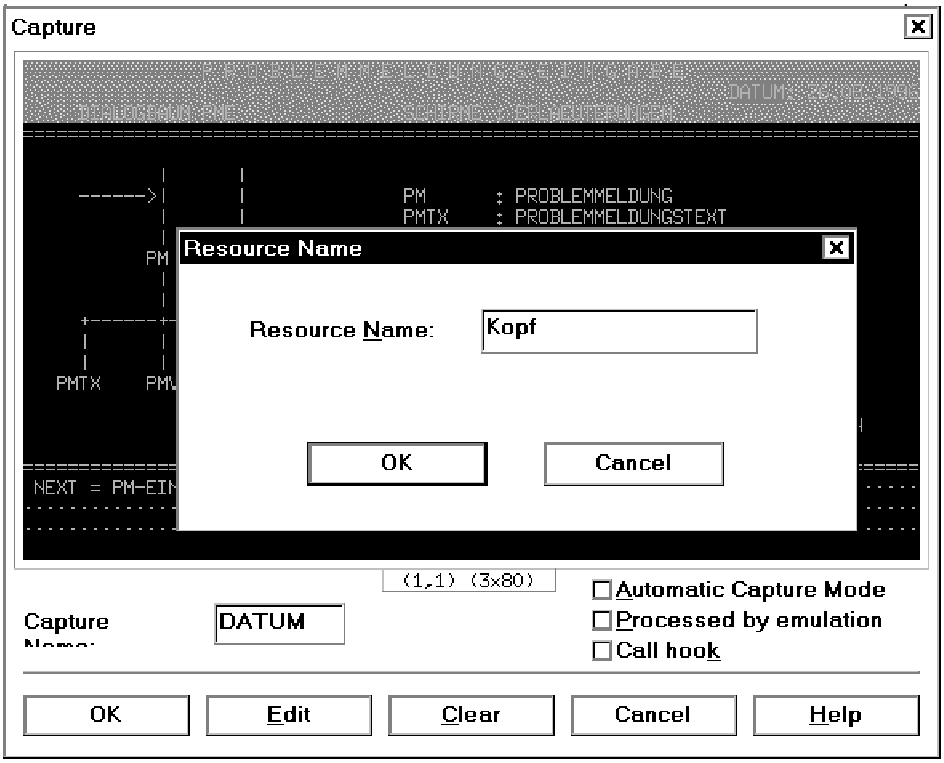
4.5 Creating and editing subpanels

Subpanels enable you to manage the standard and “dynamic” areas of a form separately. Many forms have identical headers and footers with the same design and content and differ only in terms of their work areas. Consequently, you could define a subpanel for the header area and another subpanel for the footer area of such forms so that the underlying templates are displayed on the screen whenever a match is detected during automatic conversion. This also reduces the optimization overhead when converting an application.

► Select the *Tools/Capture* command. The *Capture* and *Capture Management* dialog boxes are displayed on the screen with the alphanumeric representation of the current panel.

4.5.1 Capturing an area for a header panel

► In order to create a capture for a form, you must first select a form identifier. You can do this in the displayed form by using the left mouse button to mark the term DATE, for example. The marked term is again copied into the *Capture Name* field.



```
        P R O B L E M M E L D U N G S E I N G A B E
                                                   DATUM: 26.08.1996
   DIALOGBAUM PNE              SCHIRME : ERLAEUTERUNGEN
================================================================
             |       |
     ------->|       |          PM      : PROBLEMMELDUNG
             |       |          PMTX    : PROBLEMMELDUNGSTEXT
             |
            PM
             |
             |
     +-------+-
     |       |
     |       |
   PMTX     PMV
NEXT = PM-EIN
```

Resource Name

Resource Name: Kopf

(1,1) (3x80)

Capture Name: DATUM

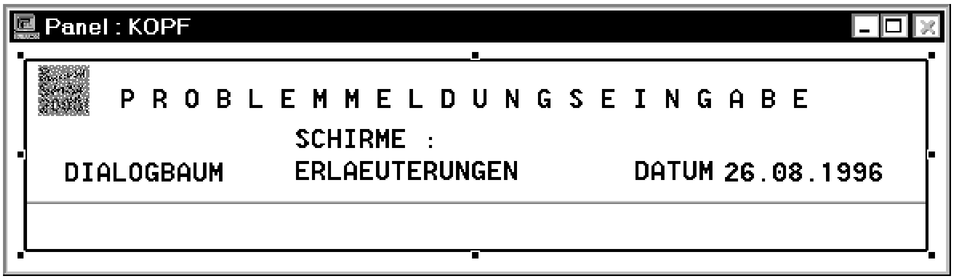
☐ Automatic Capture Mode
☐ Processed by emulation
☐ Call hook

- Now use the right mouse button to mark the first three lines of the form. The right mouse button selects the lines in the *Capture* dialog box that are to be included in the subpanel. The marked lines are shown in a different color, and the *Form Name* dialog box appears.
- Enter a name for the subpanel, e.g. header, and confirm it with *OK*.
- Confirm the *Capture* dialog box with *OK*. The subpanel will now appear in the *Capture List* of the *Capture Management* dialog box. The number 1 in the third column of the list indicates that DATE is a subpanel.
- Confirm the *Capture Management* dialog box with *OK*.
- Use the *Tools/Generate SDC File* command to create a resource file.
- Call the DOORS editor with the *Tools/Edit Resource File* command. This starts the DOORS editor with the current subpanel opened as the input file.

4.5.2 Editing a header panel

In this step, you will insert an image (i.e. a bitmap file) as a small graphical logo in the header panel.

- Click on the icon for a bitmap in the toolbar and hold the left mouse button pressed while moving the cursor to the upper left corner of the subpanel on the screen. When you now release the mouse button, a marked empty frame will be inserted at the position of the cursor.
- Open the properties window for the bitmap by using the *View/Properties* command. Note that the bitmap must be marked for this purpose.
- Select the *Graphical Tuning* tab in the properties window.
- Double-click on the *Bitmap Name* property and enter the name of the bitmap to be placed in the frame, i.e. **bstrateg.bmp**, in the list box.
- Confirm the file name with *OK*. The bitmap will be placed in the frame and can then be moved to any position or sized as required.



```
P R O B L E M M E L D U N G S E I N G A B E
                    SCHIRME :
DIALOGBAUM          ERLAEUTERUNGEN        DATUM 26.08.1996
```

- Exit the DOORS editor with the *File/Exit* command and answer the prompt to save the resource file with *YES*.

4.5.3 Capturing an area for a footer panel

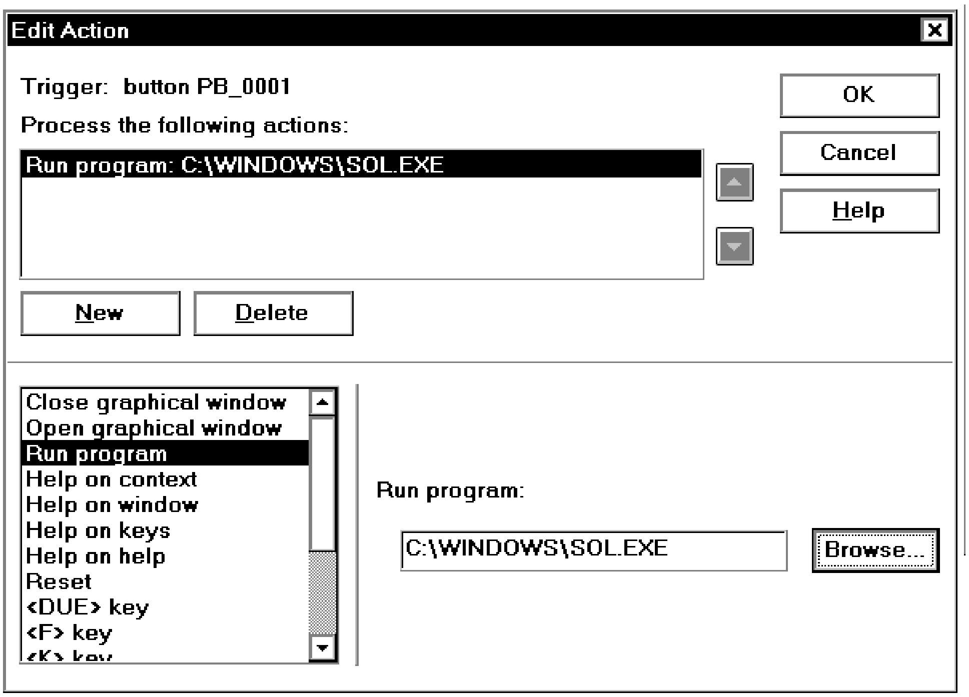
- Select the *Tools/Capture* command. The *Capture Management* dialog box appears with the alphanumeric representation of the third panel of the sample application still displayed in it.
- Select *New*.
- In order to create a capture for a form, you must first select a form identifier. You can do this in the displayed form by using the left mouse button to mark the term NEXT. The marked term is again copied into the *Capture Name* field.
- Now use the right mouse button to mark the last three lines of the form. The marked lines are shown in a different color, and the *Form Name* dialog box appears.
- Enter a name for the subpanel, e.g. footer, and confirm it with *OK*.
- Confirm the *Capture* dialog box with *OK*. The subpanel will now appear in the *Capture List* of the *Capture Management* dialog box. The number 1 in the third column of the list indicates that NEXT is a subpanel.
- Confirm the *Capture Management* dialog box with *OK*.
- Use the *Tools/Generate SDC File* command to create a resource file.
- Call the DOORS editor with the *Tools/Edit Resource File* command. This starts the DOORS editor with the current subpanel opened as the input file.

4.5.4 Editing a footer panel

In this step, you will add a button and an additional text field to the footer panel.

- Click on the icon for a button in the toolbar and hold the left mouse button pressed while moving the cursor to a position next to the existing buttons. When you now release the mouse button, a new button will be inserted at the position of the cursor. This button can then be moved or sized as required.
- Open the properties window for the bitmap by using the *View/Properties* command. Note that the button must be marked for this purpose.
- Select the *Graphical Tuning* tab in the properties window.
- Double-click on the *Title* property and enter a label for the button in the entry field of the property window, e.g. **Solitaire**.

- Confirm the button label by pressing the ENTER key.
- In the property window, select the *Main* tab.
- Double-click on *Action*. The *Edit Action* dialog box appears.



Edit Action

Trigger: button PB_0001

Process the following actions:

Run program: C:\WINDOWS\SOL.EXE

Close graphical window
Open graphical window
Run program
Help on context
Help on window
Help on keys
Help on help
Reset
<DUE> key
<F> key
<K> key

Run program:

C:\WINDOWS\SOL.EXE

- Click on *New* to link one of the actions from the list with the new button. This activates the list so that you can select an entry.
- Select the *Run Program* entry.
- Now click on the *Browse* button to search in the Windows list box for the program file that is to be executed on clicking the button.
- Go to the C:\WINDOWS directory and select the program *SOL.EXE*.
- Close the list box with *OK*. The name of the program file is copied to the *Start Program* entry field along with the path.
- Close the *Edit Action* dialog box with *OK*.

You have now completed the process of adding a new button and associating it with a program to be executed whenever the user clicks on that button at runtime. The next step is to add a text field, but you will first need to create some space for it.

► Delete the text field *NEXT* and the second and third line of the entry field.

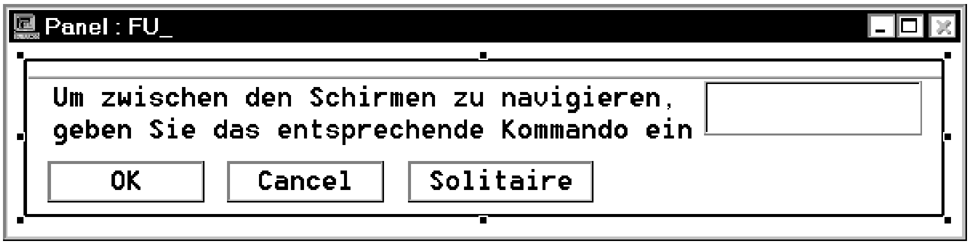
► Then add a new text field by using the icon for a text field and following the usual procedure.

► Select the *Graphical Tuning* tab in the properties window for the text field. Note that the text field must be marked for this purpose.

► Double-click on the *Title* property and enter the following or any similar text: **To navigate between screens, enter the appropriate command**.

► Confirm the new label for the text field with the Enter key. The text will appear in the text field.

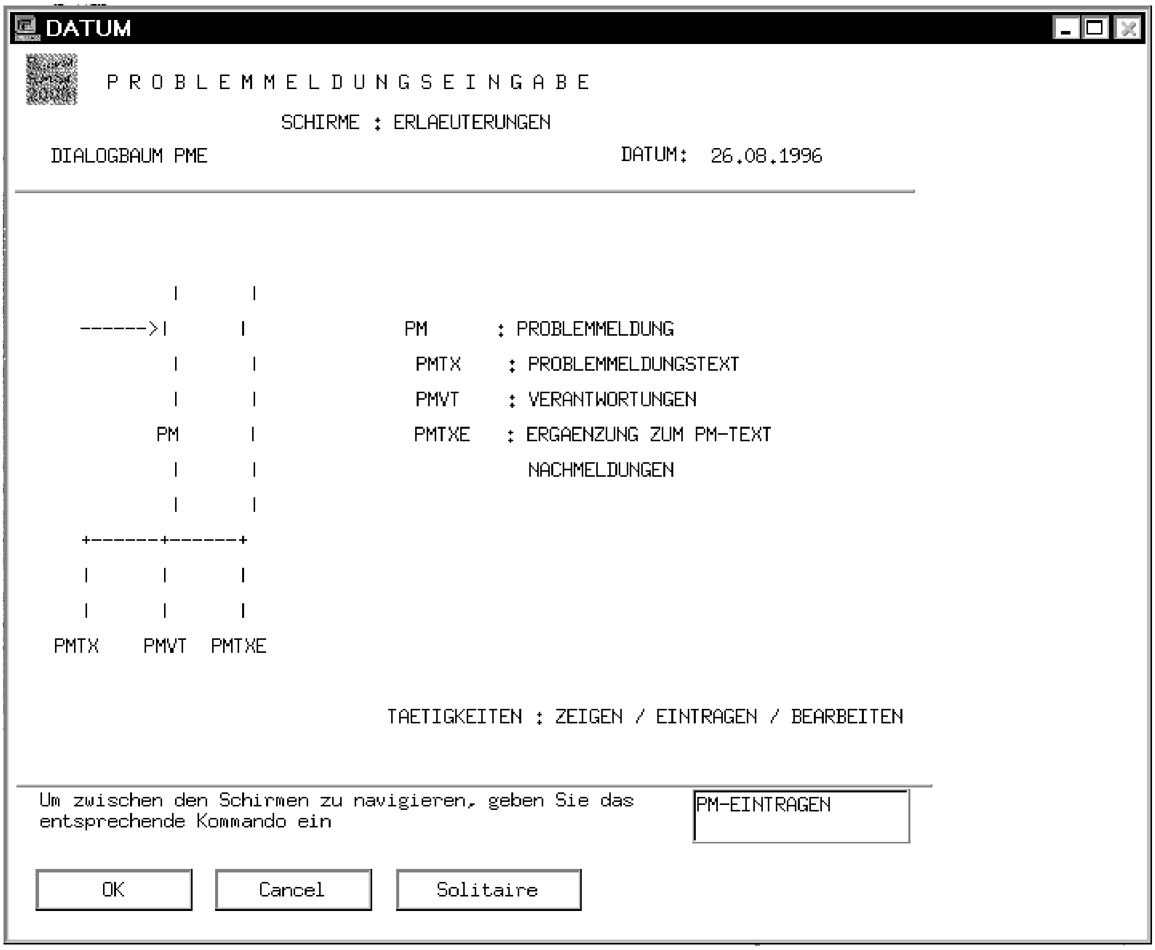
► Adjust the size of the text field to suit the entered text. If desired, you can also change the font or other display attributes of the text field such as font and background colors.


Panel : FU_

Um zwischen den Schirmen zu navigieren,
geben Sie das entsprechende Kommando ein

► Exit the DOORS editor with the *File/Exit* command and answer the prompt to save the resource file with *YES*. The editor will be closed, and you will see the panel and the primary window of WIN-DOORS again.

► Select the *Tools/Refresh* command to have the optimized subpanel displayed. Depending on your entries, a screen similar to the one below will appear:



```
P R O B L E M M E L D U N G S E I N G A B E
                    SCHIRME : ERLAEUTERUNGEN
DIALOGBAUM PME                                   DATUM:  26.08.1996

          |    |
  ------>|    |           PM      : PROBLEMMELDUNG
          |    |            PMTX    : PROBLEMMELDUNGSTEXT
          |    |            PMVT    : VERANTWORTUNGEN
         PM    |            PMTXE   : ERGAENZUNG ZUM PM-TEXT
          |    |                      NACHMELDUNGEN
          |    |
  +------+------+
  |      |      |
  |      |      |
PMTX   PMVT   PMTXE

                    TAETIGKEITEN : ZEIGEN / EINTRAGEN / BEARBEITEN

Um zwischen den Schirmen zu navigieren, geben Sie das     PM-EINTRAGEN
entsprechende Kommando ein
```

The panel is now shown with the edited subpanel. It consists of three parts, of which only the work area has not been edited. The same subpanels will also be used in other panels whenever the defined areas match.

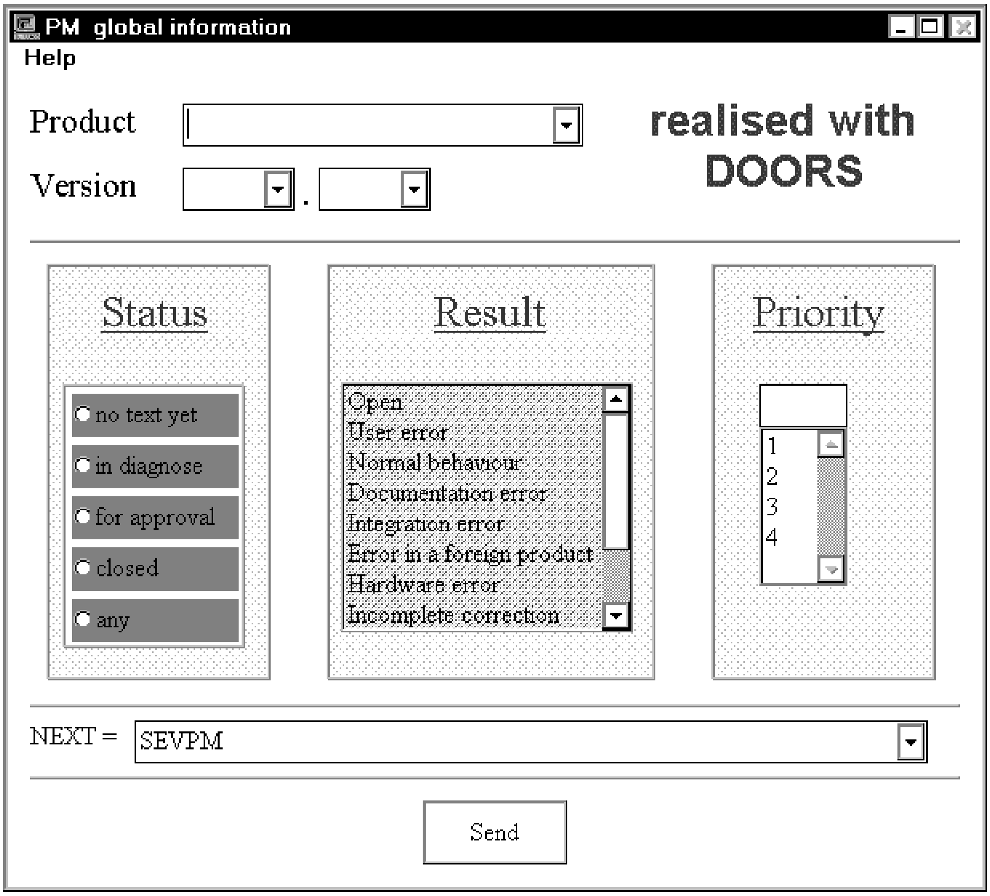
► Click on *OK* to display the fourth panel of the sample application. The footer panel will be reused in the display, since the contents of the two areas match. The contents of the header area differ, so the header of the fourth panel is converted automatically.

► Confirm the fourth panel with *OK* to display the fifth panel. The optimized header and footer panels are used here.

► Confirm this panel with *OK* as well. The sixth panel of the sample application will appear.

- ► Create a capture for this panel with the name SEGPM_Z; see section “Creating a capture” on page 43 for details.

As soon as you confirm the new capture with *OK* in the *Capture Management* dialog box, the already optimized *PM Global Information* panel will be displayed instead of the automatically converted one.



The difference between the original and optimized panels can be seen by checking the emulation window, which retains the original format while executing an application under WIN-DOORS.

4.6 Using macros

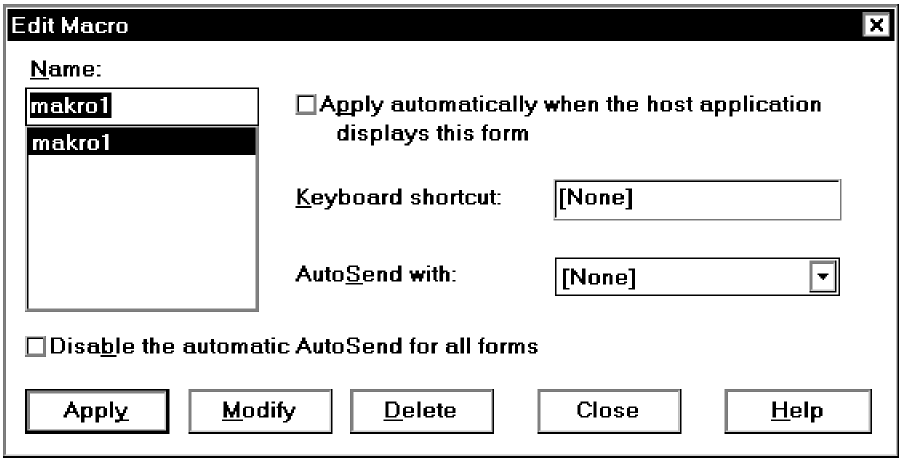
Macros allow you to automate the process of making inputs in the panel. This is done by first recording individual steps in the panel and then assigning the macro to the panel. This feature has several advantages, as you will see below.

4.6.1 Creating a macro

► In the optimized *PM Global Information* panel, select the entry *WIN-DOORS-ED* from the *Product* list.

► Select the radio button *any* from the *Status* group.

► In the main menu of WIN-DOORS, select the *Tools/Macro* command. The *Macro* dialog box appears.



Edit Macro

Name: makro1

makro1

☐ Apply automatically when the host application displays this form

Keyboard shortcut: [None]

AutoSend with: [None]

☐ Disable the automatic AutoSend for all forms

► Assign the macro a name, e.g. macro1, and click *Add*. The dialog box is closed. You have now created your first macro, which recorded the last few steps in the optimized panel.

► In the optimized *PM Global Information* panel, select the entry *WIN-DOORS-PC* from the *Product* list.

► Select the radio button *for approval* from the *Status* group.

► Select the entry *Software Error* from the *Result* list.

► Save these three steps in a second macro.

4.6.2 Assigning a macro

- ► Ensure that the mouse cursor is placed on the panel and press the right mouse button. The context menu is opened.
- ► Select the first macro. The settings that were recorded with this macro are made in the panel. Selecting the second macro causes the settings recorded in that macro to be made. You can now switch between the two states in the panel before confirming the entries with *Send*.

4.7 Subpanels

On sending the *PM Global Information* panel with *Send*, the *PM list* secondary window (or subpanel) appears. The emulation window can be used once again to verify what information of the form is contained in this panel. If a form contains too many units of information, it may be worth splitting it up and distributing its content over multiple subpanels.

- ► Close the subpanel with *Cancel*.

A Help panel was also created for the *PM Global Information* panel.

- ► Open the Help panel via the context menu. You can create such subpanels and Help panels for your applications as well.
- ► Close the Help panel with *OK*.
- ► Exit the sample session from the main menu of WIN-DOORS by choosing *Session/Exit*.

5 Interfaces

WIN-DOORS offers you the option of creating and linking libraries for form recognition that are specially customized to your application as well as an option to have data from your host application processed directly by other programs on a PC via the OLE object “Auto9750.Connection”. These two interfaces are described below.

5.1 Library for form recognition

When you run your application, WIN-DOORS usually recognizes forms sent by the host application automatically and looks in the resource directory on the PC or in a form library in BS2000 for an appropriate panel to be used in the display. If now predefined panel is available, the forms are automatically converted and displayed on the PC.

In some cases, however, it may be expedient to develop a separate algorithm for form recognition, especially if your application processes a lot of dynamic data, i.e. when querying a database. WIN-DOORS provides you with a skeleton program for this purpose as well as some predefined functions with which this skeleton program can be processed.

Prerequisite

In order to develop your own dynamic library for form recognition, you will need a C++ compiler to edit and compile the skeleton program.

Procedure

► Switch to the installation directory of WIN-DOORS (default: *C:\WINDOORS*) and then to the subdirectory *wind/hook*. This directory contains the skeleton program *Hook_gen.cpp*, which is written in C++, and the following files to create a dynamic library:

- *Hook_gen.h* (header file with predefined functions)
- *Hookdata.h* (header file with data structures)
- *Pandata.h* (header file with data structures)
- *Hook_gen.def* (definition file)
- *Hook_gen.mak* (makefile with a list of these files)

- ► Extend the *GenerateRow()* function in the skeleton program with the predefined functions and adapt it to your requirements.
- ► Compile the program with a C++ compiler into a dynamic library (DLL).
- ► When you create a new capture, activate the option *Call DLL* in the *Tools/Capture* dialog box.
- ► Specify the library in the *DLL for Form Recognition* entry field of the *Capture Management* dialog box.
- ► Alternatively, you could also enter the name of the library directly in the parameter file for your application. This can be done by entering the following line under the *[misc]* section in the *.drs* file:

  HookDllPath="library-name**"**

5.1.1 The GenerateRow() function

The *GenerateRow()* function is defined in the *Hook_gen.cpp* skeleton program as follows:

```
void GenerateRow(PanelHookData *pData, char *pszText, char *pszAtts);
```

**pData*	Points to an array with form descriptions.
**pszText*	Points to a vector containing the text of the current line in the form.
**pszAtts*	Points to a vector containing the attributes of the current line in the form.

You can process any line in the form with the predefined functions.

Macros

The following test functions can be logically combined with the *AND* and *OR* macros. In addition, you can navigate within the form by using the macros below:

PREV_LINE	Points to the previous line in the form, relative to the current line.
THIS_LINE	Points to the current line.
NEXT_LINE	Points to the next line in the form, relative to the current line.

You can use these macros to identify a line position in the form in the functions described below just like pure integer values. The macros always refer to line specifications relative to the current line in the form.

Example

If the current line in the form is line 5, for example, you can switch to line 4 with the *PREV_LINE* macro. This could also be achieved by specifying -1.

Make sure that the macros or integer specifications do not address lines outside the form. The maximum number of lines in the form is restricted by the screen size. The line count begins with 1.

Variables

If you need your own variables to specify dependencies between different lines in the form, you should define such variables as static, since the line values would otherwise be overwritten. Furthermore, all such variables should only be locally accessible for the corresponding *CAPTURE_IS()* function and be initialized on processing the first line.

5.1.2 Predefined functions for GenerateRow()

The predefined functions for the *GenerateRow()* function are subdivided into

– test functions

– processing functions

Test functions

All test functions return the value TRUE or FALSE as a result.

`BOOL ATTRIBUTE_IS(int x, char c)`

Tests whether the attribute at position *x* of the current line has the value *c*.

`BOOL CAPTURE_IS(string name)`

Tests whether the name of the currently processed capture is *name*.

`BOOL COLUMN_IS(int x, char c)`

Tests whether the character at position *x* of the current line has the value *c*.

`BOOL COLUMNS_ARE(int x1, int x2, string s)`

Tests whether the string between positions *x1* and *x2* of the current line matches string *s*.

`BOOL EMPTY(int n)`
Tests whether line *n* is empty, where *n* is counted relative to the current line.

`BOOL LINE_BETWEEN(int n1, int n2)`
Tests whether the current line lies between lines *n1* and *n2*.

`BOOL LINE_IS(int n)`
Tests whether the current line is line *n* of the form.

`BOOL NOT_EMPTY(int n)`
Tests whether line *n* contains one or more characters, where *n* is counted relative to the current line.

`BOOL NULL_LINE(int n)`
Tests whether line *n* has the value 0, where *n* is counted relative to the current line.

`BOOL NOT_NULL(int n)`
Tests whether line *n* contains a non-zero value, where *n* is counted relative to the current line.

`BOOL XCOLUMN_IS(int n, int x, char c)`
Tests whether the character at position *x* in line *n* is the character *c*.

`BOOL XCOLUMN_ARE(int n, int x1, int x2, string c)`
Tests whether the character between positions *x1* and *x2* in line *n* matches the string *s*.

Processing functions

`char *GET_DSP_ATTS(int n)`
Returns a pointer to the display attribute of line *n* as a result, where *n* is counted relative to the current line.

`char *GET_SEM_ATTS(int n)`
Returns a pointer to the semantic attribute of line *n* as a result, where *n* is counted relative to the current line.

`char *GET_TEXT(int n)`
Returns a pointer to the text of line *n* as a result, where *n* is counted relative to the current line.

`void PANEL_SET(string name, int n)`

This function is used to manage subpanels and to fill the array *pData*, which is defined in the *GenrateRow()* function. It determines the subpanel with *name* and *n* number of lines that is required for displaying and processing the next line. If the interaction between subpanels is involved, you should make sure that a resource file has been assigned with the *PANEL_SET()* function for every line of the form. Without this assignment, nothing will be displayed on the screen for the line, and no action will be executed.

Example

```
void GenerateRow(...)
{
if (CAPTURE_IS("capture1))
   {
   if (LINE_IS(n))
      {
      if (COLUMNS_ARE(...))
         PANEL_SET("panel1",3);
      }
   else if (LINE_BETWEEN(n1, n2))
      {
      if (COLUMN_IS(...))
         PANEL_SET("panel2",1)
      else
         PANEL_SET("panel3", 1)
      }
   }
   else if (CAPTURE_IS("capture2"))
      {
      ...
      }
   return;

}
```

5.2 OLE automation

OLE allows WIN-DOORS to offer you an interface which gives you simple access to the host application at program level.

This provides you, for instance, with a simple method of inserting fields from BS2000 forms in your Excel spreadsheets or to control work processes with a Visual Basic script or a C++ program.

What is OLE automation?

OLE automation allows you to process the OLE automation objects provided with WIN-DOORS from other programs. This means that you can write a program outside WIN-DOORS (e.g. an Excel V5.0 script) which accesses the fields in a host application.

The OLE automation objects are instances of classes. Each object provides properties and methods. Properties can be read and set. Methods are functions which can be called from outside the object.

The OLE automation controller includes actions such as creation of the object, the setting of properties and the invocation of methods.

OLE automation ensures that parameters are passed when methods are called across different applications and that the correct values are returned.

WIN-DOORS OLE automation objects

WIN-DOORS makes one OLE automation object of the type "Auto9750.Connection" available.

You can use "Auto9750.Connection" in any macro language or programming language which provides support for OLE automation. This includes in particular the following languages:

– Visual C++

– Visual Basic

– Visual Basic for Applications (programming interface for Excel V5.0, for instance).

5.2.1 The OLE automation object “Auto9750.Connection”

The OLE automation object "Auto9750.Connection" represents a BS2000 connection to a host application. It offers you methods

– for defining a connection to the BS2000 host and opening and closing this connection

– for manipulating BS2000 forms directly, either by entering data in entry fields or by selecting fields

– for fetching data from BS2000 forms.

If you set the connection properties for the Auto9750.Connection object, you define the name of the connection (identifier), the emulation to be used and the parameters for the emulation.

i If a method returns "TRUE", this does not necessarily mean that the action has been successfully executed on the host computer, but only that the request has been accepted by the emulation used.

There are three properties for this object: *ConnectionIdentifier*, *ConnectionParameters* and *ConnectionCmdLine*. These three properties allow direct access to the parameters of the SetConnection() method.

All methods which require a connection generate exceptions (with a value of 1000) if no connection has been established. This exception can be intercepted using the ON ERROR statement in Visual Basic, for instance.

Registration

OLE automation objects must be “registered” to allow them to be accessed from other applications.

The WIN-DOORS installation program registers the WIN-DOORS OLE automation objects automatically. You can repeat the registration of the WIN-DOORS OLE automation objects (e.g. if the Registry is faulty or has been deleted inadvertently) using the OLE automation server *AUTO9750.EXE*.

Restrictions

OLE automation provides the option of having the OLE server inform the OLE client when an event occurs (e.g. the arrival of a new screen). This means that the Visual Basic for Applications script must be written in such a way that screen navigation is implemented with the WaitUntil(...) or WaitUntilCondition() method.

Type libraries

The WIN-DOORS OLE automation objects are declared in the following files:

– A9750DEU.TLB (German)
– A9750ENG.TLB (English)

These files can be read by compilers (e.g. Visual C++) and browsers (e.g. OLE2VIEW) to allow the objects to be displayed and accessed.

5.2.2 Methods for “Auto9750.Connection”

`BOOL ActivateEmulation()`

Activates the window in which the connection to the host application is displayed. Returns TRUE if the connection exists and the window could be activated; otherwise, FALSE.

`BOOL CloseConnection()`

Closes the connection.
Returns FALSE if the connection could not be closed or if the connection was not open. Returns TRUE in all other cases.

`BOOL DefineCondition(short ConditionIdf, short Line, short Column, LPCSTR String)`

Defines a condition for the WaitUntilCondition() method defined immediately afterwards. The condition has a numeric identifier (*ConditionIdf*), a *Line/Column* coordinate and a string (*String*).
ConditionIdf must be greater than 0.
Line and *Column* start at 1.
If 0 is specified for *Line*, the condition applies to all lines.
Returns TRUE if the condition is accepted and stored in memory. Returns FALSE in all other cases (*ConditionIfd* is less than or equal to 0, a condition with the same *ConditionIdf* has already been defined, invalid *Line/Column*).

`BSTR GetArea(short Line, short Column, short Length)`

Reads a number of characters defined by *Length* from the position on the emulation screen defined by *Line/Column* and returns these characters in a string. If the end of the line is reached, the characters in the following line are read until the number of lines defined by *Length* have been read or until the end of the screen is reached.
$1 \leq Line \leq 24$ and $1 \leq Column \leq 80$.
An empty string is returned if no connection is open or if the *Line/Column* specifications are invalid or if it was not possible to read the emulation window.

`short GetCursorColumn()`

Returns the column of the current cursor position in the form. Column numbering begins with 1.

`short GetCursorLine()`

Returns the line of the current cursor position in the form. Line numbering begins with 1.

`BOOL IsConnected()`

Returns TRUE if a connection is open. Returns FALSE in all other cases.

`BOOL IsScreenModified()`

Returns TRUE if the contents of the emulation window have been changed since the last time the GetArea() method was called. Returns FALSE in all other cases.

`BOOL Mark(short Line, short Column)`

Selects (marks) the field at the screen position in the emulation window defined by *Line/Column*.
Line and *Column* start at 1.

If 0 is specified for *Line* or *Column*, the cursor is not moved in the emulation window. The field at the current position is then selected.
Returns TRUE if the field was selected successfully. FALSE is returned in all other cases (no connection, invalid *Line/Column*, errors generated by the emulation).

`BOOL OpenConnection()`

Establishes a connection previously defined with SetConnection().
If the parameters and/or the program required for establishing a connection have not been defined, a standard dialog box is displayed by the dispatcher and the user can enter the missing values.
OpenConnection uses the connection with the specified identifier if this connection has already been established. If not, it establishes a new connection.
Returns FALSE if a connection for an object of the type "Auto9750.Connection" has already been established or if the connection could not be established. In all other cases, TRUE is returned.

`BOOL SendScreen(LPCSTR Key)`

Sends the screen off to the host application using the key specified by *Key*.
The following values are permitted for *Key*: DUE1, DUE2, F01 ... F24 and K01 ... K14.
Returns TRUE if the screen was sent successfully. Returns FALSE in all other cases (no connection, invalid *Key*, errors generated by the emulation).

`BOOL SetArea(short Line, short Column, LPCSTR Text)`

Writes the string defined by *Text* to the screen location in the emulation window defined by *Line/Column*. Any existing data at this location is overwritten by *Text*.
Line and *Column* start at 1.
If 0 is specified for *Line* or *Column*, the cursor is not moved in the emulation window. *Text* is then written at the current position.
Returns TRUE if *Text* is written successfully. FALSE is returned in all other cases (no connection, empty *Text*, invalid *Line/Column*, errors generated by the emulation).

`void SetConnection(LPCSTR Name, LPCSTR Parameters, LPCSTR Program)`

Sets the parameters required for a connection. SetConnection only defines the connection and does not yet establish it. Use the OpenConnection() method to establish the connection.

Name: Identifier for the connection
Parameters: Parameters for the connection; name of the parameter file
Program: File name of the emulation program to be used

`BOOL SetCursorPos(short Line, short Column)`

Sets the cursor to the specified position in the form, where line and column numbering begins with 1. Returns TRUE if the cursor could be successfully positioned; otherwise, FALSE (e.g. for an invalid *Line/Column* or a disconnect).

`void SetTimer(short Limit)`

Defines a timeout (in seconds) indicating the latest time after which a WaitUntil/WaitUntilCondition method should return a result.
The preset value for *Limit* is 60 seconds.

`void Wait(short Seconds)`

Interrupts the script/programs for the period specified by *Seconds*.

`BOOL WaitUntil(short Line, short Column, LPCSTR ComparisonValue)`

Waits until the screen area in the emulation window defined by *Line/Column* corresponds to the string given in *ComparisonValue*.
Line and *Column* start at 1.
If 0 is specified for *Line*, all lines are compared.
A result is only returned if the condition is fulfilled or if an error occurs.
Returns TRUE if the condition is fulfilled. Returns FALSE in all other cases (invalid *Line/Column* or TIMEOUT - see also SetTimer() method).

`short WaitUntilCondition()`

Waits until a condition defined previously with the DefineCondition() method is fulfilled.
Returns -1 of no condition has been fulfilled.
Returns 0 if a timeout occurs (see also the SetTimer() method).
Returns the identifier for the condition which has been fulfilled (as defined with DefineCondition()). If a number of conditions are fulfilled simultaneously, the smallest identifier is returned.
The list of conditions defined previously using DefineCondition() is reset before a result is returned. This means that the conditions must be redefined for e new WaitUntilCondtion query.

Example

The following example illustrates the use of Auto9750.Connection in an Excel table (“Visual Basic for Applications” script).

```
Option Explicit
' Declaration of the global variables

   Dim myconn As Object
   Dim rc As Integer
   Dim msg As String
   Dim connect As Boolean

Sub start_BS2000()
' Start a BS2000 connection
   Dim pass As String

   If connect = True Then
      MsgBox "Connection already started"
      Exit Sub
```

```
    End If

    On Error GoTo err

    Set myconn = CreateObject("Auto9750.Connection")

    myconn.SetConnection "TEST", "bs2000.mts", "c:\mt9750w\mt9750.exe"
    ' Set connection parameters

    myconn.OpenConnection
    ' Open connection

    rc = myconn.WaitUntil(0, 1, "/ ")
    ' WAit for LOGON prompt

    If rc = 0 Then
       MsgBox "Time out"
       Exit Sub
    End If

    rc = myconn.SetArea(0, 0, ".JOBNAME SET-LOGON-PARAMETERS
            uid,account,'password'")
    ' LOGON ...

    rc = myconn.SendScreen("due1")
    ' and send the screen

    connect = True
    MsgBox "Connection established."
    Exit Sub

err:

    MsgBox "Error establishing connection."
    connect = False

End Sub

Sub file_status()
' Issue SHOW-FILE-ATTRIBUTE command
' Output is sent to the first two columns of the Excel spreadsheet

    Dim i As Integer
    Dim j As Integer

    If connect = False Then
       MsgBox "Connection not started"
       Exit Sub
    End If

    On Error GoTo err

    rc = myconn.SetArea(0, 0, "SHOW-FILE-ATTRIBUTE * ")
    ' SHOW-FILE-ATTRIBUTE ...
    rc = myconn.SendScreen(„due1“)
    ' and send the screen

    rc = myconn.WaitUntil(2, 1, "%    ")
    ' Wait for response

    myconn.SetTimer (10)

    ' The following loop
    ' traps the “PLEASE ACKNOWLEDGE” prompt
    ' automatically

    i = 2
    j = 2

    While myconn.GetArea(i, 1, 1) = "%" And myconn.GetArea(i, 2, 1) <> ":"
       If myconn.GetArea(i, 1, 7) = "%PLEASE" Then
          i = 1
          rc = myconn.SendScreen("DUE1")
          rc = myconn.Wait(10)
          ' Pause
       Else
          Cells(j, 1) = myconn.GetArea(i, 2, 8)
          ' Copy file size
          Cells(j, 2) = myconn.GetArea(i, 11, 54)
          ' Copy file name
          i = i + 1
          j = j + 1
       End If
    Wend
    Exit Sub

err:
    MsgBox “Connection error”
    connect = False

End Sub

Sub stop_BS2000()
' Stop BS2000 connection

    If connect = False Then
       MsgBox "Connection not started"
       Exit Sub
    End If

    rc = myconn.SetArea(0, 0, "exit-job")
        rc = myconn.SendScreen("due1")
        connect = False

End Sub
```

6 Installation and configuration

6.1 Hardware and software requirements

Hardware requirements

– A PC with a 386 or better processor (486 processors or Pentiums are recommended) and at least 4 MB RAM or a BS2000 Client PC (MFT2)

Software requirements

MS-DOS/Windows:

- 16-bit version
  - MS-DOS as of V5.0 and Microsoft Windows V3.1x
  - Dialog Builder runtime system as of V2.2; 16-bit version (included in the WIN-DOORS delivery package)
- 32-bit version
  - Windows 95 or Windows NT as of Version 4.0
  - Dialog Builder runtime system as of V2.2; 32-bit version (included in the WIN-DOORS delivery package)
- Communication Layer (TRCOMMS is included in the WIN-DOORS delivery package)
- An emulation that supports the DOORS-DDE protocol (DOORS emulation is included in the WIN-DOORS delivery package)

BS2000:

- BS2000-BC as of V9.5 or BS2000/OSD as of V1.0

6.2 Installing WIN-DOORS

Place the CD-ROM with WIN-DOORS in the CD-ROM drive and start the *setup* program.

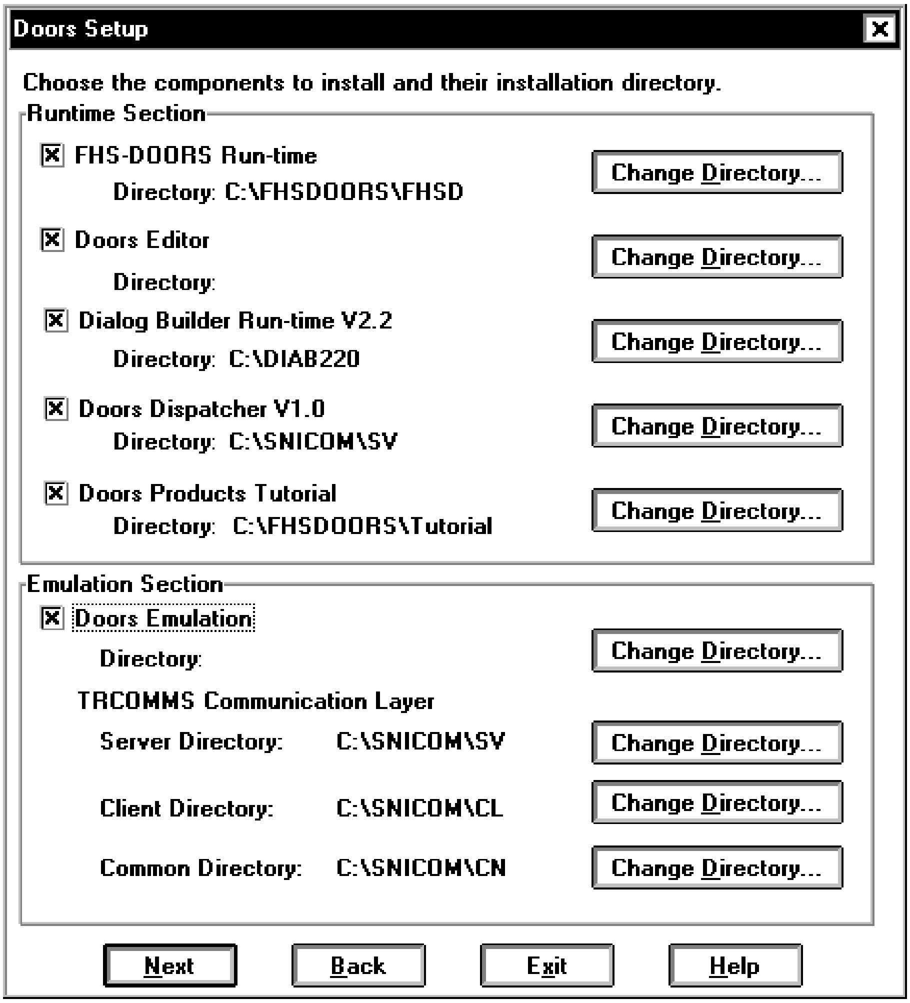
Windows V3.1x	Select the *File/Run...* command from the Windows Program Manager and enter the following in the command line: **drive-letter:setup** This starts the installation program.
Windows 95	Select *Settings*$\rightarrow$*Control Panel* from the Start menu. In the Control Panel, double-click on the *Add/Remove Programs* icon and then click the *Install* button. Follow the installation instructions displayed on the screen.

The following installation methods can be used to install WIN-DOORS:

– *Developer installation*
  This includes the installation of the WIN runtime system and the development environment needed to provide BS2000 applications with a graphical interface. In order to use this installation method, you will need a password, which is included in the supplied documentation.

– *User installation*
  This includes the installation of all WIN-DOORS components that are needed to display the converted BS2000 applications.

In both cases, the following window will be subsequently displayed so that you can define the scope and the directories for the installation.



Doors Setup

Choose the components to install and their installation directory.

Runtime Section

[✓] FHS-DOORS Run-time
Directory: C:\FHSDOORS\FHSD
Change Directory...

[✓] Doors Editor
Directory:
Change Directory...

[✓] Dialog Builder Run-time V2.2
Directory: C:\DIAB220
Change Directory...

[✓] Doors Dispatcher V1.0
Directory: C:\SNICOM\SV
Change Directory...

[✓] Doors Products Tutorial
Directory: C:\FHSDOORS\Tutorial
Change Directory...

Emulation Section

[✓] Doors Emulation
Directory:
Change Directory...

TRCOMMS Communication Layer

Server Directory:	C:\SNICOM\SV	Change Directory...
Client Directory:	C:\SNICOM\CL	Change Directory...
Common Directory:	C:\SNICOM\CN	Change Directory...

The table below shows which components are included in the WIN-DOORS package and in which directories the individual components are installed during a default installation.

Components	**Default directory**	
	16-bit	**32-bit**
WIN-DOORS runtime system	C:\WINDOORS\WIND	C:\Programs\ WINDOORS\WIND
Dialog Builder runtime system	C:\DIAB220	C:\Programs\DIAB232
DOORS Dispatcher	C:\SNICOM\SV	C:\SNICOM\SV
DOORS Emulation	C:\DOORS_EM	C:\Programs\DOORS_EM
DOORS Editor	C:\WINDOORS\DOORS_ED	C:\Programs\ WINDOORS\DOORS_ED
TRCOMMS Communication Layer	C:\SNICOM\CL (driver) C:\SNICOM\CN (dir1 file) C:\SNICOM\SV	C:\SNICOM\CL (driver) C:\SNICOM\CN (dir1 file) C:\SNICOM\SV

Table 5: Default directories for the WIN-DOORS installation

6.3 Configuring the emulation

WIN-DOORS handles communication with BS2000/OSD via 9750 emulation. This emulation must support the DDE protocol. The following emulations support this protocol:

– DOORS Emulation as of Version V1.1
– MT9750 as of Version V4.0C
– MPS as of Version V3.0
– LOG-TE as of Version V4.2
– CONWARE as of Version V4.5

The DOORS Emulation Version V1.1 or later that is supplied with WIN-DOORS is sufficient to set up the connection with BS2000. The method used to configure an emulation is described below using the DOORS emulation as an example. If you plan to use some other emulation, the environment variables, entries and parameters may differ. For a description of the parameters, refer to the documentation supplied with the emulation.

► First verify that the environment variable *CMXPATH* for the Communication Layer and the variables *DIABUILD_HOME*, *LANG*, *DIABPATH* and *BMPATH* for the Dialog Builder have been set in the *autoexec.bat* file. The following (or similar) entries must be made for these variables in the *autoexec.bat* file:

```
SET CMXPATH=C:\SNICOM\CN
SET DIABPATH=C:\DIAB220\%%L\%%F
SET DIABUILD_HOME=C:\DIAB220
SET BMPATH=C:\DIAB220\BITMAP
SET LANG=DE
```

Ensure that the installation path for both programs is included in the *PATH* variable.

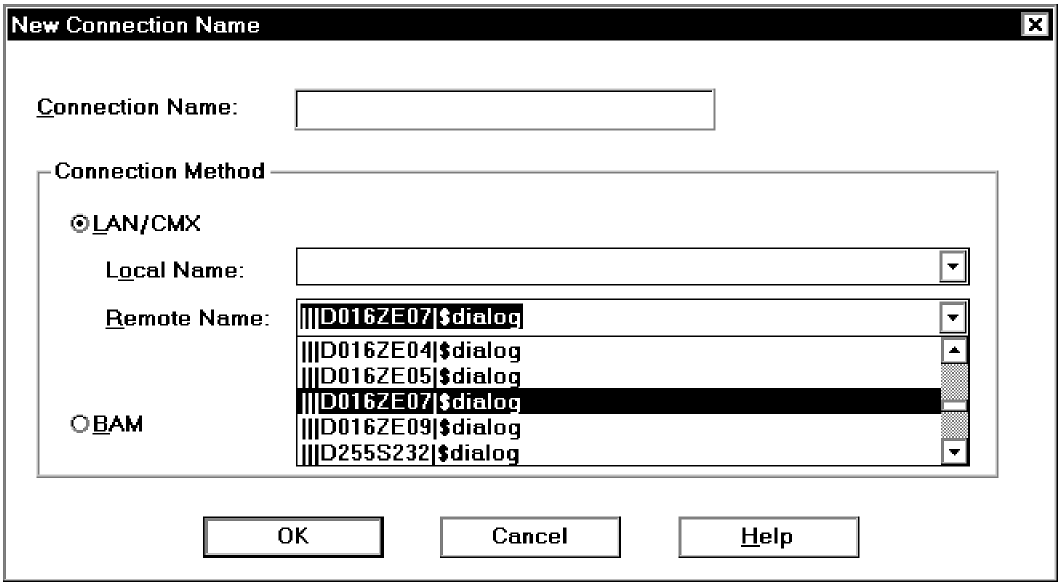
► If you are working via LAN1, you should also check whether the BS2000 host is entered in the transport name directory file *dir1* and whether this file can be accessed via the *PATH* variable (extend the *PATH* variable with the entry *C:\SNICOM\SV* if required). You will find more information on this subject in the manual for the DOORS Emulation V1.1 [3] or the communication manager CMX [9].

6.3.1 Creating a parameter file for the emulation

► Start the DOORS emulation by double-clicking on the program icon in the WIN-DOORS program group.

► Activate the *Configuration/Connection Name* command to set up a new connection.

► To set up the connection with the BS2000 host, click the *Setup* button in the *New Connection Name* dialog box.


New Connection Name

Connection Name:

Connection Method

(● LAN/CMX

Local Name:

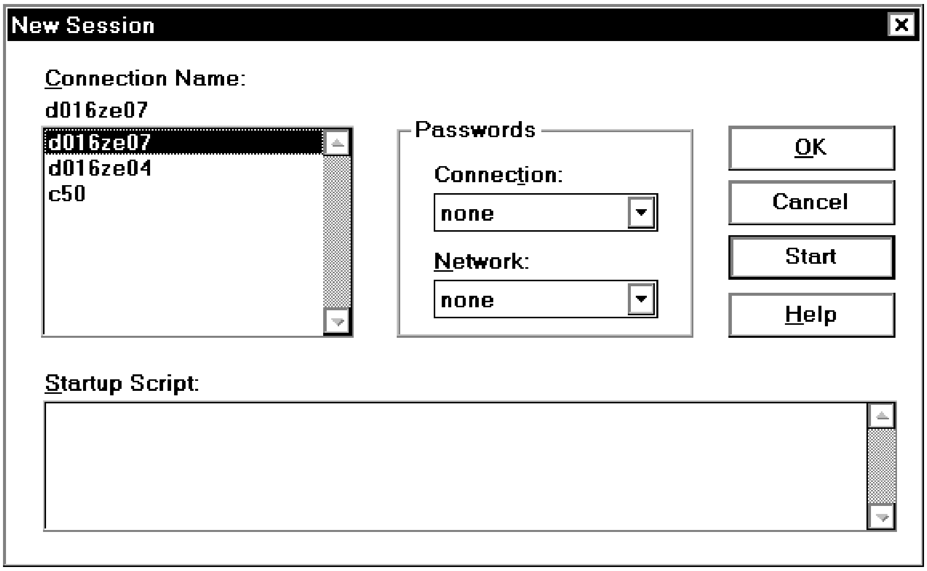
Remote Name: |||D016ZE07|$dialog

|||D016ZE04|$dialog
|||D016ZE05|$dialog
|||D016ZE07|$dialog
|||D016ZE09|$dialog
|||D255S232|$dialog

○ BAM

OK Cancel Help

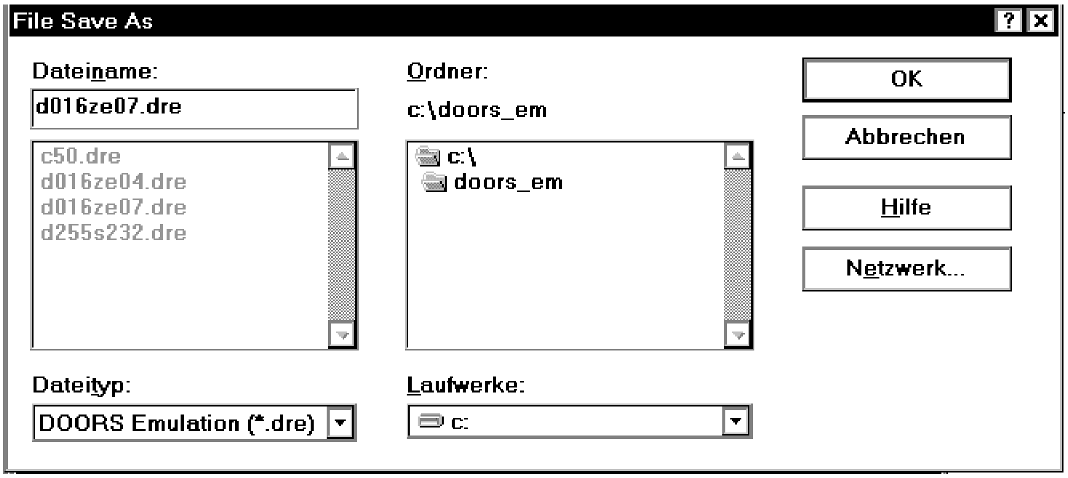
- ► Enter the name of the new connection (e.g. the name of the BS2000 host) in the *Connection Name* field of the *New Connection Name* dialog box. This name is freely selectable and may be up to 8 characters in length.
- ► Then select the connection type. In this example, you should select a LAN1/CMX connection.
- ► The *Remote Name* list shows all entries that are present in the transport name directory file *dir1*. Select the entry for the BS2000 host on which your application program is installed.
- ► Close both dialog boxes with *OK*.
- ► To configure a new session, activate the *File/New* command. The *New Session* dialog box appears.



- ► The *Connection Name* list in the *New Session* dialog box shows the newly installed connection (*D016ZE07* in this case). This connection will be already marked as selected.

  When you click *OK* in this dialog box, only an emulation window appears. Clicking on *Start* causes the connection to the specified host to be established.
- ► Click on either the *OK* or *Start* button. An emulation window with or without a connection will be displayed.

► Close the emulation window using the *Close* command of the window menu. A dialog box will appear asking you whether you want to save the session parameters. Confirm this dialog box with *OK*.



► Enter a name for the new session. The *.dre* suffix is automatically appended to the name of the configuration file. In this example, the configuration file is given the name of the BS2000 host on which the application program is installed: d016ze07.dre.

► Exit the emulation with *File/Exit*.

6.3.2 Potential problems

Depending on the BS2000/OSD application, the required terminal type may vary:

– WIN-DOORS and Desk2000 expect a 9763 terminal type. The only emulation that behaves like a 9763 terminal in the default configuration and thus requires no changes is the DOORS Emulation.

  All other emulations behave like a 9750 terminal and must therefore be adapted:

  ► Set the terminal type to 9763 in the respective emulation being used.

  ► If you are working with a BAM connection, you will also need to adapt the PDN.

– All other BS2000/OSD applications expect a 9750 terminal type. Consequently, only the DOORS Emulation needs to be adapted in this case:

- ► To do this, open the parameter file (with the *.dre* extension) and add the following line in the *[connection]* section:

  StationType=9750

- ► If you are working with a BAM connection, you will also need to adapt the PDN.

6.4 Configuring WIN-DOORS

WIN-DOORS can be configured by using the commands in the *Setup* menu of WIN-DOORS. The settings are saved in the *wind.ini* file:

i Whenever a new session is opened, WIN-DOORS uses default values from the *wind.ini* file. If you want to change these default values:

– open a new session (see section “Opening a new session” on page 29 for details)
– configure the session
– save the session under the file name *wind.ini* in the Windows directory

6.5 Uninstalling WIN-DOORS

If you are working with the 32-bit version and Windows 95 or Windows NT, select *Settings*→*Control Panel* from the Start menu. In the Control Panel, double-click on the *Add/Remove Programs* icon and then click on the *Uninstall* button. Follow the uninstallation instructions displayed.

If you are working with the 16-bit version and Windows V3.x, run the program *deinstall.exe* from the installation directory of WIN-DOORS by double-clicking on it from within the File Manager or by entering the path and program name in the *File/Run* command of the Program Manager. This starts the Uninstall program of WIN-DOORS. Follow the program instructions displayed.

6.6 Notes on the Dialog Builder runtime system

The Dialog Builder runtime system is configured using two configuration files, *diabuild.ini* and *dbrun.ini*. To avoid problems with panel presentation under WIN-DOORS, you must observe the following instructions:

– In *diabuild.ini* (in the Dialog Builder directory), the default alignment must be set to left-justified, and a default font must be specified:

```
[DiaBuild]
DefaultFont="-*-courier-*-*---*-0090-*-*-m-0000-WIN-DOORS-*"
DefaultAlignment=XmALIGNMENT_BEGINNING
```

– In *dbrun.ini* (in the Windows directory), the following lines must be added:

```
[XtXm]
Alignment=XmALIGNMENT_BEGINNING
Background=LightGrey
```

– The default font set using the WIN-DOORS *Setup/Font*... function must be the same as that defined in *diabuild.ini* (which in the above example would be WIN-DOORS, 9-point).

Glossary

active window
Only one window can be activated at one time. All actions relate to the active window. Clicking on a visible part of a window or selecting the entry for that window from the Window menu makes that window the active window.

alphanumeric object
An alphanumeric object is a reduced representation of the form window that requires fewer system resources at runtime. It is displayed alphanumerically in lines, columns, height and width and covers a selected area of the form window. This area must not contain any truncated fields.

application
An application is a program used for a specific task.

attribute
Attributes define the properties of objects. Attributes can refer to aspects such as the color, size and position of an object.

BAM
Bit-oriented Asynchronous Multipoint procedure; a connection method for setting up a connection to a remote host across a network.

callback
A callback is assigned to an object and describes a so-called higher-level event (e.g. "pushbutton pressed").

capture
A selection mechanism that is activated using the *Options/Capture* command. A capture requires a form identifier.

click on
Clicking on is a mouse technique combining the actions of pointing and clicking.

clicking
Clicking is a mouse technique which involves pressing and quickly releasing a mouse button.

clipboard

A temporary storage area used to move or copy data between applications.

command

An entry in a menu which initiates actions. Click on the command in the menu to select it.
To help you to find commands, the menu is separated from the command by a slash in the description. Thus, for instance, *Session/New* indicates that you should select the *New* command from the *Session* menu.

connection method

A method such as LAN, BAM or HDLC/AFP, used to set up a connection to a remote host across a network.

connection name

Connections are identified by their connection names.
A connection name is an alias for the parameters of a connection to the host application. A connection name contains the following information: the connection method, the local name, the remote name and whether the connection is active or passive.

control menu

The control menu is located in the title bar and contains commands for moving, sizing, maximizing or minimizing windows.

cursor

The cursor is an on-screen symbol which identifies the current position within the screen. See also *mouse pointer* and *insertion point*.

dialog box

A dialog box is a window-like area which provides information or allows you to make settings.

double-click

Double-clicking is a mouse technique in which a mouse button is clicked twice in rapid succession.

dragging

Dragging is a mouse technique in which the mouse is moved with one of its buttons held down.

dre file

Contains the settings for a connection to the DOORS emulation; also called the parameter file for the DOORS emulation.

drs file

Contains the settings for an WIN-DOORS session; also called the parameter file for WIN-DOORS.

field

Fields are used to output data from or input data to the application. Fields can be: output fields, entry fields, option buttons (radio buttons), check boxes or drop-down combo boxes (not for forms).

form

A form (sometimes referred to as a mask) is displayed on screen and allows data to be output from and input to an application.

form identifier

The form identifier is a square area in the form with which WIN-DOORS can uniquely identify the form. WIN-DOORS uses the content of the area (text) for this purpose. It is usually best to select the “form name” as the identifier. An identifier can be defined by using the left mouse button in a capture.

HDLC/AFP

A connection method for setting up a connection to a remote host across a network. This connection method employs the HDLC (high-level data link control) protocol and the AFP (alternating pulse-edge modulation) method.

host application

Application which runs on a remote computer in a network.

icon

Symbol which appears on the screen to represent items such as applications, files, commands or buttons.

insertion point

The insertion point indicates the position at which data will be inserted in text or graphics. See also *cursor*.

LAN/CMX

Local Area Network; a connection method for setting up a connection to a remote host across a network. LAN/CMX covers a number of different communication methods, such as LAN1, Win-Sockets, LAN-Manager, LAN/CMX, WAN/CMX.

list box

A list box contains a list of items (such as file names). A list box allows you to select one of the items by clicking on it.
A drop-down list box is a variant which saves space, since only the list item which is currently valid is visible at all times. If you click on the arrow to the right of the combo box, this opens or closes the portion of the list which is not always visible.

menu

A menu is an on-screen list of grouped commands represented by menu items. Selecting one of the items causes the associated command to be executed.

modal dialog box

A modal dialog box is one in which input is mandatory. The next step in the dialog will not be executed until the user supplies input.

mouse

A mouse is an input device equipped with two or three buttons. Moving the mouse causes the mouse pointer to move on the screen. The mouse buttons are used to initiate a number of different actions: see *click*, *drag*, *double-click*.

mouse pointer

The mouse pointer is an on-screen cursor controlled by the mouse. Moving the mouse across your desk or a mouse pad causes the pointer to move on the screen.

object

Objects are uniquely identifiable processing units, which can be distinguished on screen. Objects can be pushbuttons, fields or list boxes, for instance.

panel

A *panel* is a graphical window on the PC stored in a semantics file (sdc file). The panel is generated from a form using WIN-DOORS or the form converter (BS2000/OSD).

partial panel

See subpanel.

point

Pointing is a mouse technique in which the mouse pointer is moved onto an object (such as a pushbutton).

press

Pressing is a mouse technique in which a mouse button is pressed and held down.

primary window

The primary window (or main window) is the window assigned to an application. It is the first window to appear after the application is started.

pushbutton

Pushbuttons are used to confirm entries and settings and to initiate further actions.

rbn file

An rbn file (resource binary file) contains the description of the layout of a form in binary form.

res file

A res file (resource file) contains the description of the layout of a form.

scroll bar

Scroll bars let you quickly move screen components up, down, left or right if only part of a window, field or list box is currently visible.

sdc file

An sdc file (semantic description compressed file) describes the graphical representation of a form in a host application. An sdc file contains a description of the form, the way in which it is represented graphically (panel) and the links between the graphical objects in the panel and the fields in the form as a semantic description.
sdc files are created using the WIN-DOORS Capture function. They can then be edited and optimized with the DOORS editor.

subdialog box

A subdialog box is a dialog box opened as an extension to a dialog box which is already open.

subpanel

A subpanel corresponds to a selected area of a form. This area can be selected with the right mouse button in the *Capture* dialog box. The subpanel is line oriented and has an identifier. A line of a form can only be used in a subpanel.

title bar

The title bar contains the control menu and the name of the window or dialog box and can be used to move the window around the screen.

window

Windows are rectangular, framed screen areas which divide the working area into mutually independent, resizeable and freely movable segments.

Abbreviations

AFP	alternating pulse-edge modulation (German acronym)
BAM	Bit-oriented Asynchronous Multipoint procedure
BCAM	Basic Communication Access Method (BS2000/OSD)
DCAM	Data Communication Access Method (BS2000/OSD)
FHS	Format Handling System (BS2000/OSD)
FT	File Transfer
HDLC	High-level Data Link Control
IFG	Interactive Format Generator (BS2000/OSD)
LAN	Local Area Network
rbn	binary resources as of Dialog Builder V2.1
res	FHS-DOORS resources
sdc	semantic description compressed; FHS-DOORS semantic description files
SDF	System Dialog Facility (BS2000/OSD)
TIAM	Terminal Interactive Access Method (BS2000/OSD)
UTM	Universal Transaction Monitor (BS2000/OSD)
WAN	Wide Area Network

Related publications

Please apply to your local office for ordering the manuals.

Windows

[1] **FHS-DOORS** (BS2000/OSD, MS-Windows)
Graphical Interface for BS2000/OSD Applications
User Guide

Target group
The manual addresses BS2000 developers who wish to equip BS2000 applications with a graphical interface.
Contents
The manual describes the usage model and the functions of FHS-DOORS. A sample session provides an example of how you work with FHS-DOORS. The manual also describes the parameters with which sessions can be configured as required by applications, and the interfaces for a library for format recognition and OLE automation. It also contains a description of the format converter FHS-DOORS-LC and of the Event Stream Service (ESS-DOORS).

[2] **WIN-DOORS/FHS-DOORS** (BS2000/OSD, MS-Windows)
Optimizing Panels with the DOORS Editor
User Guide

Target Group
The manual addresses BS2000 developers who wish to optimize formats for use under WIN-DOORS/FHS-DOORS.
Contents
The manual describes how converted formats can be processed using the DOORS Editor. It explains how you work with the DOORS Editor, and the op-tions available for user-specific extensions using the Dialog Builder. It also contains a reference section on the object attributes. A description of the interface covers the online help system for the DOORS Editor.

[3] **DOORS Emulation** (Windows, BS2000/OSD)
Basic Emulation 9750 for BS2000/OSD Connections
User Guide

Target group
The manual addresses users wishing to use DOORS emulation for connections to BS2000/OSD partners.
Contents
The manual describes the functions of DOORS emulation. It describes working with DOORS emulation, connection setup (connection names) and the use of startup scripts for automating connection setup. A full description of the DOORS emulation interface is included in the online help for DOORS emulation.

[4] **Dialog Builder**
Development Environment for MS-WINDOWS
User Guide

[5] **MT9750**
9750 emulation
User Guide

[6] **LAN1 V3.0** (MS-DOS)
User´s Guide

[7] **LAN1 V3.0** (MS-DOS)
Administrator´s Guide

[8] **LAN1 V3.0** (MS-DOS)
Configuration Guide

[9] **CMX** (MS-DOS)
Communication Manager User Guide

Target group
Users and network administrators
Contents
CMX is a transport access system that complies with ISO 8072 and enables communication between applications in PCS and other computer systems. It can be operated using either menus or commands.

TRANSDATA

[10] **FHS** (TRANSDATA)
User Guide

Target group
Programmers
Contents
Program interfaces of FHS for TIAM, DCAM and UTM applications. Generation, application and management of formats.

[11] **IFG for FHS** (TRANSDATA)
User Guide

Target group
Terminal users, application engineers and programmers
Contents
The Interactive Format Generator (IFG) is a system that permits simple, user-friendly generation and management of formats at a terminal. In conjunction with FHS, these formats can be used on the host computer. This user guide describes how formats are generated, modified and managed, plus also the new functions of IFG V8.1.

[12] ***open*UTM**
Concepts and Functions
User Guide

Target group
Anyone who wants information about the functionality and performance capability of *open*UTM.
Contents
The manual contains a general description of all the functions and features of *open*UTM, plus introductory information designed to help first-time users of *open*UTM.

[13] ***open*UTM** (BS2000/OSD)
Generating and Handling Applications
User Guide

Target group
This manual is intended for application planners, technical programmers, administrators and users of UTM applications.
Contents
The manual describes the generation of UTM applications with distributed processing, the tools available with *open*UTM for this purpose, and the UTM objects created in the course of generation. It also contains all the information necessary for structuring, operating and monitoring a productive UTM application.

[14] **UTM** (TRANSDATA)
Programming Applications
User's Guide

Target group
Programmers of UTM applications
Contents
- Language-independent description of the KDCS program interface
- Structure of UTM programs
- KDCS calls
- Testing UTM applications
- All the information required by programmers of UTM applications

Applications
BS2000 transaction processing

[15] **TIAM** (TRANSDATA, BS2000)
User Guide

Target group
- BS2000 users (non-privileged)
- Programmers

Contents
- All TIAM commands and macros
- The TIAM COBOL interface with the TIAM COBOL macros
- Examples

Applications
BS2000 timesharing mode

[16] **DCAM** (TRANSDATA)
Program Interfaces
Reference Manual

Target group
- Managers
- Application planners
- Programmers
- System and network administrators

Contents
Description of the Data Communication Access Method DCAM

Index

Contents

WIN-DOORS V3.1 (BS2000/OSD, MS-Windows)

Graphical interface for BS2000/OSD applications

User Guide

Target group
This manual adresses BS2000 developers who wish to equip BS2000 applications with a graphical interface.
Contents

The manual describes the usage model and the functions of WIN-DOORS. A sample session provides an example of how to work with FHS-DOORS. The manual also describes the parameters with which sessions can be configured and as required by applications, and the interfaces for a library for format recognition and OLE automation.

Edition: March 1997

File: WIN_DOR.PDF