

HP DECforms

Style Guide for Character-Cell Devices

Order Number: AA-Q505C-TE

January 2006

HP DECforms is a software product for applications, services, and tools that require a structured, forms-based, or menu-based user interface. DECforms is the first commercial implementation of a proposed ANSI/ISO standard for forms-based interfaces, the CODASYL Form Interface Management System (FIMS).

Revision/Update Information:	This is a revised manual.
Operating System:	OpenVMS Alpha Version 7.3-2 or later OpenVMS I64 Version 8.2 or later
Software Version:	HP DECforms Version 4.0

Hewlett-Packard Company
Palo Alto, California

© Copyright 2006 Hewlett-Packard Development Company, L.P.

Confidential computer software. Valid license from HP required for possession, use, or copying. Consistent with FAR 12.211 and 12.212, Commercial Computer Software, Computer Software Documentation, and Technical Data for Commercial Items are licensed to the U.S. Government under vendors standard commercial license.

The information contained herein is subject to change without notice. The only warranties for HP products and services are set forth in the express warranty statements accompanying such products and services. Nothing herein should be construed as constituting an additional warranty. HP shall not be liable for technical or editorial errors or omissions contained herein.

Intel and Itanium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

Oracle CDD, Oracle/Administrator, Oracle CDD/Repository, Oracle Rdb, and Oracle TRACE are trademarks of Oracle Corporation.

Motif is a registered trademark of the Open Software Foundation, Inc.

ISO is a trademark of the International Organization for Standardization.

PostScript is a registered trademark of Adobe Systems, Incorporated.

Printed in the US

This document was prepared using VAX DOCUMENT Version 2.1.

Contents

Preface	ix
----------------------	----

Part I Elements of DECforms Style

1 User Interface Design Principles

1.1	Benefits of Using a Style Guide	1-1
1.2	Benefits of Using the DECforms Style Guide	1-2
1.3	Why This Style Guide Was Developed	1-2
1.4	Modifying and Extending the DECforms Style Guide	1-2
1.5	Elements of Good User-Interface Design	1-3
1.5.1	Keep the User in Mind	1-3
1.5.2	Let the User Be in Control	1-3
1.5.3	Provide Direct Manipulation	1-4
1.5.4	Keep Your Interface Natural	1-5
1.5.5	Provide Consistency	1-5
1.5.6	Communicate Application Actions to the User	1-6
1.5.7	Avoid Common Design Pitfalls	1-7

2 Overview of Screen Design

2.1	Main Screen	2-1
2.2	Menus	2-4
2.3	Dialog Boxes	2-5
2.3.1	When to Use Dialog Boxes	2-5
2.4	Choosing Controls or Menus for Application Tasks	2-6
2.4.1	Choosing a Single-Choice Control	2-6
2.4.2	Choosing a Multiple-Choice Control	2-6
2.4.3	Guidelines for Using a Pop-Up Menu, a Control Panel, or a Pull-Down Menu	2-7
2.5	Labeling Screen Objects	2-7
2.5.1	General Guidelines	2-7

2.5.2	Menus	2-8
2.5.3	Dialog Boxes	2-8
2.5.4	Screen Objects in Dialog Boxes	2-9
2.5.5	Push Buttons	2-9
2.6	Designing Screen Navigation	2-9

3 Controls

3.1	Push Buttons	3-1
3.1.1	Appearance	3-1
3.1.2	Label	3-3
3.1.3	Selection	3-3
3.2	Radio Fields	3-3
3.2.1	Appearance	3-4
3.2.2	Label	3-4
3.2.3	Selection	3-4
3.3	Check Fields	3-5
3.3.1	Appearance	3-5
3.3.2	Label	3-5
3.3.3	Selection	3-5
3.4	Text Entry Fields	3-6
3.4.1	Appearance	3-6
3.4.2	Label	3-6
3.4.3	Entering Text	3-6
3.4.4	Linking a Text Entry Field to a Dialog Box	3-7
3.5	List Groups	3-8
3.5.1	Appearance	3-8
3.5.2	Selection	3-9
3.5.3	Navigation Within a List Group	3-10
3.6	Option Fields	3-12
3.6.1	Appearance	3-12
3.6.2	Label	3-12
3.6.3	Selection	3-13

4 Menus

4.1	Appearance	4-1
4.2	Components	4-3
4.2.1	Menu Items	4-3
4.2.2	Keyboard Accelerators	4-3
4.2.3	Mnemonics	4-4
4.2.4	Separators	4-4
4.3	Choosing a Menu Item	4-4

4.4	Showing Unavailable Menu Items	4-5
4.5	Dismissing Menus	4-6
4.6	Menu Types	4-6
4.6.1	Bar Menus	4-6
4.6.1.1	Standard Bar Menu Items	4-7
4.6.2	Pull-Down Menus	4-8
4.6.3	Pop-Up Menus	4-10
4.7	Designing Menus	4-11
4.7.1	Naming Menu Items	4-11
4.7.2	Grouping Menu Items	4-12

5 Dialog Boxes

5.1	Purpose	5-1
5.2	Appearance	5-1
5.3	Size and Placement	5-2
5.4	Ending a Dialog	5-3
5.5	Chaining Dialog Boxes	5-3
5.6	Grouping Controls	5-3
5.6.1	Arranging Push Buttons	5-3
5.6.2	Default Push Buttons	5-4
5.6.3	Using Radio Boxes	5-5
5.6.4	Arranging Text Entry Fields	5-5
5.7	Specialized Dialog Boxes	5-6
5.7.1	Work in Progress Dialog Box	5-6
5.7.2	Informational Dialog Box	5-7
5.7.3	Question Dialog Box	5-8
5.7.4	File Selection Dialog Box	5-9

Part II Implementing DECforms Style

6 Implementing Controls

6.1	Push Buttons	6-1
6.2	Radio Fields	6-3
6.3	Check Fields	6-8
6.4	Text Entry Fields	6-11
6.5	List Groups	6-13
6.6	Option Fields	6-13

7 Implementing Menus

7.1	Bar Menu	7-1
7.2	Pull-Down Menus	7-5
7.3	Pop-Up Menus	7-12

8 Implementing Dialog Boxes

8.1	Work in Progress Dialog Box	8-1
8.2	Informational Dialog Box	8-2
8.3	File Selection Dialog Box	8-7

A Track and Field Registration Form

B Track and Field Registration Application

Glossary

Index

Examples

6-1	OK and Cancel Push Buttons IFDL Code Example	6-2
6-2	Radio Fields IFDL Code Example	6-4
6-3	Check Fields IFDL Code Example	6-8
6-4	Text Entry Field IFDL Code Example	6-11
6-5	Option Fields IFDL Code Example	6-13
7-1	TAFR Bar Menu IFDL Code Example	7-2
7-2	List Registrants Pull-Down Menu IFDL Code	7-6
7-3	TAFR Country Pop-Up Menu IFDL Code Example	7-13
8-1	Work in Progress Dialog Box IFDL Code Example	8-1
8-2	Quit Caution Dialog Box IFDL Code Example	8-3
8-3	File Selection Dialog Box IFDL Code Example	8-7

Figures

2-1	Sample Main Screen Showing the Different Screen Areas	2-2
2-2	Bar Menu	2-3
2-3	Bar Menu with Pull-Down Menu	2-4
2-4	Sample Dialog Box	2-5
3-1	Recommended Appearance of Push Buttons	3-2
3-2	Default Push Button	3-2
3-3	Alternate Appearance for Push Buttons	3-3
3-4	Set of Radio Fields	3-4
3-5	Highlighted Radio Field Within a Set of Radio Fields	3-4
3-6	Set of Check Fields	3-5
3-7	Text Entry Field	3-6
3-8	Text Entry Field Linked to a Dialog Box	3-8
3-9	List Group	3-8
3-10	Two-Column List Group	3-9
3-11	Option Field	3-12
3-12	Highlighted Option Field	3-12
3-13	Option Field with an Undefined Initial Value	3-12
3-14	Pop-Up Menu Used with an Option Field	3-13
4-1	Sample Menu	4-2
4-2	Sample Menu with All Three Types of Menu Items	4-2
4-3	Sample Menu Showing Accelerator	4-4
4-4	Sample Menu Showing a Highlighted Toggle Item	4-5
4-5	Sample Menu Showing an Unavailable Item	4-6
4-6	Bar Menu with Pull-Down Menus	4-7
4-7	Two-Line Bar Menu	4-7
4-8	Location of Pull-Down Menus	4-9
4-9	Sample of a Submenu Cascaded Downward	4-10
5-1	Sample Dialog Box	5-2
5-2	Push Buttons Arranged Horizontally	5-4
5-3	Push Buttons Arranged Vertically	5-4
5-4	Vertically Stacked Radio Box	5-5
5-5	Horizontally Arranged Radio Box	5-5
5-6	First Method of Stacking Text Entry Fields	5-6
5-7	Second Method of Stacking Text Entry Fields	5-6

5-8	Typical Work in Progress Dialog Box	5-7
5-9	Typical Informational Dialog Box	5-8
5-10	Typical Question Dialog Box	5-9
5-11	Typical File Selection Dialog Box	5-10

Tables

1	Conventions Used in This Manual	xi
2-1	Keys Used to Navigate a DECforms Screen	2-10
3-1	Keys Used to Edit Text	3-7
3-2	Keys Used Within a List Group	3-10
5-1	Additional Keys Used Within a File Selection Dialog Box ...	5-11

Preface

Who Should Use This Guide

This manual is intended for programmers who develop DECforms applications and who seek to present a uniform, usable software interface that is consistent with other DECforms applications.

It is expected that readers are familiar with DECforms software.

Structure of This Guide

The DECforms Style Guide is organized as follows:

Part I	Describes a user interface style developed for DECforms.
Chapter 1	Describes the principles involved in the design of any good user interface.
Chapter 2	Describes the basic screen components and indicates when it is appropriate to use each of these.
Chapter 3	Describes the basic controls used in this interface.
Chapter 4	Describes the structure and contents of menus.
Chapter 5	Describes the use of controls in dialog boxes.
Part II	Describes how to use DECforms to implement the style described in Part I. This description mostly takes the form of examples.
Chapter 6	Shows DECforms examples of different types of controls.
Chapter 7	Shows DECforms examples of different types of menus.
Chapter 8	Shows DECforms examples of dialog boxes.
Appendix A	Contains the IFDL that creates the Track and Field Registration form.
Appendix B	Contains the DECforms Track and Field Registration program.

For More Information

See the online help, the online release notes, or the following documents for more information about DECforms:

- *HP DECforms Installation Guide for OpenVMS Systems*—Describes how to install DECforms software on Alpha and I64 systems that are running the OpenVMS operating system.
- *HP DECforms Guide to Commands and Utilities*—Describes the DECforms FORMS commands and utilities.
- *HP DECforms IFDL Reference Manual*—Describes the DECforms syntax information of the Independent Form Description Language (IFDL).
- *HP DECforms Programmer's Reference Manual*—Describes how DECforms software operates at run time and how to call the DECforms requests from an application program.
- *HP DECforms Guide to Developing an Application*—Part I explains for the beginning DECforms programmer how to create a DECforms application, including both the form and the program. Part II contains additional guidelines and examples for more experienced DECforms programmers.
- *HP DECforms Guide to Demonstration Forms and Applications*—Describes how to use various demonstration forms and applications. This guide is contained in online files named `forms$demo_guide.txt` and `forms$demo_guide.ps` in the `FORMS$EXAMPLES` directory.

If you cannot find this document, ask your system manager to install it in the appropriate directory.

For information about displaying these forms, see the appendix section of the *HP DECforms Guide to Developing an Application*.

- *HP DECforms Guide to Converting FMS Applications*—Describes how to convert a VAX FMS or DEC FMS application to a DECforms application.

Also of interest to users of DECforms software is the *CODASYL Form Interface Management System Journal of Development* (see the Acknowledgment section).

Reader's Comments

HP welcomes your comments on this manual or any of the DECforms. Please send comments to either of the following addresses:

Internet	openvmsdoc@hp.com
Mail	Hewlett-Packard Company OSSG Documentation Group, ZKO3-4/U08 110 Spit Brook Rd. Nashua, NH 03062-2698

How To Order Additional Documentation

For information about how to order additional documentation and for online versions of most DECforms, visit the following World Wide Web address:

<http://www.hp.com/go/openvms/doc/>

Conventions

Table 1 shows the conventions used in this manual:

Table 1 Conventions Used in This Manual

Symbol or Term	Meaning
Ctrl/X	In procedures, a sequence such as Ctrl/X indicates that you must hold down the key labeled Ctrl while you press another key.
KPn	Key names that begin with KP indicate keys on the numeric keyboard on the right side of your keyboard. For example, KP4 and KP. are keys on the numeric keypad.
PF1-X	A sequence such as PF1-X indicates that you must first press and release the key labeled PF1, then press and release another key.

(continued on next page)

Table 1 (Cont.) Conventions Used in This Manual

Symbol or Term	Meaning
...	In examples, a horizontal ellipsis indicates one of the following possibilities: <ul style="list-style-type: none"> • Additional optional arguments in a statement have been omitted. • The preceding item or items can be repeated one or more times. • Additional parameters, values, or other information can be entered.
. . .	A vertical ellipsis indicates the omission of items from a code example or command format; the items are omitted because they are not important to the topic being discussed.
()	In format descriptions, parentheses indicate that, if you choose more than one option, you must enclose the choices in parentheses.
\$ user input	Bold in examples shows user input.
Bold	Boldface text indicates a new term in text that is also defined in the Glossary.
\$	The default user prompt is your system name followed by a right angle bracket (>). The dollar sign is used to indicate the DCL prompt on VMS systems. This prompt might be different on your system.
DECforms	References to DECforms throughout this manual refer to HP DECforms software.

Acknowledgment

DECforms is the HP implementation of a Form Interface Management System (FIMS) ANSI/ISO standard prepared by the CODASYL Form Interface Management System Committee and ISO/IEC JTC1/SC22 Working Group 18. The FIMS standard is documented in *ISO IS 11730:1994* and can be purchased from the International Organization for Standardization or the American National Standards Institute.

Part I

Elements of DECforms Style

This part contains a basic overview of good user interface design, and a style guide for DECforms software.

1

User Interface Design Principles

DECforms software is a powerful and flexible tool used to create and control the user interface of an application. DECforms currently supports only character-cell terminals. This guide is intended to assist users of character-cell terminals in creating screens.

The great flexibility of DECforms lets you create user interfaces that look and act differently, depending on the needs of your applications. For example, you would design the user interface of an accounts payable application differently from an application that teaches children simple arithmetic.

Although there are instances in which you want an application to have a totally unique user interface, you probably are designing applications that are part of a group of related applications or are used by people who use other applications. In this instance, there should be a consistent look and feel among all the applications. The screens in all the applications should look similar, and the way the user moves through the applications should feel similar. This consistency across applications makes it easier for a user to learn a new application. The knowledge learned from one application can be applied to another application, reducing the amount of learning and subsequent recall. When an application works with the same look and feel as other applications, the new function seems familiar, comfortable, and appropriate to users.

1.1 Benefits of Using a Style Guide

A style guide is a set of rules and guidelines that helps you make design decisions during your application development.

The advantages of following an established style during development include the following:

- Consistency within each application
- Consistency among all applications that you develop

- Consistency among all applications designed by different application developers
- Faster application development, because many of the design decisions have already been made

1.2 Benefits of Using the DECforms Style Guide

Although following any style guide provides consistency, it is important that the style you choose provides a well-designed user interface that is easy to use. The style presented in this guide provides this ease of use.

This style guide is:

- Based on good design principles
- A collaborative design effort involving DECforms engineers, forms developers, users, and user-interface designers
- Designed with maximum use of DECforms features and attention to performance issues

1.3 Why This Style Guide Was Developed

Many DECforms customers have asked for guidelines to create usable and consistent user interfaces for DECforms applications. This style guide was developed in response to those requests.

This style was chosen as many people use DECforms applications on character-cell terminals.

1.4 Modifying and Extending the DECforms Style Guide

Although many DECforms developers will find the style documented in this guide applicable to the applications they are developing, you may determine that it is not totally appropriate for your application or is not detailed enough for your application. In this case, you should use this guide as a starting point and modify or extend it to fit your requirements.

Consider the following guidelines in designing the modifications and extensions to your style guide:

- Start by talking to your users. Ask them questions that will help you understand their requirements for a user interface.

- Organize a small group of people who can help you develop a style guide. The contributors to this group will depend on your company. At a minimum, you should include some developers and some users. If possible, include others who might have valuable input (for example, from documentation, customer service, quality control, and so on).
- Consider using the services of a user-interface designer.
- Meet with your group to discuss major topics of style (for example, menus, controls, help, navigation, and so on).
- Become aware of any existing standards that your style must follow.
- Create samples or prototypes of your proposed style. Show these prototypes to users and gather their feedback.
- Document your style clearly. Use pictures and samples of your style whenever possible.
- Create a mechanism by which suggestions and comments on the documented style can be collected, discussed, and implemented (or rejected).
- Read and keep in mind the elements of good interface design described in Section 1.5.

1.5 Elements of Good User-Interface Design

A good user interface is one that is consistent and easy to use. This section includes guidelines for designing a usable interface that are applicable to any style.

1.5.1 Keep the User in Mind

Good interface design involves adopting the user's point of view. Your users are your audience. For your interface to communicate well with users, you must understand their vocabulary, the work they need to perform, and how they do their work. The best way to learn this is to talk with your users, observe them, involve them in the design, and become a user yourself. The user's tasks should drive the interface design.

1.5.2 Let the User Be in Control

Users want to be in control of their tools.

You can provide them with this control as follows:

- Give the user a choice*
- Provide flexibility by providing multiple ways of accessing an application's functions. This increases the user's sense of control. Allow your users to choose the way they want to accomplish a task. For example, a user can access a function through a pull-down menu, a mnemonic key sequence, or an accelerator.

You can also provide flexibility by allowing users to change default settings and customize options. To be effective, the interface between the user and the customization feature must be easily accessible.

- Put first things first*
- Design your application using progressive disclosure. This means that the necessary and common functions are presented first and in a logical order. The more sophisticated and less frequently used functions should be hidden from view, but easily available. For example, use a dialog box to hide settings that are used infrequently.

Decisions about the placement of functions are not easy to make. From the implementation standpoint, all functions are important. Often, however, a relatively small number of functions account for the majority of use. These important functions need to be prominently featured in the presentation of the interface, but they can be prominent only if other functions are hidden.

1.5.3 Provide Direct Manipulation

Direct manipulation describes the relationship between the user and the user's tools. Direct manipulation connects the user's actions with the response of physical objects. In direct manipulation interfaces, users should experience immediate and visible results of their actions.

You can provide direct manipulation as follows:

- Make it fast*
- Make your application respond to input as rapidly as possible. When using controls, the user should experience the application's response as immediate and in proportion to the user's action. The response speed of the application must also be consistent. Delays, disproportionate responses, or inconsistent responses can make an otherwise well-designed application unusable. Performance problems make it difficult for the user to concentrate on the task at hand.
- Make it selectable*
- Make the output of your application available as input. For example, if one action produces a list of names, another action can select from this list of names. The user manipulates objects by highlighting them and selecting them rather than typing in their names. The only time the user should need to type a name is when objects are created. An application should reduce the amount of information the user must memorize to perform work.

1.5.4 Keep Your Interface Natural

Design your application so that tasks flow naturally. Users need to be able to anticipate the natural progression of each task.

Each screen object needs to have a distinct appearance that your user can easily recognize and quickly understand. At the same time, the style of the interface needs to unify these elements graphically and ensure a consistent and attractive appearance.

You can ensure task flow as follows:

Make navigation easy

- Make navigation easy by providing an easy mechanism for moving through the work area of your application. Moving easily and quickly within the work area gives users a sense of mastery over the application and their work. For example, pressing the down arrow key is an easy and intuitive way to move downward from one object to the next.

Arrange elements on the screen according to their use; optimal arrangement assists the user's decision-making process and reduces the possibility of errors. Present screen objects in an orderly, simple, and uncluttered manner.

Use color carefully

- Use color carefully. Color is a powerful cue for differentiating screen objects; however, you should use it conservatively. If there are many objects with strong contrast or bright colors on the screen, the user will have difficulty knowing where to look first because all these objects compete equally for attention. Be sure that any use of color coding supports the user's task.

Use color only as a secondary cue; that is, to provide additional differentiation among screen objects. The usability of an application should not rely on the user's ability to distinguish colors.

Be aware of any cultural expectations of certain colors. For example, in Western cultures the color red is often associated with a warning, error, or emergency situation.

1.5.5 Provide Consistency

The main purpose of any style guide is to ensure ease of use. Consistency is an excellent tool to help provide this. Consistency is important both across applications and within a single application. Consistency helps the user transfer familiar skills to new situations. When components work in a manner that is consistent with other components, the user will be less afraid to try new functions.

Consistency means the following:

- Group by similarity*

 - Similar components operate similarly and have similar uses. For example, because pull-down and pop-up menus are similar components, their operation and use should be similar.
- Same action and result*

 - The same action should always have the same result. For example, pressing the down arrow key in a list group should always move the highlight to the next item in the list.
- Keep the same function*

 - The function of components should not change based on context. For example, pressing the Select key at a push button should always perform the same action. Even though the action is the same (initiate the command), the result of the action can depend on context (the particular command).
- Keep the same position*

 - The position of components should not change based on context. Adding and removing components as needed makes it difficult to find the desired component quickly. Make unneeded components unavailable and indicate this by differentiating their labels. For example, help is usually the last selection on a menu. Do not move help to a higher level because you feel that your user might select it more often. The user looks for components to remain in similar positions in different contexts.

1.5.6 Communicate Application Actions to the User

Well-designed applications let the users know what is happening. This feedback increases user satisfaction.

You can communicate with users as follows:

- Give feedback*

 - Give users feedback whenever they have initiated an action. Feedback lets users know that the computer has received their input. If certain operations take more than a few seconds, provide a message to let the user know that the computer is working on that operation.
- Give help*

 - Anticipate likely errors. By anticipating errors, you can avoid them in your design, enable the support of recovery attempts, and provide messages informing the user of the proper corrective action.

Use context-sensitive help to aid understanding, reduce errors, and ease recovery efforts. Help information text needs to be clear, concise, and written in everyday language. Help information needs to be readily accessible and just as readily removable.

Give warnings

- Use explicit destruction. Explicit destruction means that when an action has irreversible negative consequences, it should require the user to take an explicit action to perform it. For example, whereas a file can be saved simply by selecting a Save push button, deleting the file should require selecting a Delete push button and answering a warning question, such as “Are you sure you want to delete this file?” by choosing a selection in the warning dialog box.

Warnings protect the user from inadvertent destructive operations, yet allow them to remain in control of the application. Warnings also encourage the user to experiment without fear of loss. Operations that can cause a serious loss of data should warn the user of the consequences and request explicit confirmation.

1.5.7 Avoid Common Design Pitfalls

The process of achieving good design presents many challenges and potential pitfalls.

The following guidelines can help you avoid common pitfalls:

Watch the details

- Pay attention to details.
The details of an application express the sense of craft that you apply to your application. The details of an elegantly designed interface should both please users and make their work easier. For example, consistent capitalization of menu items and dialog box labels is a design detail that reduces textural distraction for users.

Refine the design

- Redesign as long as possible.
A common design pitfall is assuming too early that a design is complete. This tendency is aggravated by schedule pressures and difficulty in pinpointing the inadequacies of a design. Although it is important to begin designing early, it is also important to allow for redesigning for as long as possible. The first design of an application is not a solution but a fresh perspective from which to view interface design problems.

Refine the design again

- Design iteratively.
The development cycle of implementation, feedback, evaluation, and change avoids costly errors by allowing for early recognition and correction of unproductive designs.

*Start
fresh*

- Start with a fresh perspective.

Avoid the temptation to convert existing software by simply translating it to a new style of interface. Because direct manipulation changes the way users work, a simple one-to-one translation is unlikely to be successful. Command-line applications that are converted to direct manipulation should be extensively reconsidered and revised. The functions, structure of the function hierarchy, and presentation need to be completely redesigned.

*Make it
simple*

- Hide implementation details.

User interfaces should hide the underlying software and present a consistent interface to the user. A good user interface does not allow implementation details of the application to show through. It frees the user from focusing on the mechanics of an application. An excellent example of this is using a bank machine to get money. You do not see the underlying software that finds your account, validates the transaction, subtracts the amount requested from your balance, and then returns that information to the bank. All you do is press a few keys to get cash. In other words, the interface design should be driven by the user's task, not by the underlying system.

*Do
what's
best for
the user*

- Recognize conflicting guidelines.

There may be times when two or more design guidelines conflict. Recognize these occurrences and carefully weigh all the factors in designing a solution. For example, assume direct manipulation is an important guideline being followed, but in one area of the application it goes against the user's point of view. Recognize this conflict and examine tradeoffs for various solutions. One of the real challenges in designing a good user interface is to determine what is best for the user when conflicts arise.

2

Overview of Screen Design

One of the main challenges of designing an application interface is to decide how to best present the different pieces of information to users. The DECforms interface described in this style guide has three major methods for presenting information: the main screen, menus, and dialog boxes.

This chapter describes:

- The main screen, menus, and dialog boxes, including the controls used to design menus and dialog boxes. (Menus and dialog boxes are described in more detail in Chapters 4 and 5, respectively.)
- Guidelines for choosing controls or menus for application tasks.
- Guidelines for labeling screen objects.
- Guidelines for designing screen navigation.

2.1 Main Screen

The design of the application's main screen depends very much on the application itself. The work area might contain controls for the primary application actions. It also might contain important information that is constantly being updated, or a control panel for components that are used frequently throughout the application. Many applications use only the bar menu on the main screen and leave the work area open, ready to display menus and dialog boxes.

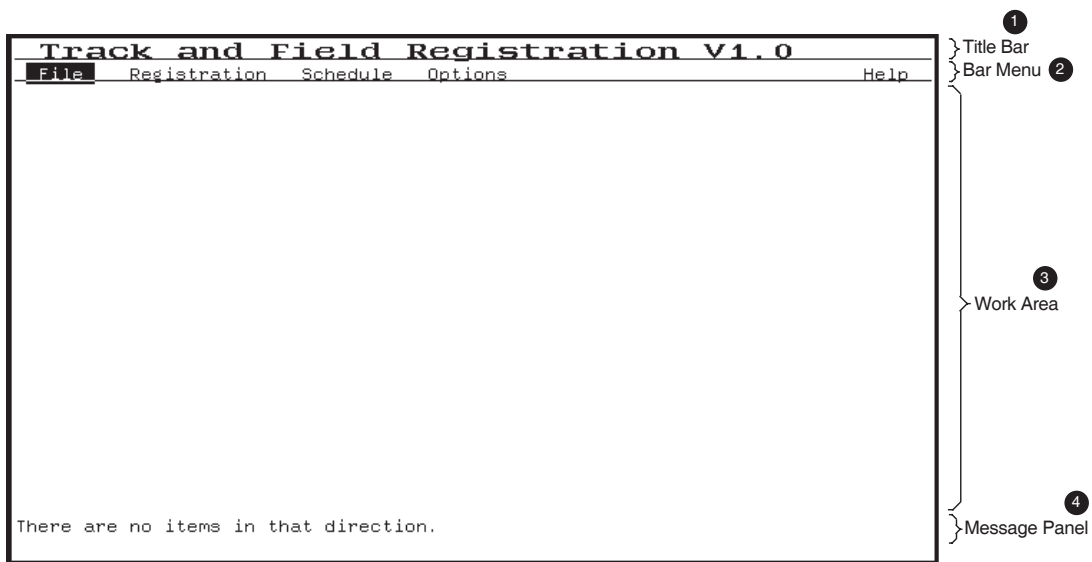
One of the first tasks of your application is to present the main screen of the application to your users. Effective screen design uses different areas of the main screen to deliver different kinds of information. This consistent method of delivery helps users easily find the information they need.

The main screen of your DECforms application should display the following areas on the screen:

- Application title bar
- Bar menu
- Work area
- Message panel

Figure 2-1 shows a main screen with a title bar, a bar menu, a work area, and a message panel.

Figure 2-1 Sample Main Screen Showing the Different Screen Areas



ZK-9810-GE

1 Application Title Bar

The application title bar is the line that identifies your application. It occupies line 1 on the screen. The title is left-justified and the whole line is underlined.

2 Bar Menu

The bar menu is a horizontal bar that contains the names of the menus for each major function of the application. The bar menu is located immediately below the title bar (on line 2) and extends the full width of the screen. If the space required for the menu names exceeds the screen width, the bar menu can occupy two lines (lines 2 and 3). For more detailed information about menus, see Chapter 4.

Figure 2–2 shows a bar menu in detail.

If your application provides interactive help, you should include a Help menu, as shown in Figure 2–2. The Help menu should be located on the right side of the bar menu.

Figure 2–2 Bar Menu



3 Work Area

The work area is the portion of the screen in which users perform most application tasks. It is located directly below the bar menu. The work area of the application main screen can contain permanently displayed application information, such as a control panel. It is also the area in which pull-down menus, pop-up menus, and dialog boxes are displayed.

4 Message Panel

You use the message panel to display short application messages. The recommended size and position is a two-line message panel located at the bottom of the screen (lines 23 and 24). There is only one message panel.

Note

Design your application so that other panels do not overlay the message panel. If you do not, the message panel comes to the surface when a new message is displayed, obscuring the currently active panel. This removes control from users and leads to frustration.

2.2 Menus

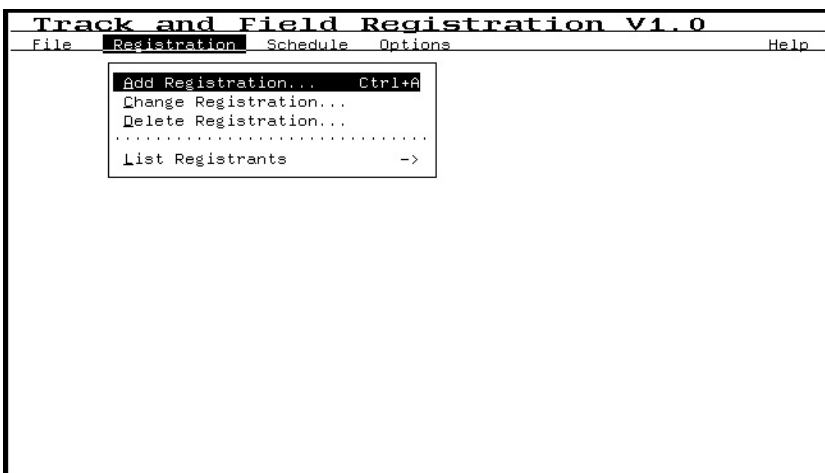
Menus are readily available, quick to be displayed and dismissed, and easy to browse through. However, menus are suitable only for simple actions and cannot be used for text entry or for complicated functions.

Menus should be used in the following cases:

- For application functions that are used frequently
- For application functions that are accessed by most users
- For simple actions

Menus consist of lists of items from which to choose, and the controls used to choose them. The **bar menu** is one type of menu used by DECforms applications. Other types of menus include **pull-down menus** and **pop-up menus**. A pull-down menu appears when users select a menu name from another menu. For example, Figure 2-3 shows a pull-down menu that appears when users select *Registration* from the bar menu. Pop-up menus look similar to pull-down menus and are displayed as a result of a special key sequence. Chapter 4 describes both pull-down and pop-up menus in detail.

Figure 2-3 Bar Menu with Pull-Down Menu



2.3 Dialog Boxes

Dialog boxes are pop-up panels that are used to gather or display information. They can contain controls that help users enter information and control application tasks. Dialog boxes remain visible only until their purpose has been completed. Chapter 5 describes dialog boxes in more detail.

Figure 2–4 shows a sample dialog box displayed in the work area of the main screen.

Figure 2–4 Sample Dialog Box



2.3.1 When to Use Dialog Boxes

Dialog boxes are primarily used to present information to users and to take input from users. Dialog boxes are displayed only when needed, thus leaving the main work area less cluttered. Dialog boxes can be used to perform more than one action at a time, and for more complex interactions than are available in a menu, such as file selection.

Dialog boxes should be used in the following cases:

- For ancillary application actions
- For seldomly used actions
- For application functions that are not accessed by most users

- For complex actions
- For transient information
- For messages that require some action

2.4 Choosing Controls or Menus for Application Tasks

Once you have determined which presentation methods to use for the major components of your application, you need to determine how best to present the tasks within each of the major components. Chapter 3 describes in detail the different types of controls that can be used to accomplish tasks. This section presents guidelines for which controls or menus should be selected to accomplish different kinds of tasks.

2.4.1 Choosing a Single-Choice Control

Use a single-choice control when you want users to choose a single item from a group of items. For example, you might want users to choose an address label format in a customer database program.

The single-choice controls are as follows:

- Radio field
- List group
- Option field

The choice of which single-choice control to use depends on the number of items in the list and the space available to display the control. A radio box typically contains from 2 to 10 **radio fields**. A list of items more than 10 is better displayed in a **list group**. If there is little space available in your application, use an **option field**, as it takes up the least room. When selected, the option field should display a pop-up menu (if there is room to display the whole list of items) or a list group (if there is only room for part of the list to be displayed at one time).

2.4.2 Choosing a Multiple-Choice Control

Use a multiple-choice control when you want users to be able to choose more than one item from a group of items. For example, you might want users to select any number of athletic events from a list of athletic competitions.

The multiple-choice controls are as follows:

- Check field
- List group

The major factor in determining which control to use depends on the number of items in the list. For small groups of 10 or fewer items, use a group of **check fields**. For larger numbers of items, use a list group. Depending on how it is set up, a list group can be a single- or multiple-choice control.

2.4.3 Guidelines for Using a Pop-Up Menu, a Control Panel, or a Pull-Down Menu

Users select both pop-up menus and control panels as shortcuts to access frequently used controls. Pop-up menus are visible only when requested by users. Whereas pop-up menus are hidden, they can be difficult for novice users to find. However, they do not take up any permanent space in the work area.

A control panel is always visible and is thus easier than a pop-up menu for novice users. However, a control panel takes up permanent space in the work area.

An application should use a control panel when users make frequent or multiple selections, and when there is adequate space in the work area. An application should use a pop-up menu only when there is not enough space in the work area to provide a permanent control panel. Pull-down menus provide a good compromise between the availability of a control panel and the space savings of a pop-up menu.

2.5 Labeling Screen Objects

To ensure the consistency of your application with other DECforms applications, use the following guidelines for labeling controls, menus, and dialog boxes.

2.5.1 General Guidelines

The following guidelines apply equally well to controls, menus, and dialog boxes. You should use them as frequently as possible to ensure consistency.

- Avoid using all uppercase text. Mixed case improves readability and style.

Use: Add Registration

Avoid: ADD REGISTRATION

- Use initial caps for menu names, menu items, and labels. If you use a compound word, capitalize the first word and all other words that are nouns or proper adjectives or that have equal force with the first word. Use all lowercase letters for any article, coordinating conjunction, or preposition that falls between two major words.

Side-by-Side
Cross-References

- Follow the capitalization rules of the foreign language when you use foreign words.
- Avoid abbreviations. Pop-up and pull-down menus take up space on the screen only temporarily, so spelling out a word costs little. Abbreviations can be ambiguous and can cause translation problems.
Follow international standards for abbreviations of units of measurement.
- Use an acronym only if the term has previously been spelled out on the screen with the acronym following in parentheses.

2.5.2 Menus

Use the following guidelines to label menus:

- Capitalize each word in a menu name or menu item with three or fewer words, except for articles and prepositions fewer than five characters in length that fall between two other words.

Print List
Print to File...

- If a menu name or menu item contains four or more words, capitalize only the first word.

Use row and column numbers
Print only right-hand pages
Place page number at bottom

2.5.3 Dialog Boxes

Use the following guidelines to label dialog boxes:

- Where possible, use sentences or sentence-like wording for informational, question, or work in progress boxes. Capitalize the first word of each sentence. Use punctuation unless a file specification occurs at the end of a sentence.

```
Loading file [TRACK.REGISTRANTS]AUSTRALIA_REGISTRANTS.TXT
The file [USER.REPORTS]MY_REPORT.TXT cannot be found.
Check the file name and directory for errors and try again.
```

- The title of the dialog box should reflect the command that created it. Left-justify the title and capitalize all words except for articles and prepositions fewer than five characters in length that fall between two other words.

```
List Registrants by Country
```

2.5.4 Screen Objects in Dialog Boxes

Use the following guidelines to label objects in dialog boxes:

- Capitalize each word of a screen object within a dialog box (such as labels for a check field, radio field, option field, or text-entry field), except for articles and prepositions fewer than five characters in length that fall between two other words.

```
Display on Screen
Print List
Print to File
```

- If the text for a screen object within a dialog box contains four or more words, capitalize only the first word.

```
Delete file when printed
```

2.5.5 Push Buttons

Use the following guidelines to label push buttons:

- Capitalize the first letter of each word.

```
Cancel
Delete File
```

- The only exception to this rule is the OK push button, in which case both letters of the word are capitalized.

2.6 Designing Screen Navigation













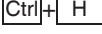

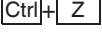



Part of designing a screen is designing how users will move the cursor from object to object. Consistency in navigation is very important for a well-designed screen.

Help users by providing other feedback when the cursor does not move or when they press a function key with no defined action. If users press a navigational key when no movement can occur, an informational message should appear in the message panel. For example, if the cursor is located on the topmost object

and users press the up arrow key, you could display the message “Press F10 to complete, or down arrow to go to next item.”

Table 2–1 shows the keystrokes used to navigate a DECforms screen.

Table 2–1 Keys Used to Navigate a DECforms Screen

Action	LK-Series Keyboard Keys ¹	VT100-Series Keyboard Keys
Moves the cursor to the object to the right		
Moves the cursor to the object to the left		
Moves the cursor to the object above		
Moves the cursor to the object below		
Moves to the next object ²		
Moves to the previous object ²		 or 
Accelerator for the default operation (typically, the OK operation)		
Accelerator for the CANCEL operation		 

¹LK-series keyboards can also use the VT100-series keys, except for the Linefeed and Backspace keys, which are not on an LK-series keyboard.

²The activation order defines the next and previous object. You should design the form so that the next object in the activation order corresponds to the object that the user logically expects to be the next object.

3

Controls

Users use different types of controls to manipulate applications. These controls include:

- Push buttons
- Radio fields
- Check fields
- Text entry fields
- List groups
- Option fields

This chapter describes each of these individual types of controls. Chapter 5 describes how to combine and use these controls within dialog boxes.

3.1 Push Buttons

A **push button** allows a user to initiate an action. The push button appears as a label within a rectangular area. When the user positions the cursor to the push button and selects it, the action represented by the label occurs. Push buttons are used in a variety of ways in menus and dialog boxes. This section discusses the push button as an individual component. Chapter 5 discusses the use and grouping of push buttons in menus and dialog boxes.

3.1.1 Appearance

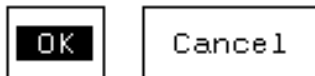
A push button can appear one of two ways, depending on the space requirements of your application. The recommended appearance uses a rectangular box around a label, occupying three vertical lines on the screen. If you do not currently have enough space for this on your screen, consider splitting the information into more than one screen, or using additional dialog boxes. Remember, a tight, cramped screen that is full of information is often difficult to use. However, if your users' requirements necessitate a full screen, then you can use the alternate appearance for a push button.

Whichever style of push button box you choose, use it consistently within your application. Do not mix the two different box styles within an application.

Recommended Appearance—The preferred way to create a push button is to surround the label with a rectangular box, as shown by the Cancel push button in Figure 3–1. Leave one space character before the label and one space character after. (The exception to this rule occurs when push buttons of different label lengths are stacked vertically. In this case, all boxes are made the same size and each label is centered within its box. For more information on stacked push buttons, see Section 5.6.1.)

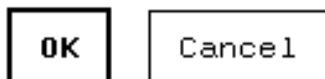
When the user positions the cursor to the push button, the blinking block cursor appears in the space to the left of the label, and the characters of the label appear in reverse video, as illustrated by the OK push button in Figure 3–1.

Figure 3–1 Recommended Appearance of Push Buttons



To distinguish a default push button (described in Section 5.6.2) visually from other push buttons, bold the surrounding box and the label, as shown by the OK push button in Figure 3–2.

Figure 3–2 Default Push Button

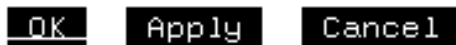


Alternate Appearance—If available space on the screen is limited and you do not have enough room to surround the push button label with a box, you can use reverse video for the label to give a box-like effect. Although using reverse video to create a box requires less space on the screen, a lot of reverse video can distract the user.

Leave one space character before the label and, if space allows, one space character after the label. When the user positions the cursor to the push button, the blinking cursor should appear in the space before the label. To distinguish a default push button visually, underline the label.

Figure 3-3 shows push buttons using this alternate appearance.

Figure 3-3 Alternate Appearance for Push Buttons



3.1.2 Label

Use a short label in a push button. Action verbs such as Apply, Cancel, Reset, and Print are best. Labels such as Yes, No, or OK are commonly used as replies to questions. Do not use abbreviations in your push button label, because this often confuses the user.

If the push button is used to display another menu or dialog box, the label is followed by an ellipsis (. . .). For example, the label Options . . . is used in a push button whose action is to display another dialog box with options.

3.1.3 Selection

After positioning the cursor to a push button, the user presses the Select key or the keypad period key to select the push button. The action described by the label then occurs. If the user must wait for the action and if there are no other visible clues to what is happening, you should either display a message in the message panel or display a “Work In Progress” dialog box to indicate that the action is taking place.

3.2 Radio Fields

A **radio field** allows users to select one item from a group of items that are *mutually exclusive*. For example, when you select one button on a car radio, the previously selected button is no longer selected.

Figure 3–4 illustrates a typical set of radio fields.

Figure 3–4 Set of Radio Fields

```
Print Format:  
<◆> Display on Screen  
< > Print List  
< > Print to File
```

3.2.1 Appearance

When the radio field is on, the radio indicator is a set of angle brackets surrounding a diamond character. When the field is off, a space character replaces the diamond. The indicator is positioned to the left of the label, separated by one space. When the user positions the cursor to the radio field, the blinking cursor appears in this space and the label appears in reverse video, as shown in Figure 3–5.

Figure 3–5 Highlighted Radio Field Within a Set of Radio Fields

```
Print Format:  
<◆> Display on Screen  
< > Print List  
< > Print to File
```

3.2.2 Label

Use a short, descriptive phrase as the label in a radio field. The label of a radio field should clearly indicate the action caused by choosing the field. Capitalize the first and last words of the label, and all nouns, pronouns, adjectives, verbs, and adverbs. Do not capitalize conjunctions, articles, and prepositions.

Do not try to make the label short at the expense of clarity.

3.2.3 Selection

After positioning the cursor to a radio field, the user presses the Select key or the keypad period key to select the radio field. A filled-in diamond appears within the angle brackets of the currently selected field and disappears from the previously selected radio field. If the user selects a radio field that is already on, the radio field remains on.

3.3 Check Fields

A **check field** allows users to select items from a group of choices that are *not mutually exclusive*. A user can toggle a check field on or off independently from the other check fields in its group.

Figure 3–6 illustrates a typical set of check fields.

Figure 3–6 Set of Check Fields

```
Events
[ ] Shot Put
[ ] High Jump
[◆] Javelin
[ ] Pole Vault
[◆] Discus
[ ] Long Jump
```

3.3.1 Appearance

When the check field is on, the check indicator is a set of square brackets surrounding a diamond character. When the field is off, a space character replaces the diamond. The indicator is positioned to the left of the label, separated by one space. When the user positions the cursor to the check field, the blinking cursor appears in this space and the label appears in reverse video.

3.3.2 Label

Use a short, descriptive phrase as the label in a check field. The label of a check field should clearly indicate the choice caused by selecting the field. Capitalize the first and last words of the label, and all nouns, pronouns, adjectives, verbs, and adverbs. Do not capitalize conjunctions, articles, and prepositions.

Do not try to make the label short at the expense of clarity.

3.3.3 Selection

After positioning the cursor to a check field, the user presses the Select key or the keypad period key to select the check field. If the indicator was off, the diamond appears within the square brackets, indicating that it is now on. If the indicator was on, the diamond disappears, leaving empty brackets to show that the indicator is now off.

3.4 Text Entry Fields

A **text entry field** allows the user to enter and manipulate character strings. A text entry field consists of a label and a field for text.

3.4.1 Appearance

The text entry field label appears to the left of the entry field. The label and field are separated by a colon and a blank space. The text entry field is underlined to show the size of the field. The underlining also helps the user visually distinguish between a field in which text can be entered and a field that is display-only.

Figure 3–7 is an example of a text entry field in which data has been entered.

Figure 3–7 Text Entry Field

Registration Number: 123456789

3.4.2 Label

The label of a text entry field should clearly identify what is to be entered in the field. Use initial caps to make the text readable, and avoid ambiguous or unknown abbreviations.

3.4.3 Entering Text

When the user positions the cursor to the text entry field, the blinking cursor appears as the first character of the field. As the user enters text, the characters appear underlined.

Table 3–1 shows the keys used to edit text.

Table 3–1 Keys Used to Edit Text

Action	LK-Series Keyboard Keys ¹	VT100-Series Keyboard Keys
Moves the cursor one character to the right. (The cursor does not move further to the right when positioned at the end of the text entry field.) ²	→	→
Moves the cursor one character to the left. (The cursor does not move further to the left when positioned at the beginning of the line.) ²	←	←
Deletes the character before the cursor, and moves all text to the right of the deleted character one space to the left.	Delete	Delete
Deletes all text in the field and places the cursor at the beginning of the line.	F13	Ctrl+J or Linefeed

¹LK-series keyboards also can use the VT100-series keys, except for the Linefeed and Backspace keys, which are not on an LK-series keyboard.

² This overrides the higher-level action of the arrow keys in navigating DECforms screens.

3.4.4 Linking a Text Entry Field to a Dialog Box

To provide a separate dialog box to help users fill in a field, put an ellipsis (. . .) to the right of the label. This clues users visually that they can either type in data or press the Select key or the keypad period key to get additional help in filling in the field. For example, the dialog box might include a list of possible responses from which users can select one. The user's response fills in the text entry field automatically.

Figure 3–8 shows a text entry field linked to a dialog box.

Figure 3–8 Text Entry Field Linked to a Dialog Box

Customer Code . . . : 

3.5 List Groups

A **list group** is used to display a group of items when there is not enough room on the screen to display the whole list at one time. The scroll area is a window behind which the list can be scrolled. At any one time, the scroll area shows a window-sized sublist of the items in the list.

List groups can be used for display only, for selection purposes, and for text entry.

3.5.1 Appearance

A list group contains a stacked list of items, a list indicator, and a surrounding box. Figure 3–9 shows a sample list group.

Figure 3–9 List Group



ZK-9062A-GE

The list of items is stacked vertically and left-justified. Because the list indicator requires at least three lines, the minimum number of item lines is three. The optimum number of item lines depends on your application.

When the cursor is positioned to an item in the list, the item is displayed in reverse video, with the blinking cursor appearing in the space to the left of the item name.

With DECforms software, you can display more than one field on each item line. You can also display text entry fields in a list group. The text entry fields are underlined to indicate their length.

Figure 3–10 shows a list group with two fields per item line, one of which is a text entry field. In this case, it helps the user to have column headings on the line above the list group box.

Figure 3–10 Two-Column List Group

Sport Event	Location
Bicycling	<u>Field 2</u>
Diving	<u>Pool C</u>
Gymnastics	_____
Soccer	_____
Swimming	<u>Pool A</u>
Track	<u>Field 1</u>

The **list indicator** is a vertical line joining the top and bottom indicator characters. It shows whether there are more items in the underlying list in either direction. The top indicator character is “⌞” when the first item displayed in the list is at the top of the list. The character “:” appears when there are other items above the first item displayed. The bottom indicator character is “⌟” if the last item displayed is at the bottom of the item list and is “:” when there are other items below. The list indicator in Figure 3–10 shows that there are more items on the menu than are currently shown.

The box surrounding the list leaves one blank space to the left of the item list and one blank space to the right of the list indicator.

Any column headings or descriptive text appears on the line directly above the surrounding box.









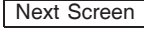
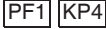
3.5.2 Selection

After positioning the cursor to an item in the list, the user presses the Select key or the keypad period key to select the item. The action initiated by the selection then occurs.

3.5.3 Navigation Within a List Group

Table 3–2 shows the keys used to navigate within a list group.

Table 3–2 Keys Used Within a List Group

Action	LK-Series Keyboard Keys ¹	VT100-Series Keyboard Keys
Moves the highlight up one item. ² If necessary, the window moves up through the underlying list by one line. If the highlight is at the top line, the highlight does not move and a message is displayed: “No more items in that direction.”		
Moves the highlight down one item. ² If necessary, the window moves down through the underlying list by one line. If the highlight is at the last line, the highlight does not move and a message is displayed: “No more items in that direction.”		
Moves the highlight to the next field to the right (within the scroll box). ² If the highlight is at the right-most item, the highlight moves to the left-most item on the next line.		
Moves the highlight to the next field to the left (within the list group box) ² . If the highlight is at the left-most item, the highlight moves to the right-most item on the previous line.		
Moves the window down through the underlying file by one window length minus one line for overlap. The item second from the top is highlighted. If the next screen’s worth of items includes the last item in the list, the window moves down only enough to display the last item on the bottom line of the window.		

¹LK-series keyboards can also use the VT100-series keys, except for the Linefeed and Backspace keys, which are not on an LK-series keyboard.

² This overrides the higher-level action of the arrow keys in navigating DECforms screens.

(continued on next page)

Table 3–2 (Cont.) Keys Used Within a List Group

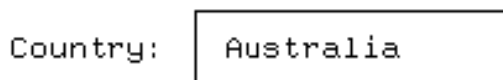
Action	LK-Series Keyboard Keys ¹	VT100-Series Keyboard Keys
Moves the window up through the underlying file by one window length minus one line for overlap. The top item is highlighted.	Prev Screen	PF1 KP5
Moves the window to display the last item in the list as the bottom line of the window. The bottom item is highlighted.	PF1 Next Screen	PF1 B
Moves the window to display the first item in the list as the top line of the window. The top item is highlighted.	PF1 Prev Screen	PF1 T
Moves the highlight to the next field. (This typically moves the highlight down to the next line. However, if there is more than one field in the line, the highlight moves to the next item to the right.)	Return	Return
Selects the highlighted item for action.	Select	KP.
Moves the cursor out of the list group to the object above.	PF1 ↑	PF1 ↑
Moves the cursor out of the list group to the object below.	PF1 ↓	PF1 ↓
Moves the cursor out of the list group to an object to the right. If there is more than one object to the right of the list group, the cursor should go to the most frequently used object (for example, an OK push button).	PF1 →	PF1 →
Moves the cursor out of the list group to an object to the left. If there is more than one object to the left of the list group, the cursor should go to the most frequently used object (for example, an OK push button).	PF1 ←	PF1 ←

¹LK-series keyboards can also use the VT100-series keys, except for the Linefeed and Backspace keys, which are not on an LK-series keyboard.

3.6 Option Fields

An **option field** can have any value selected from an associated list. To save space, only the current value appears in this field, as shown in Figure 3–11. When the user selects the option field, the list of possible values is displayed and the user can select a new value from the list.

Figure 3–11 Option Field



3.6.1 Appearance

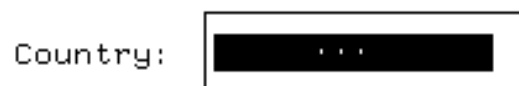
An option field contains a label and a value. The value is surrounded by a box. When the user positions the cursor to an option field, the blinking cursor appears in the space to the left of the value, and the characters of the value appear in reverse video, as shown in Figure 3–12.

Figure 3–12 Highlighted Option Field



If the initial value of the option field is undefined, an ellipsis (. . .) is centered in the field, as shown in Figure 3–13.

Figure 3–13 Option Field with an Undefined Initial Value



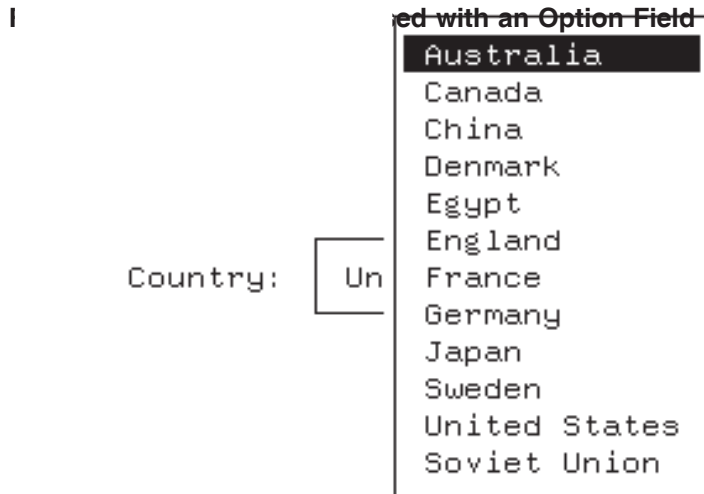
3.6.2 Label

The label of an option field should clearly identify what is to be entered in the field. Use initial caps to make the text readable and avoid ambiguous or unknown abbreviations.

3.6.3 Selection

After positioning the cursor to an option field, the user presses the Select key or the keypad perid key to select the option field. The list of possible values appears in a pop-up menu or in a list group within a dialog box.

A pop-up menu is used when there is enough room in the menu to display all the possible values, as shown in Figure 3-14. This pop-up menu should contain only action items and **cascade items**. **Toggle fields** should not be used.



ZK-9812-GE

A list group can be used when the list of possible values is too large to be displayed all at once. The list group appears in a dialog box. The dialog box should not contain any other controls.

The box containing either the pop-up menu or the list group should be centered vertically over the option field and positioned horizontally so that the first two characters of the current value are visible. This helps to remind the user of the current value.

Once the user has selected an item from the pop-up menu or the list group, the pop-up menu or list group disappears and the selected item appears as the value of the option field.

4

Menus

Menus provide quick access to the functions of your application. A menu consists of a list of items from which to choose.

This chapter discusses menu components, what menus look like, using menus, and the following three types of menus:

- A **bar menu** appears on the application's main panel. It is a horizontal bar that shows the major functional groups of the application. Each item on a bar menu is the title of an associated pull-down menu. The bar menu is the only permanent menu, as it is always visible from the main panel. An application has only one bar menu. Section 4.6.1 describes bar menus in more detail.
- A **pull-down menu** appears when the user selects a menu name from the bar menu or a **cascade item** from another menu. (A cascade item is a menu item that displays a cascade menu.) Section 4.6.2 describes pull-down menus in more detail.
- A **pop-up menu** looks like a pull-down menu, and is displayed as a result of a special key sequence. They are context-sensitive, and are directly associated with an object on the screen, such as a panel, a field, or a dialog box. Section 4.6.3 describes pop-up menus in more detail.

4.1 Appearance

With the exception of the bar menu, all menus contain a vertical list of the menu items within a surrounding box, as shown in Figure 4-1. The menu item labels are left-justified with the first letter of each entry being vertically stacked. If keyboard accelerators or cascade indicators (right arrows, →) are displayed, they are right-justified and positioned far enough to the right so that they do not interfere with the item names, as shown in Figure 4-3. Leave at least two spaces between the end of the longest item name and the first character of the accelerator. Indicate mnemonics by underscoring the corresponding letter.

The surrounding box is a rectangle created by drawing lines directly above and below the text and leaving one blank column between the left and right text margins and the sides of the box.

Figure 4-1 Sample Menu

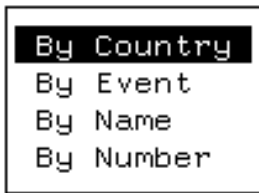
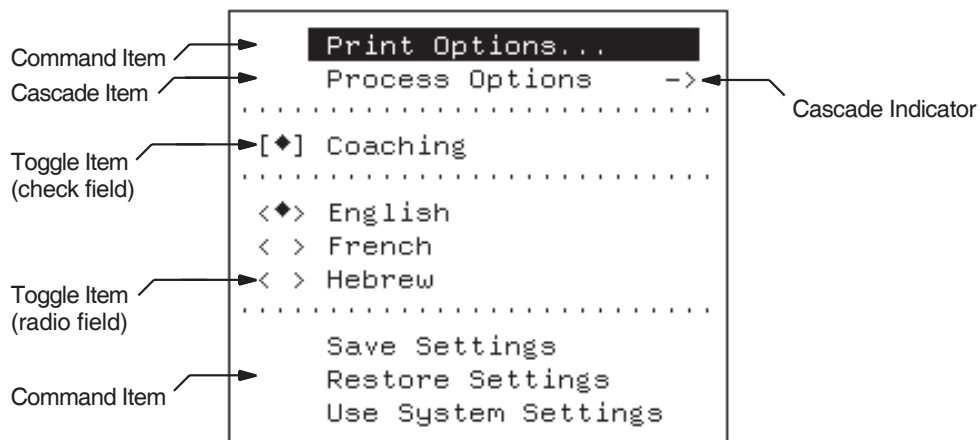


Figure 4-2 shows a menu that includes toggle items. The left margin is extended by four characters for the toggle indicators. Only the toggle indicators should appear in this extended left margin.

Figure 4-2 Sample Menu with All Three Types of Menu Items



ZK-9811-GE

4.2 Components

This section describes each component of a menu. All menus have the following components described in the following sections:

- Menu items
- Keyboard accelerators
- Mnemonics
- Separators

4.2.1 Menu Items

There are three different types of menu items:

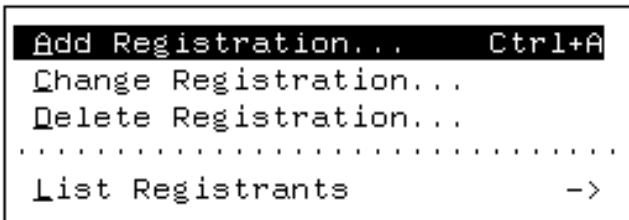
- **Command items** perform some action or command when selected, similar to the use of a push button in a dialog box. Once the user selects a command item, the menu disappears and the command is performed. Sometimes the command is deferred until the user supplies more information. In this case, a dialog box is displayed first. Once the user supplies the necessary information and exits the dialog box, the command is performed. An ellipsis (. . .) is used after the menu item label as a visual clue that a dialog box will be displayed.
- **Toggle items** set a particular state in the application through the use of toggle fields (check fields or radio fields). The filled-in button indicates that the setting is on. Once a toggle item is selected, the menu disappears and the toggle condition is in effect.
- **Cascade items** display a submenu when selected. A cascade indicator (→) at the right of the entry label is used to show that a submenu will be displayed.

The three types of menu items appear in the sample menu in Figure 4–2.

4.2.2 Keyboard Accelerators

Your application can associate a frequently chosen menu item with a key or key sequence. Keys associated with menu items are called **keyboard accelerators**. An accelerator usually consists of a function key or a modifier key combined with a letter (for example, F20 or Ctrl/A). Accelerators are displayed at the right side of the menu item label. For example, as shown by the menu in Figure 4–3, a user could select the Add Registration . . . menu item by pressing Ctrl/A. An accelerator is active from anywhere in the application.

Figure 4-3 Sample Menu Showing Accelerator



4.2.3 Mnemonics

A mnemonic is a single character in a menu item (indicated by an underscore) that is used with a mnemonic introducer key (the PF4 key) as a shortcut for choosing that menu item. For example, at the menu shown in Figure 4-3, a user could select the Delete Registration . . . menu item by pressing PF4-D, instead of using the navigational keys. A mnemonic is only available for the current panel.

Mnemonics are optional; however, in designing your application, you should consider using them for frequently chosen menu items. Use a letter that your users will easily associate with the corresponding function. If a menu item appears on more than one menu, its mnemonic letter should be consistent throughout the application. A mnemonic must be case insensitive.

4.2.4 Separators

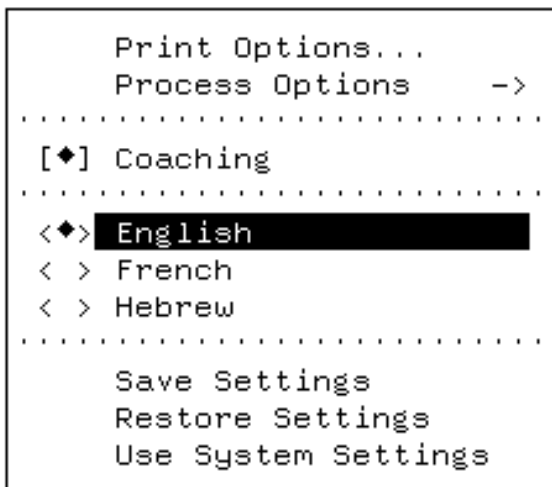
Horizontal separators create logical groupings of menu items. The separator, a single line of dots, extends the full width of the menu, as shown in Figure 4-3.

4.3 Choosing a Menu Item

A user chooses a menu item by highlighting it and pressing the Select key or the keypad period key. The user moves the highlight on the menu by pressing the up arrow and the down arrow keys. (The right arrow and the left arrow keys move the highlight across the bar menu.) When an item is highlighted, the label appears in reverse video with the blinking cursor in the space character to the left of the label (see Figure 4-1).

If there is a toggle indicator in the extended left margin, it is not displayed in reverse video, as shown in Figure 4-4. The reverse video does, however, extend into the extended right margin to include the cascade indicator.

Figure 4-4 Sample Menu Showing a Highlighted Toggle Item



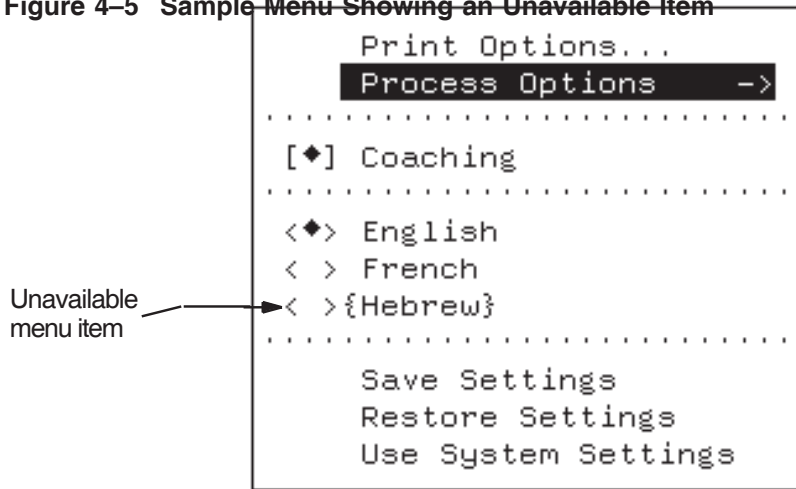
4.4 Showing Unavailable Menu Items

A particular menu item might become inappropriate as the state of your application changes. When this occurs, do not remove the menu item from the menu. Leave the item on the menu to ensure consistency and to remind the user of its existence. However, your application can make the menu item unavailable to the user; that is, disable the menu item. Menu items that are currently unavailable cannot be highlighted, and thus cannot be selected. To de-emphasize an unavailable menu item visually, place it within braces, as shown in Figure 4-5. Enclose any accelerators and cascade indicators within braces, but do not enclose the toggle indicators.

Note

Dimmed text is not available on character-cell terminals, braces were chosen as an alternative.

Figure 4-5 Sample Menu Showing an Unavailable Item



ZK-9809-GE

4.5 Dismissing Menus

A user can dismiss a menu without selecting a menu item by pressing the F8 key or the PF1-Q key sequence. The current menu disappears and the highlight returns to the previous menu location.

4.6 Menu Types

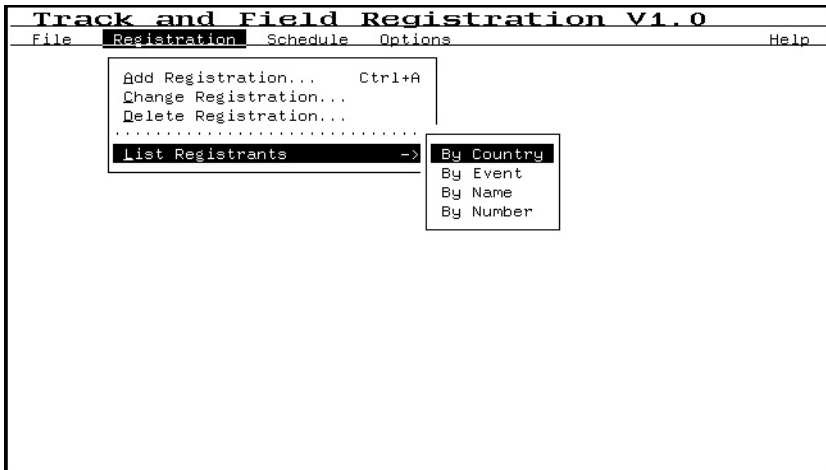
There are three basic types of menus: bar menus, pull-down menus, and pop-up menus. The following sections discuss each type of menu.

4.6.1 Bar Menus

The bar menu is a permanent menu that is always visible near the top of the application's main panel. This menu is a horizontal bar that shows the major functional groups of the application. Each item on a bar menu is a cascade item that, when selected, displays an associated pull-down submenu. Bar menus do not contain command items because they would inhibit menu browsing.

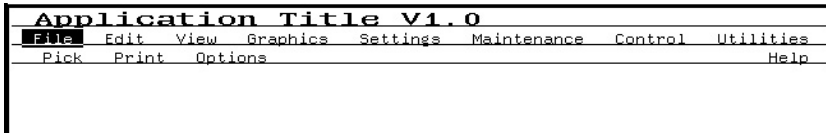
Figure 4-6 shows a bar menu.

Figure 4-6 Bar Menu with Pull-Down Menus



Each entry on a bar menu is separated by at least three spaces. If you cannot fit all entries on one line, create a two-line bar menu, as shown in Figure 4-7. If necessary, add an additional space between entries so that the first characters of two menu items are not vertically stacked, because this is visually confusing. Do not use more than two lines for your bar menu.

Figure 4-7 Two-Line Bar Menu



4.6.1.1 Standard Bar Menu Items

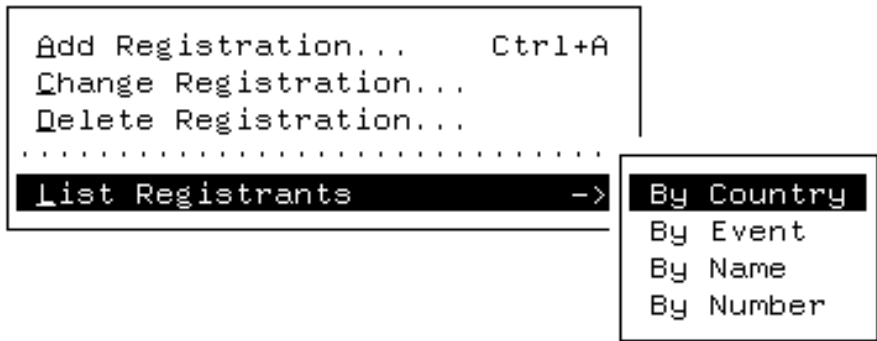
Your choice of bar menu items depends on the nature of your application. The following list of standard bar menu items includes functions that are common to many applications. If your application uses any of these functions, Hewlett-Packard Company recommends that you use the standard terminology and positions specified here.

File	All applications that perform actions on files, such as opening, saving, closing, and printing, should have a pull-down File menu. The File menu provides items to perform these actions as well as any actions on the application as a whole, such as quitting. Standard File menu items include New, Open . . . , Save, Save As . . . , Print, Print . . . , Close, Exit, and Quit. If your application uses a File menu, position it as the first item on the bar menu. If your application uses mnemonics, the File menu mnemonic should be F .
Edit	The Edit menu contains items that allow users to manipulate text within the file or between files. Standard Edit menu items include Undo, Cut, Copy, Paste, Clear, Delete, and Select. If your application uses an Edit menu, position it to the right of the File menu. If your application uses mnemonics, the File menu mnemonic should be E .
View	The View menu contains items that allow users to change the way they view the data. This might include items that change the appearance of the data, the amount of data that is displayed, or the order in which the data is displayed. The contents of this menu is specific to your application. If your application uses a View menu, position it to the right of the Edit menu. If your application uses mnemonics, the View menu mnemonic should be V .
Options	The Options menu contains items that allow users to customize the application. The contents of this menu is specific to your application. If your application uses an Options menu, position it to the right of the View menu. If your application uses mnemonics, the Options menu mnemonic should be O .
Help	The Help menu contains items that allow users to access help information. If your application uses a Help menu, position it as the last item on the bar menu, flush right. If your application uses mnemonics, the Help menu mnemonic should be H .

4.6.2 Pull-Down Menus

A pull-down menu appears when the user selects an item from a bar menu or a cascade item from another menu. Pull-down menus are contained within a box. When a pull-down menu is displayed from a bar menu, the top-left corner of the pull-down menu box appears one line below and one character to the left of the first letter of a bar menu item. When a pull-down menu is displayed as a result of selecting a cascade item from another menu, the top-left corner of the menu box appears one line above and one column to the right of the cascade indicator (→), as shown in Figure 4–8.

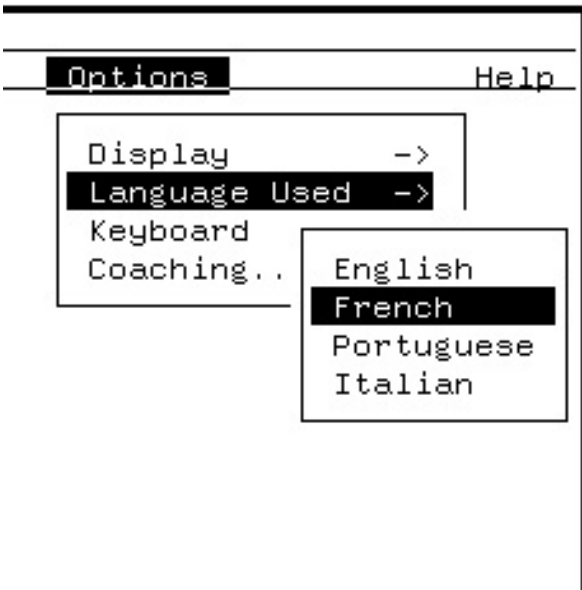
Figure 4-8 Location of Pull-Down Menu



When a pull-down menu is displayed, the menu item from which it came remains highlighted in reverse video to indicate clearly the path used to display the current menu.

There are some conditions under which the recommended menu positions are not possible or do not make sense to the application. For example, in Figure 4-9 there is not enough room on the right for the submenu to be completely displayed, so it is shifted to the left until it can be displayed and dropped down a line so that the name of the menu entry that evoked it can be seen.

Figure 4-9 Sample of a Submenu Cascaded Downward



If you do not have room for a submenu because you run out of space at the right side of the screen, use one of the following methods:

- If you need only a couple more columns to fit a submenu on the screen, shift the submenu to the left to increase the overlap with the menu entry from which it came. Avoid obscuring important menu labels.
- Cascade the submenu downward. The right side of the menu will be flush right. The top of the cascaded menu should appear on the line beneath the selected submenu entry, as shown in Figure 4-9.
- If there is not enough vertical room to cascade a submenu downward, position the bottom-right corner of the submenu in the bottom-right corner of the work area.

4.6.3 Pop-Up Menus

Pop-up menus look similar to pull-down menus and are displayed when the user presses the PF3 key. Pop-up menus can be context sensitive; that is, you can condition the menu contents based on field values. Pop-up menus are available quickly and at any time. They serve as a shortcut by reducing the number of keystrokes needed to perform a task. However, nothing on the application screen indicates the availability of pop-up menus.

Because pop-up menus provide shortcuts, each item in a pop-up menu must be available through another mechanism, such as a pull-down menu or a dialog box.

You can implement pop-up menus in one of two ways:

- Create one generic pop-up menu that includes all the pop-up menu items appropriate for your application. Then, depending on the context when users select the pop-up menu, deactivate the inappropriate menu items.
- Create a number of pop-up menus for different contexts.

Pop-up menus generally appear in the center of what the user is doing. However, the needs of the application determine the best position for a pop-up menu. For example, if the user needs to see certain data on the screen when using the pop-up, you should place the pop-up where it does not obscure that data. If a pop-up menu can generate three levels of cascade menus, you might need to place the pop-up toward the left side of the display to make room for the cascade menus.

4.7 Designing Menus

When you are designing menus, there are two major questions you have to answer: how to name the menu items, and how to group the menu items. Section 4.7.1 and Section 4.7.2 provide guidelines for naming and grouping menu items.

4.7.1 Naming Menu Items

Consider the following guidelines when naming menu items:

- Use the standard terms for common menus, such as File, Edit, Options, and Help.
- Use distinct names for each menu and its menu items.
- Use verbs or adjectives for your menus and menu items. Verbs and adjectives give users a better idea than nouns do of what action a command performs.
- Use terminology that is familiar to your users.
- Provide keyboard accelerators or mnemonics for frequently chosen menu items. This allows users familiar with the application to take short cuts, increasing their efficiency.
- Ask users to review your choices of menus and menu items.

4.7.2 Grouping Menu Items

Consider the following guidelines when grouping and ordering menu items:

- Organize the items in your menu into logical groups.
- Sets of related items, such as radio items or toggle items, should be located together and separated from other menu items by a horizontal separator.
- Order menu items according to the frequency of use. Place items that are more frequently used at the top of the menu. The fewer keystrokes required to perform the most common tasks, the better.
- Separate destructive commands (such as Delete or Quit) from other frequently chosen items. One of the most common problems with menus is the *off by one* error, in which the user mistakenly chooses the item next to the one intended.
- Keep your menu structure simple. Use as few submenu levels as possible, with three levels as a recommended maximum. If your application seems to need more levels than this, consider using dialog boxes or more menu items on your bar menu.

Dialog Boxes

5.1 Purpose

A dialog box is a pop-up panel that is used to gather or display information. An application can display a dialog box to notify the user of some event, such as caution or work in progress information. An application can also display a dialog box in response to a user command to obtain more information. For example, an ellipsis (...) following a menu item or text entry field label clues the user visually that selecting that item will display a dialog box. The user uses controls in the dialog box to respond to the input requests or messages.

As you design your application, use dialog boxes to organize data and objects to help your users. If you use dialog boxes to solicit detailed or additional input, to control subtasks, or to gather information on infrequently used choices, you can keep your main screens simple and uncluttered.

5.2 Appearance

A dialog box is a pop-up panel that contains a combination of text and controls (push buttons, radio buttons, check buttons, list groups, text entry fields, and so forth). Most dialog boxes have a title. This title should reflect the command that created it. For example, the Add Registration... menu item generates a dialog box with the title *Add Registration*. The title of a dialog box is left justified in the banner at the top of the dialog box. This banner is created by a horizontal line of reverse video characters that are indented by one character from either side of the box, as shown in Figure 5-1.

Figure 5-1 Sample Dialog Box

The dialog box has a title bar that reads "Add Registration". Below the title bar, there are four input fields with labels and values: "Registration Number: 123456789", "First Name: Arnold", "Last Name: Brookston", and "Country... : Australia". Below these fields is a section titled "Events" containing a list of sports events, each with a checkbox and a corresponding distance or relay type. The events are: Shot Put (checked), High Jump (unchecked), Javelin (checked), Pole Vault (unchecked), Discus (checked), and Long Jump (unchecked). The distances are: 100 Meter, 400 Meter, 5,000 Meter, 10,000 Meter, and 4x4 Relay. At the bottom of the dialog box are two buttons: "OK" and "Cancel".

Event	Selected	Distance/Relay
Shot Put	<input checked="" type="checkbox"/>	100 Meter
High Jump	<input type="checkbox"/>	400 Meter
Javelin	<input checked="" type="checkbox"/>	5,000 Meter
Pole Vault	<input type="checkbox"/>	10,000 Meter
Discus	<input checked="" type="checkbox"/>	4x4 Relay
Long Jump	<input type="checkbox"/>	

5.3 Size and Placement

A dialog box should be large enough to present the dialog information and controls without looking crowded.

The placement of the dialog box depends on its purpose and contents. When a dialog box is evoked, it overlays whatever is on that portion of the screen and it cannot be moved on the screen. Make sure the dialog box does not obscure information in the underlying panel that the user might need when responding to the dialog box.

5.4 Ending a Dialog

Each dialog box needs a mechanism to make the dialog box disappear once the user is finished with it. Also, any dialog box that solicits additional information should give the user the option of canceling the request. When a dialog box is cancelled, the application returns to the state that existed before the dialog box was evoked. The typical way of providing these controls is to have the dialog box include an OK push button and a Cancel push button (when appropriate). Keyboard accelerators should always be available (the F10 key and the Ctrl/Z keys for OK and the F8 key and the PF1-Q key sequence for Cancel).

5.5 Chaining Dialog Boxes

You might need more than one dialog box to complete an informational exchange with a user. To chain dialog boxes, use a Continue... push button or a More... push button.

5.6 Grouping Controls

The overall design of your dialog box depends on the content and complexity of your application. Group controls and labels into logical sections to make the user interface as easy as possible. Arrange these sections within the dialog box based on the order in which the user needs them and on the order in which the user scans them (in most cases this will be from top to bottom and left to right). This arrangement helps to limit the amount of navigation that is necessary for the user to complete the dialog. Separate these sections with blank spaces rather than horizontal or vertical lines.

5.6.1 Arranging Push Buttons

Arrange push buttons in a dialog box in an order that corresponds to the progression in which the user needs them. Because push button controls are often used to end a dialog, they are placed typically either in a row at the bottom (preferable) or in a column at the right side of the dialog box. When the push buttons are placed in a horizontal row, make each push button box the size of the label, as shown in Figure 5-2. When the push buttons are stacked vertically, make the boxes the same size and center the label, as shown in Figure 5-3.

Figure 5-2 Push Buttons Arranged Horizontally

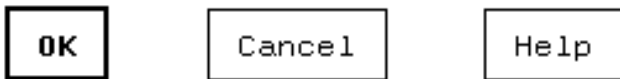


Figure 5-3 Push Buttons Arranged Vertically



Push buttons commonly found in grouped sets are:

- OK, Cancel, Help
- OK, Reset, Cancel, Help
- OK, Apply, Reset, Cancel, Help
- Yes, No, Help

When grouping push buttons, place the positive (active) choices before the negative (passive) choices, and make the Help selection the last choice.

5.6.2 Default Push Buttons

If a dialog box has more than one push button, you might want to make one of them the default push button. This is the most likely response to the dialog box query. Generally, the OK push button is the default push button. The user can press the F10 key or Ctrl/Z, the accelerator for the default operation, to quickly select the default push button.

If the action evoked by a push button is potentially destructive (for example, if loss of data could occur), that button should not be the default push button. Section 3.1.1. describes the appearance of a default push button.

5.6.3 Using Radio Boxes

The radio box is a set of related radio fields and a corresponding title. When the user selects from a small number of options (typically from 2 to 10), a radio box is the most direct method.

There are two cases in which a radio box does not need a title:

- When the radio box is the only control in the dialog box, in which case the title in the dialog box banner acts as the title of the radio box
- When the information in the radio box is so clear that there can be no confusion over what is being chosen

The title of the radio box is left-justified over the left angle bracket of the radio fields. The title uses initial caps. Whenever possible, radio buttons should be stacked vertically, as shown in Figure 5–4.

Figure 5–4 Vertically Stacked Radio Box

```
Print Format:  
<◆> Display on Screen  
< > Print List  
< > Print to File
```

If you must arrange radio fields horizontally, left-justify the title and use a colon (:) to separate it from the radio buttons, as shown in Figure 5–5.

Figure 5–5 Horizontally Arranged Radio Box

```
Aspect Ratio: <◆> 1 to 1   < > 2 to 1
```

5.6.4 Arranging Text Entry Fields

When multiple text entry fields appear near each other on the same screen, there are two methods to stack the fields:

- When the text entry fields are related logically and the labels are all about the same length, stack the colons vertically, right-justify the labels, and left-justify the data fields. Figure 5–6 shows this first method.

- When the text entry fields are not related logically or if the label lengths are very different, left-justify the labels. The colon appears directly to the right of each label. Left-justify the data fields with the data field corresponding to the longest label. Figure 5–7 shows this second method.

Figure 5–6 First Method of Stacking Text Entry Fields

```

First Name : Richard
Last Name  : Brown
Middle Name: Worthington

```

Figure 5–7 Second Method of Stacking Text Entry Fields

```

Major/Minor: History / Psychology
University Attended: State University
Degree: Bachelor of Arts
Date: 06/03/73

```

5.7 Specialized Dialog Boxes

Your DECforms applications can use the following standard dialog boxes for common operations:

- Work in Progress dialog box
- Informational dialog box
- Question dialog box
- File Selection dialog box

5.7.1 Work in Progress Dialog Box

Your application should display a Work in Progress dialog box whenever it determines that an operation will take longer than 5 seconds. Use a short simple message in the dialog box to tell your user what is happening. Display this message as blinking text. This gives the user confidence that the application is processing normally. This type of dialog box is for display only and does not allow user input. (If the operation will take 5 seconds or less, a message displayed in the message panel is sufficient.)

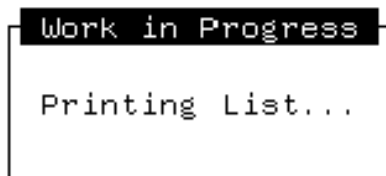
If your application can determine the amount of progress that has been made, you also can display the progress in the dialog box. The type of progress indicator depends on your application. For example, you might use a horizontal bar as a percentage indicator.

Make the dialog box a pleasing size with at least two blank lines above and below the message and two spaces on either side of the centered message. Center the Work in Progress dialog box on the screen, unless there is some reason not to obscure that part of your screen. The title in the dialog box banner should be *Work in Progress*

When the operation is complete, your application should display a completed message (for example, “File conversion completed.”) in the message panel and dismiss the dialog box.

Figure 5–8 shows a typical Work in Progress dialog box.

Figure 5–8 Typical Work in Progress Dialog Box



5.7.2 Informational Dialog Box

You should display routine informational messages in the message panel. However, an informational dialog box can be used when a message is important enough to interrupt the user and require acknowledgement (for example, if your application is unable to open a data file). This type of dialog box can be used for important messages, warnings, and error messages.

Center the informational dialog box on the screen, unless there is some reason not to obscure that part of your screen. Although the title in the dialog box banner depends on the purpose of the message, try to use a simple title such as *Warning*, *Error*, or *Information*.

The dialog box must contain an OK push button that the user selects to dismiss the box. The dialog box also might contain a Help push button, used to display additional information.

Figure 5-9 shows a typical informational dialog box.

Figure 5-9 Typical Informational Dialog Box



5.7.3 Question Dialog Box

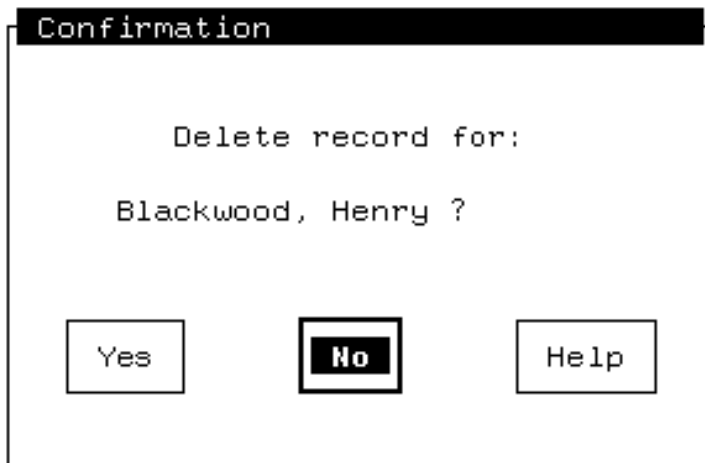
A question dialog box asks the user a brief question and provides a choice of mutually exclusive alternatives as push buttons. The push buttons use easy label pairs such as Yes and No or OK and Cancel. You can also provide a Help push button.

Question dialog boxes are used typically to caution the user and to confirm an action.

Center the question dialog box on the screen, unless there is some reason not to obscure that part of your screen. Although the title in the dialog box banner depends on the purpose of the question, try to use a simple title, such as *Question*, *Caution*, or *Confirmation*.

Figure 5–10 shows a typical Question dialog box.

Figure 5–10 Typical Question Dialog Box



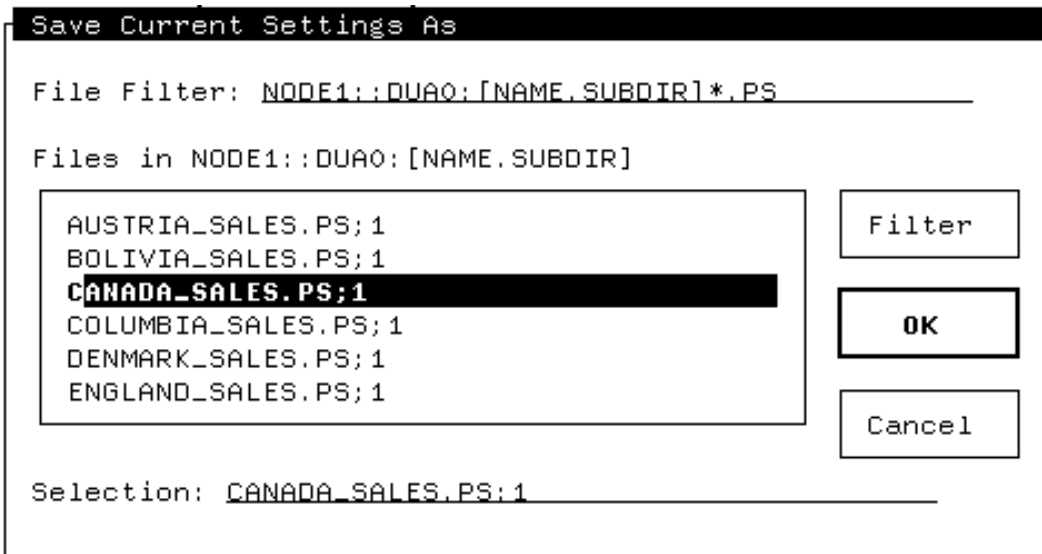
5.7.4 File Selection Dialog Box

A file selection dialog box allows the user to specify a file name within your application. A file selection box contains the following controls:

- File Filter text entry field
- Selection text entry field
- List box and list box label *Files in xxx* where *xxx* is the current directory name
- Filter, OK, and Cancel push buttons

Figure 5–11 shows a file selection dialog box.

Figure 5-11 Typical File Selection Dialog Box



To choose a file, the user selects a file name from the list box; the file name then appears in the Selection text entry field. The user then selects the OK push button.

The user can also enter a file name in the Selection text entry field and then select the OK push button. If an error occurs (for instance, if the file does not exist), your application should display an appropriate Caution dialog box.

To display a specific type of file in the list box, the user enters a file filter in the File Filter text entry field, using a directory name or wildcards. The user then selects the Filter push button or presses the Return key in the Filter field to display a new list of specified files. The application can provide a default File Filter string when the file selection box first is displayed, thereby limiting the user's choices to the appropriate files.

Table 3-2 shows the keys used to navigate within the scrolled list. Table 5-1 shows the additional keys used to navigate within a file selection box.

Table 5–1 Additional Keys Used Within a File Selection Dialog Box

LK-Series Keyboard Keys ¹	VT100-Series Keyboard Keys	Action
F10	Ctrl + Z	Pressing this is like selecting the OK push button. The file in the Selection field is chosen and the file selection box disappears. If the Selection field is empty, the highlighted file is selected.
Return	Return	Moves the highlight to the next field. If the cursor is in the Filter field when this is pressed, the scrolled list is updated to match the filter specifications, and the highlight moves to the top line of the scrolled list.
F8	PF1 Q	Pressing this is like selecting the Cancel push button. The file selection box is dismissed with no other effects.

¹LK-series keyboards also can use the VT100-series keys, except for the Linefeed and Backspace keys, which are not on an LK-series keyboard.

Part II

Implementing DECforms Style

This part contains information about how to use DECforms to implement the user interface described in Part I.

Each chapter discusses specific style elements and the code used to create them. Appendix A contains the IFDL code used to create the Track and Field Registration application (TAFR). Appendix B contains the C program, as well as instructions on how to run TAFR.

6

Implementing Controls

DECforms software has provided a sample program, the Track and Field Registration program (TAFR), that shows how to implement DECforms style. TAFR is written using the DECforms portable API.

TAFR was written by DECforms engineers who made decisions on how to implement each element, taking processing speed, good coding practice, and integral DECforms elements into account.

Although the TAFR application is not entirely complete, it should be useful in assisting you in developing DECforms style. This chapter describes how TAFR implements controls.

6.1 Push Buttons

Push buttons are implemented as icons. Icons are IFDL elements used to display information that does not change. Icons accept function key input only (no data input), so they work well as push buttons.

Example 6–1 shows the IFDL code used to create the OK and Cancel push buttons on the Add Registration dialog box, shown in Figure 5–1.

Example 6–1 OK and Cancel Push Buttons IFDL Code Example

```
Form TAFR_FORM
.
.
.
Panel ADD_REGISTRATION_DIALOG
.
.
.
    Icon ADD_REG_OK_BUTTON                1
        Apply Field Default BUTTON_DEFAULTS  2
        Function Response SELECT            3
            Message
                "Recording registrant data."
            Deactivate
                Panel ADD_REGISTRATION_DIALOG
            Remove
                ADD_REGISTRATION_VIEWPORT
            Position To Previous Item
        End Response
    Literal Text                            4
        Line 17
        Column 20
        Value " OK "
        Display
            Bold
        End Literal
    End Icon
    Icon ADD_REG_CANCEL_BUTTON            5
        Apply Field Default BUTTON_DEFAULTS  6
        Function Response SELECT            7
            Deactivate
                Panel ADD_REGISTRATION_DIALOG
            Remove
                ADD_REGISTRATION_VIEWPORT
            Position To Previous Item
        End Response
    Literal Text                            8
        Line 17
        Column 27
        Value " Cancel "
        End Literal
    End Icon
```

- 1 The icon is declared and named ADD_REG_OK_BUTTON.

- 2 The field default application specifies that a set of defaults, `BUTTON_DEFAULTS`, be applied to the icon. `BUTTON_DEFAULTS` was declared earlier in the form, as follows:

```
Field Default BUTTON_DEFAULTS
  Active Highlight
  Reverse
End Default
```

This means that the OK push button is highlighted in reverse video when the cursor is positioned on it. The push button is bolded to show that it is the default.

- 3 This statement declares a function response, `SELECT`, for the OK push button. When OK is selected (by pressing the Select key while the cursor is positioned on OK), the message "Recording registrant data" is displayed in the default message panel, and the Add Registration dialog box is removed from the display. The cursor is then positioned to the previous item, which is the File selection on the TAFR bar menu.
- 4 This literal declaration specifies that the push button is displayed at line 17, column 20, and that its label is displayed in bolded text.
- 5 The icon that specifies the Cancel push button, `ADD_REG_CANCEL_BUTTON`, is declared.
- 6 The same set of field defaults that were applied to the OK push button are specified for the icon that creates the Cancel button.
- 7 The `SELECT` function response is declared for the Cancel push button. If the Select key is pressed while the cursor is positioned on Cancel, the Add Registration panel is deactivated and the Add Registration dialog box is removed from the display. The cursor is once again positioned on the previous item, the File selection on the bar menu.
- 8 This literal declaration specifies that Cancel is displayed at line 17, column 27.

6.2 Radio Fields

Radio fields are implemented as panel fields. Fields are IFDL elements used to request and display information on panels. Icons are not used to implement radio fields because icons do not allow dynamic changes to labels or field values. In radio fields, you may want to change field values. The field value "Hebrew" changes to "{ Hebrew }" in Example 6-2, when it becomes an invalid choice.

Example 6–2 shows the IFDL code used to create the radio fields on the Print Options dialog box, shown in Figure 4–2.

Example 6–2 Radio Fields IFDL Code Example

```

Form TAFR_FORM
.
.
.
Group LANGUAGE_RADIOBOX                                1
  Occurs 3
  Current N3                                           2
  TOGGLE Unsigned Byte                                3
  TAG Character(21)                                    4
  PROTECT Unsigned Byte                               5
End Group
.
.
.
Let LANGUAGE_RADIOBOX(1).TAG = " English"             6
Let LANGUAGE_RADIOBOX(2).TAG = " French"              7
Let LANGUAGE_RADIOBOX(3).TAG = " Hebrew"             8
.
.
.
Panel OPTIONS_PULLDOWN_PANEL                           9
  Viewport OPTIONS_PULLDOWN_VIEWPORT                 10
  Function Response MAGIC                             11
    If (LANGUAGE_RADIOBOX(3).PROTECT = 0) Then
      Let LANGUAGE_RADIOBOX(3).PROTECT = 1
      Let LANGUAGE_RADIOBOX(3).TAG = "{Hebrew}"
    Else
      Let LANGUAGE_RADIOBOX(3).PROTECT = 0
      Let LANGUAGE_RADIOBOX(3).TAG = " Hebrew"
    End If
  End Response
.
.
.
Group LANGUAGE_RADIOBOX                                12
  Vertical
  Literal Text
    Line 7
    Column 3
    Value "< >"
  End Literal

```

(continued on next page)

Example 6–2 (Cont.) Radio Fields IFDL Code Example

```

Field TOGGLE                                     13
  Line 7
  Column 4
  Display                                         14
    Character Set Private_Rule
  Output Picture X
  Output " "                                       15
    When (LANGUAGE_RADIOBOX(**).TOGGLE = 0)
  Output ""                                        16
    When (LANGUAGE_RADIOBOX(**).TOGGLE = 1)
  Protected                                       17
End Field

Field TAG                                         18
  Line 7
  Column 6
  Function Response SELECT                         19
    Reset
      LANGUAGE_RADIOBOX(*).TOGGLE
    Let LANGUAGE_RADIOBOX(N3).TOGGLE = 1
  End Response

  Active Highlight                               20
    Reverse
  Output Picture X(22)
  No Data Input
  Protected                                       21
    When (LANGUAGE_RADIOBOX(**).PROTECT = 1)
End Field

End Group

Literal Text                                     22
  Line 10
  Column 2
  Value "....."
End Literal

```

- 1 A form data group, LANGUAGE_RADIOBOX, is declared. The LANGUAGE_RADIOBOX group is a one-dimensional array of three elements, LANGUAGE_RADIOBOX(1), LANGUAGE_RADIOBOX(2), and LANGUAGE_RADIOBOX(3). TAG, TOGGLE, and PROTECT are the three fields.
- 2 The CURRENT clause specifies that the data item N3 holds the index of the current field in LANGUAGE_RADIOBOX. For example, if the cursor is positioned on "French", then N3 equals 2.

- 3 This statement specifies that the first form data item in the LANGUAGE_RADIOBOX group has an UNSIGNED BYTE data type and is specified as LANGUAGE_RADIOBOX(n).TOGGLE.
- 4 This statement specifies that the value of the second form data item in the LANGUAGE_RADIOBOX group has a CHARACTER data type and is specified as LANGUAGE_RADIOBOX(n).TAG.
- 5 This statement specifies that the value of the third form data item in the LANGUAGE_RADIOBOX group has an UNSIGNED BYTE data type and is specified as LANGUAGE_RADIOBOX(n).PROTECT.
- 6 This LET response assigns 'English' as the value of the LANGUAGE_RADIOBOX(1).TAG form data item. LANGUAGE_RADIOBOX(1).TAG appears as the first choice of the radio fields in the Print Options dialog box—English.
- 7 This LET response assigns 'French' as the value of the LANGUAGE_RADIOBOX(2).TAG form data item, making French the second choice of the radio fields.
- 8 This LET response assigns 'Hebrew' to the LANGUAGE_RADIOBOX(3).TAG form data item, making Hebrew the third choice of the radio fields.
- 9 This PANEL declaration begins the OPTIONS_PULLDOWN_PANEL panel.
- 10 This VIEWPORT declaration specifies where OPTIONS_PULLDOWN_PANEL should be displayed. The viewport was specified earlier in the form in the following code:

```
Viewport OPTIONS_PULLDOWN_VIEWPORT
  Lines 3 Through 16
  Columns 11 Through 67
End Viewport
```

- 11 The MAGIC function response is declared. (Earlier in the form, MAGIC was associated with the F20 key.) If the F20 key is pressed within the OPTIONS_PULLDOWN_PANEL, the value of LANGUAGE_RADIOBOX(3).PROTECT is examined. Depending on its value, LANGUAGE_RADIOBOX(3).PROTECT and LANGUAGE_RADIOBOX(3).TAG are modified. The new values are used in the TAG panel field description in ¹⁸.

This switches the Hebrew choice from available to unavailable, or from unavailable to available. Pressing the F20 key demonstrates how to make a choice unavailable.

- 12 This GROUP declaration specifies a panel group, LANGUAGE_RADIOBOX, associated with the form data group of the same name. LANGUAGE_RADIOBOX is specified as a vertical group, appearing at line 7, column 3. The panel group is composed of angle brackets, and the TOGGLE and TAG fields.
- 13 The TOGGLE field associates a field on the panel with the items in LANGUAGE_RADIOBOX(n).TOGGLE. This field is displayed at line 7, column 4, between the angle brackets specified in 12.
- 14 This DISPLAY response step specifies that the values in LANGUAGE_RADIOBOX(n).TOGGLE are displayed in the Rule character set.
- 15 This OUTPUT WHEN clause tells the form to output nothing when the value in LANGUAGE_RADIOBOX.TOGGLE is set to 0.
- 16 This OUTPUT WHEN clause tells the form to output a filled-in diamond (an apostrophe (') in the Rule character set) when the value in LANGUAGE_RADIOBOX.TOGGLE is set to 1.
- 17 The PROTECTED clause specifies that the field is a display-only field.
- 18 This FIELD declaration specifies that a field named TAG be displayed at line 7, column 6—three columns over from the angle brackets specified in 12. The values specified in the LANGUAGE_RADIOBOX(1).TAG, LANGUAGE_RADIOBOX(2).TAG, and LANGUAGE_RADIOBOX(3).TAG are English and French, respectively. The value in LANGUAGE_RADIOBOX(3).TAG depends on the value in LANGUAGE_RADIOBOX(3).PROTECTED. It may be "Hebrew" or "{ Hebrew }" and the TAG field may or may not be protected. These values are displayed in a vertical list, starting at line 7, column 6.
- 19 This function response specifies that when the Select key is pressed in this field, LANGUAGE_RADIOBOX(*).TOGGLE is reset to 0. LANGUAGE_RADIOBOX(N3).TOGGLE is set to 1 where N3 is the index of the current occurrence. A filled-in diamond appears between the angle brackets specified in 12.
- 20 This ACTIVE HIGHLIGHT clause specifies that the LANGUAGE_RADIOBOX.TAG should be displayed in reverse video if the cursor is positioned on it.
- 21 This PROTECTED WHEN clause specifies that when the value in LANGUAGE_RADIOBOX(n).PROTECT is set to 1, the field is protected. This is what causes the Hebrew choice to be protected when you press the F20 key.

- 22 This literal places a line of dots between the radio fields and the command items in the Print Options... menu.

6.3 Check Fields

Check fields are fields that are not mutually exclusive: you can select more than one check field in a list. The check fields in the Add Registration dialog box (Figure 5–1) are implemented as fields and icons.

Example 6–3 shows the IFDL code to implement the check fields in the Add Registration dialog box.

Example 6–3 Check Fields IFDL Code Example

```
Form TAFR_FORM
.
.
.
Panel ADD_REGISTRATION_DIALOG                                1
  Viewport ADD_REGISTRATION_VIEWPORT                        2
  Function Response DISCARD                                3
    Deactivate
      Panel ADD_REGISTRATION_DIALOG
      Remove
        ADD_REGISTRATION_VIEWPORT
      Position To Previous Item
    End Response
.
.
.
  Literal Text                                              4
    Line 8
    Column 3
    Value "Events"
  End Literal
  Literal Text                                              5
    Line 9
    Column 3
    Value "[ ]"
  End Literal
```

(continued on next page)

Example 6–3 (Cont.) Check Fields IFDL Code Example

```
Field TOGGLE_SHOT_PUT                                6
  Line 9
  Column 4
  Display
    Character Set Private_Rule
  Output Picture X
  Output ``"                                         7
    When (TOGGLE_SHOT_PUT = 1)
  Output " "
    When (TOGGLE_SHOT_PUT <> 1)
  Protected
End Field

Icon EVENT_SHOT_PUT                                  8
  Apply Field Default BUTTON_DEFAULTS
  Function Response SELECT                             9
    If (TOGGLE_SHOT_PUT = 0) Then
      Let TOGGLE_SHOT_PUT = 1
    Else
      Let TOGGLE_SHOT_PUT = 0
    End If
  End Response

  Literal Text                                       10
    Line 9
    Column 6
    Value " Shot Put"
  End Literal

End Icon
```

- 1 This **PANEL** declaration specifies the panel that is displayed as the Add Registration dialog box.
- 2 This **VIEWPORT** declaration specifies the viewport in which the **ADD_REGISTRATION_DIALOG** panel is displayed. The **ADD_REGISTRATION_VIEWPORT** was specified earlier in the form as:

```
Viewport ADD_REGISTRATION_VIEWPORT
  Lines 2 Through 20
  Columns 11 Through 67
End Viewport
```

- 3 The **DISCARD** function response is declared for the **ADD_REGISTRATION_DIALOG** panel. When you press the F8 key or the PF1-Q key sequence on this panel, the **ADD_REGISTRATION_DIALOG** panel is deactivated, and the cursor is positioned on the previous item (the Registration choice on the main panel).

- 4 This LITERAL declaration places a string, "Events" on the panel at line 8, column 3.
- 5 This LITERAL declaration places a pair of square brackets ([]) on the panel at line 9, column 3. (This pair of brackets is the set of brackets next to the Shot Put check field.)
- 6 The TOGGLE_SHOT_PUT field is declared. TOGGLE_SHOT_PUT is displayed at line 9, column 4 (between the brackets specified in 5), and is displayed as a filled-in diamond when the value of TOGGLE_SHOT_PUT is 1. A filled-in diamond indicates that the corresponding toggle field was selected.
- 7 This OUTPUT WHEN clause outputs a filled-in diamond when the value of TOGGLE_SHOT_PUT is 1, and outputs nothing when the value of TOGGLE_SHOT_PUT is not equal to 1.
- 8 The ICON that corresponds to TOGGLE_SHOT_PUT is declared. The EVENT_SHOT_PUT icon has the attributes specified in BUTTON_DEFAULTS, declared earlier in the form as:

```
Field Default BUTTON_DEFAULTS
  Active Highlight
  Reverse
End Default
```

The result of applying BUTTON_DEFAULTS is that the string "Shot Put" is highlighted when active (displayed in reverse video).

- 9 The SELECT function response specifies that if the Select key is pressed in the EVENT_SHOT_PUT icon, the value of TOGGLE_SHOT_PUT is modified. If TOGGLE_SHOT_PUT is currently set to 0 (the corresponding toggle field was not already selected), it becomes 1, marking the corresponding field as selected. If the current value of TOGGLE_SHOT_PUT is 1, it is changed to 0, deselecting the corresponding toggle field. A filled-in diamond is displayed if the value is 1, and nothing is displayed if the value is 0.
- 10 This LITERAL TEXT declaration places the string "Shot Put" on the icon at line 9, column 6, next to the brackets specified in 5.

6.4 Text Entry Fields

Text entry fields allow you to enter and manipulate character strings. Text entry fields are implemented in TAFR as fields and literals.

Example 6–4 shows the IFDL code used to create the text entry fields in the Add Registration dialog box. (See Figure 5–1.)

Example 6–4 Text Entry Field IFDL Code Example

```
Form TAFR_FORM
.
.
.
    Literal Text                                1
        Line 3
        Column 3
        Value "Registration Number:"
    End Literal

    Field REGISTRATION_NUMBER                    2
        Line 3
        Column 24
        Apply Field Default TEXT_DEFAULTS        3
        Output Picture 9(9)
    End Field

    Literal Text                                4
        Line 4
        Column 3
        Value "First Name:"
    End Literal

    Field FIRST_NAME                             5
        Line 4
        Column 24
        Apply Field Default TEXT_DEFAULTS
    End Field

    Literal Text                                6
        Line 5
        Column 3
        Value "Last Name:"
    End Literal
```

(continued on next page)

Example 6–4 (Cont.) Text Entry Field IFDL Code Example

```
Field LAST_NAME                                7
  Line 5
  Column 24
  Apply Field Default TEXT_DEFAULTS
End Field

Literal Text                                    8
  Line 6
  Column 3
  Value "Country..."
End Literal

Field COUNTRY                                  9
  Line 6
  Column 24
  Apply Field Default TEXT_DEFAULTS
  Function Response SELECT                      10
    Let CALL_FROM = "ADD_REGISTRANTS "
    Activate
      Panel COUNTRY_LIST_OPTION_MENU
      Position To Panel COUNTRY_LIST_OPTION_MENU
    End Response
End Field
```

- 1 A text string, "Registration Number:" is displayed at line 3, column 3. This serves as the label for the text entry fields.
- 2 The REGISTRATION_NUMBER field, an all-numeric output picture, is declared at line 3, column 24.
- 3 The REGISTRATION_NUMBER field has the TEXT_DEFAULTS attributes applied to it. TEXT_DEFAULTS was specified earlier in the form, as follows:

```
Field Default TEXT_DEFAULTS
  Active Highlight
  Underlined
  Display
  Underlined
End Default
```

This set of defaults specifies that text is displayed with an underline.

- 4 A text string "First Name:" is displayed at line 4, column 3. (This places "First Name:" beneath "Registration Number:").
- 5 The FIRST_NAME field, a character text panel field, is declared at line 4, column 24, with the same defaults as specified in 3.

- 6 A text string, "Last Name:" is displayed at line 5, column 3. (This places "Last Name:" beneath "First Name:").
- 7 The LAST_NAME field, a character text panel field, is declared at line 5, column 24, with the same defaults as specified in 3.
- 8 A text string, "Country . . . :" is displayed at line 6, column 3. (This places "Country . . . :" beneath "Last Name:").
- 9 The field COUNTRY is declared; it is quite similar to the fields specified in 2, 5 and 7, with the same defaults.
- 10 This function response specifies that when the Select key is pressed, CALL_FROM is set to the text string "ADD_REGISTRANTS". CALL_FROM is used later in the COUNTRY_LIST_OPTIONS_MENU.
Next, the COUNTRY_LIST_OPTION_MENU panel is activated, and the list of participating countries is displayed in a pull-down menu on the screen. The cursor is positioned on the first country on the COUNTRY_LIST_OPTION_MENU.

6.5 List Groups

List groups are used to display lists of items when there is not enough room on the screen to display the entire list at one time. The best example of list groups in TAFR is in the File Filter dialog box. For an example of how list groups are implemented, see Section 8.3.

6.6 Option Fields

Option fields consist of labels and values. Example 6-5 is from the List Registrants by Country menu shown in Figure 2-4.

Example 6-5 Option Fields IFDL Code Example

(continued on next page)

Example 6-5 (Cont.) Option Fields IFDL Code Example

Form TAFR_FORM

```
.  
. .  
. .  
Panel COUNTRY_LIST_DIALOG_PANEL 1  
  Viewport COUNTRY_LIST_DIALOG_VIEWPORT 2  
  Function Response TRANSMIT 3  
    Deactivate  
      Panel COUNTRY_LIST_DIALOG_PANEL  
    Remove  
      COUNTRY_LIST_DIALOG_VIEWPORT  
    Include PERFORM_LIST_FUNCTION  
  End Response  
  
  Function Response DISCARD 4  
    Deactivate  
      Panel COUNTRY_LIST_DIALOG_PANEL  
    Remove  
      COUNTRY_LIST_DIALOG_VIEWPORT  
    Position To Previous Item  
  End Response  
  
  Literal Text 5  
    Line 4  
    Column 3  
    Value "Country:"  
  End Literal  
  
  Literal Rectangle 6  
    Line 3 Column 12  
    Line 5 Column 28  
  End Literal  
  
  Field COUNTRY_OPTIONS 7  
    Line 4  
    Column 13  
  Function Response SELECT 8  
    Let CALL_FROM = "LIST_REGISTRANTS"  
  Activate  
    Panel COUNTRY_LIST_OPTION_MENU  
    Position To Panel COUNTRY_LIST_OPTION_MENU  
  End Response
```

(continued on next page)

Example 6–5 (Cont.) Option Fields IFDL Code Example

```
        Active Highlight
          Reverse
        Output Picture ' 'X(13)
        No Data Input
        Output "      ...      "          9
          When (COUNTRY_OPTIONS = " ")
      End Field
    .
    .
    .
  End Panel
```

- 1 A panel, COUNTRY_LIST_DIALOG_PANEL, is declared.
- 2 The viewport in which COUNTRY_LIST_DIALOG_PANEL is displayed, COUNTRY_LIST_DIALOG_VIEWPORT, was declared earlier in the form, as follows:

```
Viewport COUNTRY_LIST_DIALOG_VIEWPORT
  Lines 6 Through 19
  Columns 25 Through 55
End Viewport
```

- 3 The TRANSMIT function response is declared for the COUNTRY_LIST_DIALOG_PANEL panel. When you press the F10 key on this panel, COUNTRY_LIST_DIALOG_PANEL is deactivated. The PERFORM_LIST_FUNCTION response is performed.
- 4 The DISCARD function response is declared for the COUNTRY_LIST_DIALOG_PANEL panel. When you press the F8 key or the PF1-Q key sequence on this panel, the COUNTRY_LIST_DIALOG_PANEL panel is deactivated, and the cursor is positioned on the previous item (the Registration choice on the main panel).
- 5 This LITERAL declaration places the label "Country:" in the option field.
- 6 This LITERAL declaration draws the box next to the "Country:" label in the option field.
- 7 The COUNTRY_OPTIONS field is declared. This field is associated with a CHARACTER(13) form data item that was declared earlier in the form. COUNTRY_OPTIONS reserves a space for the country that will be selected by the registrant.

- 8 When the Select key is pressed, `CALL_FROM` is set to the text string "LIST_REGISTRANTS". The value of `CALL_FROM` is used later in the `COUNTRY_LIST_OPTION_MENU`. The `COUNTRY_LIST_OPTION_MENU` panel is activated, and the cursor is positioned on the topmost item on the menu.
- 9 If the `COUNTRY_OPTIONS` field is not yet filled, an ellipsis (. . .) is displayed in the option field.

Implementing Menus

All menus are lists of items from which to choose. Although all menus are basically alike, not all menus are implemented in the same way. The following sections show how to implement bar menus, pull-down menus, and pop-up menus.

7.1 Bar Menu

The bar menu is implemented as a separate panel in the Track and Field Registration application (TAFR). Each menu item on the bar menu is an icon with an associated control field. The control field is used to display the selected menu item in reverse video when its pull-down menu appears. This makes it easy for the user to see the path of various menu items through the menu system.

Example 7-1 shows the TAFR bar menu with one menu entry, File. All other menu items are implemented in a similar fashion. The bar menu is shown in Figure 2-2.

Example 7-1 TAFR Bar Menu IFDL Code Example

```
Form TAFR_FORM
.
.
.
Form Data
  MODIFIED Unsigned Byte          1
  Value 0
.
.
.
  FILE_ENTRY_CONTROL Unsigned Byte 2
  REGISTRATION_ENTRY_CONTROL Unsigned Byte
  SCHEDULE_ENTRY_CONTROL Unsigned Byte
  OPTIONS_ENTRY_CONTROL Unsigned Byte
  HELP_ENTRY_CONTROL Unsigned Byte
  LIST_ENTRY_CONTROL Unsigned Byte
.
.
.
End Data
.
.
.
Layout VT_LAYOUT
.
.
.
  Viewport BAR_MENU_VIEWPORT      3
  Lines 2 Through 2
  Columns 1 Through 80
  End Viewport
  .
  .
  .
  Function DISCARD                 4
  Is %F8
  (%PF1 %CAPITAL_Q)
  (%PF1 %SMALL_Q)
  End Function
  .
  .
  .
  Internal Response QUIT_APPLICATION 5
  Signal
  Message
  "Quitting application. Data discarded."
  Return Immediate
```

(continued on next page)

Example 7-1 (Cont.) TAFR Bar Menu IFDL Code Example

```
End Response
.
.
Field Default MENU_DEFAULTS                                6
  Active Highlight
  Reverse
End Default
.
.
Panel BAR_MENU_PANEL                                      7
  Viewport BAR_MENU_VIEWPORT                              8
    Function Response DISCARD                              9
      If (MODIFIED = 1) Then
        Activate
          Panel CAUTION_BOX_PANEL
          Position To Icon QUIT_NO_BUTTON On CAUTION_BOX_PANEL
        Else
          Include QUIT_APPLICATION
        End If
      End Response
    Function Response MNEMONIC_F                            10
      Let FILE_ENTRY_CONTROL = 1
      Activate
        Panel FILE_PULLDOWN_PANEL
        Position To Panel FILE_PULLDOWN_PANEL
      End Response
    Apply Field Default MENU_DEFAULTS                      11
    Icon FILE_CASCADE_BUTTON                               12
      Function Response SELECT                              13
        Let FILE_ENTRY_CONTROL = 1
        Activate
          Panel FILE_PULLDOWN_PANEL
          Position To Panel FILE_PULLDOWN_PANEL
        End Response
```

(continued on next page)

Example 7–1 (Cont.) TAFR Bar Menu IFDL Code Example

```
Highlight                                14
    Reverse
    When (FILE_ENTRY_CONTROL = 1)
Literal Text                              15
    Line 1
    Column 2
    Value " File "
    Display
        Underlined
    End Literal
End Icon
```

- 1 The MODIFIED form data item is declared. It has an UNSIGNED BYTE data type, and its value is 0. The value of MODIFIED is used later in the DISCARD function response.
- 2 This statement declares the FILE_ENTRY_CONTROL form data item to have an UNSIGNED BYTE data type. The value of FILE_ENTRY_CONTROL is used later in the MNEMONIC_F function response.
- 3 The viewport for the TAFR bar menu, BAR_MENU_VIEWPORT, is specified as line 2 through 2, columns 1 through 80. The BAR_MENU_VIEWPORT viewport is the top two lines of the display, all the way across the display.
- 4 The DISCARD function is declared to occur when the F8 key, or the PF1-Q key sequence is pressed. The action that occurs when the DISCARD function is invoked is specified in 9.
- 5 An internal response, called QUIT_APPLICATION, is specified. QUIT_APPLICATION displays a message, "Quitting application. Data discarded.", when the DISCARD function is invoked in 9.
- 6 When MENU_DEFAULTS is applied to a field, the field is displayed in reverse video whenever the cursor is positioned to it.
- 7 The BAR_MENU_PANEL panel is declared.
- 8 This statement specifies that BAR_MENU_PANEL is displayed within BAR_MENU_VIEWPORT.

- 9 The DISCARD function response is defined for this panel. If the value of the MODIFIED form data item is equal to 1, CAUTION_BOX_PANEL is activated. The value of MODIFIED was set to 0 in 1. It is set to 1 later in the form, when the values in the COUNTRY_LIST_OPTIONS box are set to their initial values. If the CAUTION_BOX_PANEL panel is displayed, the cursor is positioned to the NO button on the panel that asks you if you wish to quit. If you have not changed any data, you quit the application immediately without being asked if you want to quit.
- 10 This function response associates the MNEMONIC_F function with this response. The value of the FILE_ENTRY_CONTROL form data item is set to 1, and FILE_PULLDOWN_PANEL is activated. This panel displays a pull-down menu with Exit and Quit choices. The cursor is positioned to the panel.
The MNEMONIC_F function occurs when the PF4 key and the F key are pressed in sequence. This function was declared earlier in the form, as follows:

```
Function MNEMONIC_F
  Is (%PF4 %CAPITAL_F)
    (%PF4 %SMALL_F)
End Function
```
- 11 This FIELD DEFAULT application specifies that the field defaults declared in 6 are applied to all icons within the BAR_MENU_PANEL. This apply clause is defined for the panel level.
- 12 The FILE_CASCADE_BUTTON icon is declared.
- 13 When the Select key is pressed within the icon, the FILE_PULLDOWN_PANEL panel is activated, and the cursor is positioned to it. This is the same action that the MNEMONIC_F function response invokes. An internal response could have been used to create the same effect.
- 14 When the icon is selected, a highlight is applied to it. The icon is displayed in reverse video.
- 15 This LITERAL TEXT declaration specifies that "File" is displayed at line 1, column 2, and that it is underlined. This makes "File" the first choice on the bar menu.

7.2 Pull-Down Menus

Pull-down menus appear when a user selects a menu name from the bar menu or a cascade item from another menu. The implementation of the List Registrants pull-down menu of the Add Registration dialog box, shown in Figure 4-6 is discussed in Example 7-2.

Example 7-2 List Registrants Pull-Down Menu IFDL Code

Form TAFR_FORM

```

.
.
.
Panel REGISTRATION_PULLDOWN_PANEL                                1
  Viewport REGISTRATION_PULLDOWN_VIEWPORT                        2
  Function Response DISCARD                                       3
    Deactivate
      Panel REGISTRATION_PULLDOWN_PANEL
    Remove
      REGISTRATION_PULLDOWN_VIEWPORT
    Position To Previous Item
    Let REGISTRATION_ENTRY_CONTROL = 0
  End Response
.
.
.
Function Response MNEMONIC_L                                      4
  Let LIST_ENTRY_CONTROL = 1
  Activate
    Panel LIST_PULLDOWN_PANEL
    Position To Panel LIST_PULLDOWN_PANEL
  End Response
Apply Field Default MENU_DEFAULTS                                5
Literal Text                                                     6
  Line 5
  Column 2
  Value "....."
End Literal
.
.
.
Icon LIST_CASCADE_BUTTON                                        7
  Function Response MOVE_RIGHT                                    8
    Let LIST_ENTRY_CONTROL = 1
    Activate
      Panel LIST_PULLDOWN_PANEL
      Position To Panel LIST_PULLDOWN_PANEL
    End Response
  Function Response SELECT                                        9
    Let LIST_ENTRY_CONTROL = 1
    Activate
      Panel LIST_PULLDOWN_PANEL
      Position To Panel LIST_PULLDOWN_PANEL
    End Response

```

(continued on next page)

Example 7-2 (Cont.) List Registrants Pull-Down Menu IFDL Code

```
Display                                     10
  Underlined
Highlight
  Reverse
  When (LIST_ENTRY_CONTROL = 1)
Literal Text                                 11
  Line 6
  Column 2
  Value " List Registrants                 ->"
  Display
  Nounderlined
End Literal

Literal Text
  Line 6
  Column 3
  Value "L"                                 12
  Display
  Underlined
End Literal

End Icon

Literal Rectangle                             13
  Line 1 Column 1
  Line 7 Column 33
End Literal

End Panel

Panel LIST_PULLDOWN_PANEL                     14
  Viewport LIST_PULLDOWN_VIEWPORT
  Function Response DISCARD                   15
    Let LIST_ENTRY_CONTROL = 0
    Deactivate
      Panel LIST_PULLDOWN_PANEL
    Remove
      LIST_PULLDOWN_VIEWPORT
    Position To Previous Item
  End Response

Apply Field Default MENU_DEFAULTS            16
Literal Rectangle                             17
  Line 1 Column 1
  Line 6 Column 14
End Literal
```

(continued on next page)

Example 7–2 (Cont.) List Registrants Pull-Down Menu IFDL Code

```
Icon BY_COUNTRY_ENTRY                                18
  Function Response SELECT                            19
    Deactivate
      Panel LIST_PULLDOWN_PANEL
    Remove
      LIST_PULLDOWN_VIEWPORT
    Deactivate
      Panel REGISTRATION_PULLDOWN_PANEL
    Remove
      REGISTRATION_PULLDOWN_VIEWPORT
    Activate
      Panel COUNTRY_LIST_DIALOG_PANEL
    Position To Panel COUNTRY_LIST_DIALOG_PANEL
    Let LIST_ENTRY_CONTROL = 0
    Let REGISTRATION_ENTRY_CONTROL = 0
  End Response

  Literal Text                                        20
    Line 2
    Column 2
    Value " By Country "
  End Literal

End Icon

Icon BY_EVENT_ENTRY                                  21
  Function Response SELECT                            22
    Signal
    Message
      "LIST BY EVENT function not yet implemented."
  End Response

  Literal Text                                        23
    Line 3
    Column 2
    Value " By Event  "
  End Literal

End Icon

Icon BY_NAME_ENTRY                                   24
  Function Response SELECT                            25
    Signal
    Message
      "LIST BY NAME function not yet implemented."
  End Response
```

(continued on next page)

Example 7–2 (Cont.) List Registrants Pull-Down Menu IFDL Code

```
        Literal Text                                26
            Line 4
            Column 2
            Value " By Name      "
        End Literal
    End Icon
    Icon BY_NUMBER_ENTRY                            27
        Function Response SELECT                    28
            Signal
            Message
                "LIST BY NUMBER function not yet implemented."
        End Response
        Literal Text                                29
            Line 5
            Column 2
            Value " By Number    "
        End Literal
    End Icon
End Panel
```

- 1 The panel on which the registration pull-down menu appears, `REGISTRATION_PULLDOWN_PANEL`, is declared.
- 2 The viewport in which `REGISTRATION_PULLDOWN_PANEL` is displayed, `REGISTRATION_PULLDOWN_VIEWPORT`, is declared.
- 3 The `DISCARD` function response is specified for `REGISTRATION_PULLDOWN_PANEL`. The `DISCARD` function response deactivates `REGISTRATION_PULLDOWN_PANEL` and removes the viewport from the display. `DISCARD` also positions the cursor to the previous item, and sets the `REGISTRATION_ENTRY_CONTROL` form data item to 0.
The `DISCARD` function response is invoked if the user presses the F8 key or the PF1-Q key sequence.
- 4 This function response associates the `MNEMONIC_L` function with this response. The value of the `LIST_ENTRY_CONTROL` form data item is set to 1, and the `LIST_PULLDOWN_PANEL` panel is activated. The cursor is positioned to the panel.

The MNEMONIC_L function occurs when the PF4 key and the L key are pressed in sequence. This function was declared earlier in the form, as follows:

```
Function MNEMONIC_L
  Is (%PF4 %CAPITAL_L)
    (%PF4 %SMALL_l)
End Function
```

- 5 A FIELD DEFAULT application specifying that MENU_DEFAULTS is applied to all fields and icons on the panel is declared. MENU_DEFAULTS specifies that fields are displayed in reverse video when they are active.
- 6 This LITERAL TEXT declaration places a line of periods between the List Registrants choice and the bottom of the menu.
- 7 The LIST_CASCADE_BUTTON icon is specified. LIST_CASCADE_BUTTON is an icon, not a field: only function key input is allowed.
- 8 The MOVE_RIGHT function response is declared for the icon. MOVE_RIGHT specifies that when the right arrow key is pressed in this icon, the LIST_PULLDOWN_PANEL panel is activated and the cursor is positioned to the first item on that panel. Pressing the right arrow key sets the form data item LIST_ENTRY_CONTROL to 1.
- 9 The SELECT function response is identical to the MOVE_RIGHT function response. Pressing the Select key has the same effect as pressing the right arrow key.
- 10 This DISPLAY clause specifies that LIST_CASCADE_BUTTON is underlined. The HIGHLIGHT clause specifies that this button is displayed in reverse video when LIST_ENTRY_CONTROL is 1.
- 11 This clause specifies that a text string "List Registrants ->" is displayed at line 6, column 2, of the icon.
- 12 This clause specifies that the L in "List Registrants" is underlined. It must be declared after the full text in 11.
- 13 This clause specifies the rectangle in which the Add Registrants menu is displayed.
- 14 The LIST_PULLDOWN_PANEL panel is specified. LIST_PULLDOWN_PANEL is displayed in the LIST_PULLDOWN viewport, declared earlier in the form, as follows:

```
Viewport LIST_PULLDOWN_VIEWPORT
  Lines 7 Through 12
  Columns 41 Through 54
End Viewport
```

- 15 The DISCARD function response is specified for LIST_PULLDOWN_PANEL. The DISCARD function response deactivates LIST_PULLDOWN_PANEL and removes the viewport from the display. DISCARD also positions the cursor to the previous item, and sets the form data item LIST_ENTRY_CONTROL form data item to 0.
The DISCARD function response is invoked if the user presses the F8 key or the PF1 and Q keys in sequence.
- 16 This FIELD DEFAULT application specifies that MENU_DEFAULTS be applied to the fields and icons declared in LIST_PULLDOWN_PANEL.
- 17 This LITERAL declaration specifies the rectangle in which the List Registrants pull-down menu is displayed.
- 18 This icon specifies the first choice in the List Registrants pull-down menu, By Country.
- 19 The SELECT function response is declared for the icon. SELECT specifies that when the Select key is pressed in this icon, the LIST_PULLDOWN_PANEL panel is deactivated and removed from the display. The REGISTRATION_PULLDOWN_PANEL panel is also deactivated and removed from the display. The COUNTRY_LIST_DIALOG_PANEL panel is activated, and the cursor is positioned to the first item on that panel. Pressing the Select key sets the form data item LIST_ENTRY_CONTROL to 0, and the form data item REGISTRATION_ENTRY_CONTROL to 0.
- 20 This LITERAL declaration specifies that a text string "By Country" is displayed at line 2, column 2—the first line of the List Registrants pull-down menu.
- 21 This icon specifies the second choice in the List Registrants pull-down menu, By Event.
- 22 The SELECT function response is declared for the icon. SELECT specifies that when the Select key is pressed in this icon, the message "LIST BY EVENT function not yet implemented" is displayed in the message panel.
- 23 This LITERAL declaration specifies that a text string "By Event" is displayed at line 3, column 2—the second line of the List Registrants pull-down menu—directly beneath ¹⁸.

- 24 This icon specifies the third choice in the List Registrants pull-down menu, By Name.
- 25 The SELECT function response is declared for the icon. SELECT specifies that when the Select key is pressed in this icon, the message "LIST BY NAME function not yet implemented" is displayed in the message panel.
- 26 This LITERAL declaration specifies that a text string "By Name" is displayed at line 4, column 2—the third line of the List Registrants pull-down menu—directly beneath 21.
- 27 This icon specifies the fourth choice in the List Registrants pull-down menu, By Number.
- 28 The SELECT function response is declared for the icon. SELECT specifies that when the Select key is pressed in this icon, the message "LIST BY NUMBER function not yet implemented" is displayed in the message panel.
- 29 This LITERAL declaration specifies that a text string "By Number" is displayed at line 5, column 2—the fourth line of the List Registrants pull-down menu—directly beneath 24.

7.3 Pop-Up Menus

Pop-ups are used to display lists of items when there is not enough room on the screen to display the entire list at time. In Example 7-3 the pop-up menu that is implemented is the country pop-up menu from the Add Registration dialog box, shown in Figure 3-14.

Example 7-3 TAFR Country Pop-Up Menu IFDL Code Example

```
Form TAFR_FORM
.
.
.
Group COUNTRY_LIST_OPTIONS                                1
  Occurs 12                                              2
  Current N2                                             3
  STRING Character(13)                                    4
  CONTROL Unsigned Byte                                  5
End Group
.
.
.
Enable Response                                          6
.
.
.
  Let COUNTRY_LIST_OPTIONS(1).STRING = "Australia"      7
  Let COUNTRY_LIST_OPTIONS(2).STRING = "Canada"         8
  Let COUNTRY_LIST_OPTIONS(3).STRING = "China"
  Let COUNTRY_LIST_OPTIONS(4).STRING = "Denmark"
  Let COUNTRY_LIST_OPTIONS(5).STRING = "Egypt"
  Let COUNTRY_LIST_OPTIONS(6).STRING = "England"
  Let COUNTRY_LIST_OPTIONS(7).STRING = "France"
  Let COUNTRY_LIST_OPTIONS(8).STRING = "Germany"
  Let COUNTRY_LIST_OPTIONS(9).STRING = "Japan"
  Let COUNTRY_LIST_OPTIONS(10).STRING = "Sweden"
  Let COUNTRY_LIST_OPTIONS(11).STRING = "United States"
  Let COUNTRY_LIST_OPTIONS(12).STRING = "Soviet Union"
.
.
.
Panel COUNTRY_LIST_OPTION_MENU                            9
  Viewport COUNTRY_LIST_OPTION_VIEWPORT                 10
  Apply Field Default MENU_DEFAULTS                    11
  Group COUNTRY_LIST_OPTIONS                            12
    Vertical                                             13
    Field STRING                                         14
      Line 2
      Column 3
      Output Picture X(14)
      Highlight                                          15
      Reverse
        When (COUNTRY_LIST_OPTIONS(**).CONTROL = 1)
      Protected
    End Field
```

(continued on next page)

Example 7-3 (Cont.) TAFR Country Pop-Up Menu IFDL Code Example

```
Field CONTROL                                16
  Line 2
  Column 2
  Entry Response                             17
    Let COUNTRY_LIST_OPTIONS(N2).CONTROL = 1
  End Response
  Exit Response                               18
    Let COUNTRY_LIST_OPTIONS(N2).CONTROL = 0
  End Response
  Function Response SELECT                   19
    If CALL_FROM = "LIST_REGISTRANTS" Then
      Let COUNTRY_OPTIONS = COUNTRY_LIST_OPTIONS(N2).STRING
    End If

    IF CALL_FROM = "ADD_REGISTRANTS " THEN
      Let COUNTRY = COUNTRY_LIST_OPTIONS(N2).STRING
    End If

    Deactivate
      Panel COUNTRY_LIST_OPTION_MENU
    Remove
      COUNTRY_LIST_OPTION_VIEWPORT
  End Response

  Output Picture X                           20
  No Data Input                              21
  Output " "                                 22
    When 1 = 1
  End Field
End Group

Literal Rectangle                            23
  Line 1 Column 1
  Line 14 Column 17
End Literal
End Panel
```

- 1 A form data group, COUNTRY_LIST_OPTIONS, is declared.
- 2 The COUNTRY_LIST_OPTIONS group is a one-dimensional array of 12 elements.
- 3 The variable N2 is used in 17, 18, and 19 to control actions based on cursor position.

- 4 This statement specifies that the first form data item in COUNTRY_LIST_OPTIONS(n) has a CHARACTER(13) string data type and is specified as COUNTRY_LIST_OPTIONS(n).STRING.
- 5 This statement specifies that the value of the second form data item in COUNTRY_LIST_OPTIONS(n) has an UNSIGNED BYTE data type and is specified as COUNTRY_LIST_OPTIONS(n).CONTROL.
- 6 This ENABLE response initializes all the values in the LET response steps that follow it when the form is enabled.
- 7 This LET response assigns 'Australia' as the value of the COUNTRY_LIST_OPTIONS(1).STRING form data item. COUNTRY_LIST_OPTIONS(1).STRING appears as the first menu choice in the By Country pop-up menu.
- 8 This LET response assigns 'Canada' as the value of the COUNTRY_LIST_OPTIONS(2).STRING form data item. COUNTRY_LIST_OPTIONS(2).STRING appears as the second menu choice in the By Country pop-up menu. The following LET statements assign values to all 12 elements in COUNTRY_LIST_OPTIONS(n).STRING
- 9 This PANEL declaration begins the COUNTRY_LIST_OPTION_MENU panel.
- 10 This VIEWPORT declaration specifies where COUNTRY_LIST_OPTION_MENU should be displayed. The viewport was specified earlier in the form, as follows:


```

      Viewport COUNTRY_LIST_OPTION_VIEWPORT
        Lines 2 Through 15
        Columns 40 Through 56
      End Viewport
      
```
- 11 A set of field defaults, MENU_DEFAULTS, is specified for the panel. MENU_DEFAULTS, declared earlier in the form, specify that fields are displayed in reverse video when activated.
- 12 This GROUP declaration specifies a panel group, COUNTRY_LIST_OPTIONS, associated with the form data group of the same name.
- 13 COUNTRY_LIST_OPTIONS is specified as a vertical group.
- 14 The field STRING appears at line 2, column 3. This field contains the values that are in COUNTRY_LIST_OPTIONS(n).STRING.
- 15 The field STRING is output in reverse video when its corresponding control variable (COUNTRY_LIST_OPTIONS(**).CONTROL) is set to 1.
- 16 The field CONTROL appears at line 2, column 2, to the left of 14.

- 17 The `COUNTRY_LIST_OPTIONS(N2).CONTROL` variable is used to specify where the cursor is (see 14).
- 18 When the `CONTROL` field is exited, the value in `COUNTRY_LIST_OPTIONS(N2).CONTROL` is set to 0.
- 19 When the Select key is pressed within this field, depending where the pop-up menu was called from, the `SELECT` response sets the appropriate variable.

If the user was on the List Registrants pull-down menu before the cursor moved to the pop-up menu, the `COUNTRY_LIST_OPTIONS(N2).STRING` value is stored in `COUNTRY_OPTIONS` to reflect the correct value when the user views List Registrants.

If the user was on the Add Registrants pull-down menu before he selected the pop-up menu, the `COUNTRY_LIST_OPTIONS(N2).STRING` value is stored in the `COUNTRY` field to reflect the correct value when the user views Add Registrants.

The `COUNTRY_LIST_OPTION_MENU` panel is deactivated. The `COUNTRY_LIST_OPTION_VIEWPORT` viewport is removed from the display.

- 20 An output picture is specified for the `CONTROL` field.
- 21 You can only type function keys, like the Select key or the right arrow key.
- 22 The control field is always displayed as a space.
- 23 This `LITERAL` declaration specifies the rectangle that is drawn around the country pop-up menu.

8

Implementing Dialog Boxes

Dialog boxes can be simple or complex. In the following examples, both simple and complex dialog boxes are implemented.

8.1 Work in Progress Dialog Box

The IFDL coding for the Work in Progress dialog box shown in Figure 5-8 is shown in Example 8-1.

Example 8-1 Work in Progress Dialog Box IFDL Code Example

```
Form TAFR_FORM
.
.
.
Panel WORK_IN_PROGRESS_BOX                1
  Viewport WORK_IN_PROGRESS_VIEWPORT      2
  Literal Rectangle                        3
    Line 1 Column 1
    Line 5 Column 20
  End Literal
  Literal Text                              4
    Line 1
    Column 2
    Value " Work in Progress "
    Display
    Reverse
  End Literal
```

(continued on next page)

Example 8–1 (Cont.) Work in Progress Dialog Box IFDL Code Example

```
Literal Text                    5
  Line 3
  Column 3
  Value "Printing List..."
  Display
    Blinking
End Literal
End Panel
```

- 1 The panel, `WORK_IN_PROGRESS_BOX`, is declared.
- 2 The viewport in which `WORK_IN_PROGRESS_BOX` is displayed, `WORK_IN_PROGRESS_VIEWPORT`, was declared earlier in the form, as follows:

```
Viewport CAUTION_BOX_VIEWPORT
  Lines 7 Through 18
  Columns 16 Through 64
End Viewport
```

- 3 This rectangle frames the panel.
- 4 This `LITERAL TEXT` declaration specifies that a text string, "Work in Progress" is displayed at line 1, column 2 of the panel. This text literal is displayed in reverse video. This text overlaps the rectangle specified in 3.
- 5 This `LITERAL TEXT` declaration specifies that a text string "Printing List..." is displayed at line 3, column 3. The text literal blinks.

8.2 Informational Dialog Box

Informational dialog boxes are used to convey warnings, cautions, and information to the user. Example 8–2 is the IFDL coding for the Quit Caution box from TAFR, similar to the box shown in Figure 5–10.

Example 8–2 Quit Caution Dialog Box IFDL Code Example

Form TAFR_FORM

```
.  
. .  
. .  
Panel CAUTION_BOX_PANEL 1  
  Viewport CAUTION_BOX_VIEWPORT 2  
  Function Response TRANSMIT 3  
    Include CANCEL_QUIT  
  End Response  
  Function Response DISCARD 4  
    Include NO_FUNCTION  
  End Response  
  Apply Field Default MENU_DEFAULTS 5  
  Literal Rectangle 6  
    Line 1 Column 1  
    Line 12 Column 49  
  End Literal  
  Literal Text 7  
    Line 1  
    Column 2  
    Value " Quit Caution Box"  
    Display  
    Reverse  
  End Literal  
  Literal Text 8  
    Line 3  
    Column 4  
    Value "Modifications made during this application"  
  End Literal  
  Literal Text 9  
    Line 4  
    Column 5  
    Value "session will be discarded. Do you really"  
  End Literal  
  Literal Text 10  
    Line 5  
    Column 19  
    Value "wish to quit?"  
  End Literal
```

(continued on next page)

Example 8–2 (Cont.) Quit Caution Dialog Box IFDL Code Example

```
Icon QUIT_YES_BUTTON                                11
    Function Response SELECT                          12
        Include QUIT_APPLICATION                     13
    End Response

    Literal Text                                     14
        Line 9
        Column 17
        Value " Yes "
    End Literal

End Icon

Literal Rectangle                                  15
    Line 8 Column 16
    Line 10 Column 22
End Literal

Literal Rectangle                                  16
    Line 8 Column 29
    Line 10 Column 34
End Literal

Icon QUIT_NO_BUTTON                                17
    Function Response SELECT                          18
        Include CANCEL_QUIT
    End Response

    Literal Text                                     19
        Line 9
        Column 30
        Value " No "
    End Literal

End Icon

End Panel
```

- 1 The panel, CAUTION_BOX_PANEL, is declared.
- 2 The viewport in which CAUTION_BOX_PANEL is displayed, CAUTION_BOX_VIEWPORT is specified. CAUTION_BOX_VIEWPORT was declared earlier in the form, as follows:

```
Viewport CAUTION_BOX_VIEWPORT
    Lines 7 Through 18
    Columns 16 Through 64
End Viewport
```

- 3 The TRANSMIT function response is specified for the panel. The TRANSMIT function response includes the CANCEL_QUIT internal response, declared earlier in the form, as follows:

```
Internal Response CANCEL_QUIT
  Signal
  Message
    "Quit function cancelled."
  Deactivate
    Panel CAUTION_BOX_PANEL
  Remove
    CAUTION_BOX_VIEWPORT
  Position To Previous Item
End Response
```

This function response, invoked when the F10 key is pressed, displays the message "Quit function cancelled" in the message panel and deactivates the panel. The viewport is removed, and the cursor is positioned to the previous panel.

- 4 The DISCARD function response specifies that the NO_FUNCTION internal response happens when the F8 key or the PF1-Q key sequence is pressed. The NO_FUNCTION internal response was specified earlier in the form, as follows:

```
Internal Response NO_FUNCTION
  Signal
  Message
    "That key has no function at this level."
End Response
```

The message "That key has no function at this level" is displayed in the message panel.

- 5 This FIELD DEFAULT application specifies that MENU_DEFAULTS be applied to all fields on CAUTION_BOX_PANEL. MENU_DEFAULTS specifies that fields be displayed in reverse video when they are active.
- 6 This LITERAL declaration draws the box around the caution.
- 7 This LITERAL TEXT declaration displays "Quit Caution Box" at line 1, column 2—the label for the box.
- 8 This LITERAL TEXT declaration displays "Modifications made during this application" at line 3, column 4. This is the first line of the caution box question.
- 9 This LITERAL TEXT declaration displays "session will be discarded. Do you really" at line 4, column 5. This is the second line of the caution box question.

- 10 This LITERAL TEXT declaration displays "wish to quit?" at line 5, column 19. This is the last line of the caution box question.
- 11 An icon, QUIT_YES_BUTTON, is specified.
- 12 The SELECT function response is specified for the QUIT_YES_BUTTON icon. The SELECT function response, invoked when the Select key or the keypad period key is pressed, is the QUIT_APPLICATION internal response.
- 13 QUIT_APPLICATION was specified earlier in the form, as follows:

```
Internal Response QUIT_APPLICATION
Signal
Message
    "Quitting application. Data discarded."
Return Immediate
End Response
```

QUIT_APPLICATION gives the user the "Quitting application. Data discarded" message, and quits the application, bypassing any validation.

- 14 This LITERAL TEXT declaration outputs a "Yes" at line 9, column 17, in the middle of the box drawn in 15.
- 15 This LITERAL declaration draws the box around the "Yes".
- 16 This LITERAL declaration draws the box around the "No", specified in 18.
- 17 An icon, QUIT_NO_BUTTON, is specified.
- 18 The SELECT function response is specified for the QUIT_NO_BUTTON icon. The SELECT function response, invoked when the Select key or the keypad period key is pressed, is the CANCEL_QUIT internal response.

CANCEL_QUIT was specified earlier in the form, as follows:

```
Internal Response CANCEL_QUIT
Signal
Message
    "Quit function cancelled."
Deactivate
    Panel CAUTION_BOX_PANEL
Remove
    CAUTION_BOX_VIEWPORT
Position To Previous Item
End Response
```

CANCEL_QUIT gives the user the message "Quit function cancelled.", then deactivates the panel and positions the cursor to the previous item.

- 19 This LITERAL TEXT declaration outputs a "No" at line 9, column 30, in the middle of the box drawn in 16.

8.3 File Selection Dialog Box

The file selection dialog box allows a user to select a file name within your application. File selection boxes contain file filter text entry fields, selection text entry fields, list boxes and list box labels, and Filter, OK, and Cancel push buttons, as shown in Figure 5-11.

Example 8-3 shows the IFDL code used to create Figure 5-11.

Example 8-3 File Selection Dialog Box IFDL Code Example

```

Form TAFR_FORM
.
.
.
    Group FILEBOX_LIST                                1
        Occurs 64                                     2
        current current_file_index                    3
        FILE_SPEC Character(48)                       4
    End Group
    FILEBOX_LIST_TOP_INDICATOR Character(1)           5
    FILEBOX_LIST_BOTTOM_INDICATOR Character(1)        6
    FILEBOX_LIST_TOP_CONTROL Unsigned Longword        7
    FILEBOX_FILTER_FIELD Character(48)                8
        Value " "                                     "
    FILEBOX_SELECTION_FIELD Character(48)             9
End Data
Form Record FILEBOX_LIST_RECORD                      10
    Group FILEBOX_LIST
        Occurs 64
        FILE_SPEC Character(48)
    End Group
End Record
.
.
.
    Panel PRINT_FILEBOX_PANEL                          11
        Viewport PRINT_FILEBOX_VIEWPORT              12
        Function Response TRANSMIT                    13
            If (FILEBOX_SELECTION_FIELD = " ") Then
                Signal
                Message "You must make a selection."

```

(continued on next page)

Example 8–3 (Cont.) File Selection Dialog Box IFDL Code Example

```
        Position To Field FILEBOX_SELECTION_FIELD on PRINT_FILEBOX_PANEL
    Else                                     14
        Deactivate
            Panel PRINT_FILEBOX_PANEL
        Remove
            PRINT_FILEBOX_VIEWPORT
        Position To Previous Item
    End if
End Response

Function Response DISCARD                   15
    Deactivate
        Panel PRINT_FILEBOX_PANEL
    Remove
        PRINT_FILEBOX_VIEWPORT
    Position To Previous Item
End Response

Literal Rectangle                           16
    Line 1 Column 1
    Line 16 Column 68
End Literal

Literal Text                                17
    Line 1
    Column 2
    Value " Print to File                    "_
        " "
    Display
        Reverse
End Literal

Literal Text                                18
    Line 3
    Column 3
    Value "Filter:"
End Literal

Field FILEBOX_FILTER_FIELD                 19
    Line 3
    Column 11

    Function Response SELECT                 20
        Include DO_FILEBOX_FILTER
    End Response

    Exit Response                           21
        Include DO_FILEBOX_FILTER
    End Response
End Field
```

(continued on next page)

Example 8–3 (Cont.) File Selection Dialog Box IFDL Code Example

```
Group FILEBOX_LIST                                22
  Vertical
    Displays 8 First FIRST_FILE_INDEX
  Field FILE_SPEC                                  23
    Line 5
    Column 4

    Entry Response                                24
      Include COMPUTE_UNSEEN_FILE_COUNT
    End Response

    Function Response MOVE_DOWN                    25
      If FILEBOX_LIST(current_file_index+1).FILE_SPEC = " " Then
        Signal
        Message "No more files."
      Else
        Position To Down Occurrence
      End If
    End Response

    Function Response MOVE_UP                      26
      Position To Up Occurrence
    End Response

    Function Response MOVE_LEFT                    27
      Message "No items in that direction."
      Signal
    End Response

    Function Response MOVE_RIGHT                   28
      Position To Icon FILEBOX_OK_BUTTON On PRINT_FILEBOX_PANEL
    End Response

    Function Response SELECT                       29
      Let FILEBOX_SELECTION_FIELD =
        FILEBOX_LIST(current_file_index).FILE_SPEC
    End Response

    Function Response JUMP_UP                      30
      Position To Field FILEBOX_FILTER_FIELD On PRINT_FILEBOX_PANEL
    End Response

    Function Response JUMP_DOWN                    31
      Position To Field FILEBOX_SELECTION_FIELD On PRINT_FILEBOX_PANEL
    End Response

    No Data Input
    Active Highlight Reverse
  End Field
```

(continued on next page)

Example 8–3 (Cont.) File Selection Dialog Box IFDL Code Example

```
End Group
Literal Text                                     32
    Line 5
    Column 52
    Value " "
End Literal
Field FILEBOX_LIST_TOP_INDICATOR                33
    Line 5
    Column 53 Protected
    Display
        Character Set Private_Rule
    Output "w" When FIRST_FILE_INDEX = 1
    Output ":" When FIRST_FILE_INDEX <> 1
End Field
Field FILEBOX_LIST_BOTTOM_INDICATOR            34
    Line 12
    Column 53 Protected
    Display
        Character Set Private_Rule
    Output "v" When UNSEEN_FILE_COUNT <= 0
    Output ":" When UNSEEN_FILE_COUNT > 0
End Field

Literal Polyline                                35
    Line 6 Column 53
    Line 11 Column 53
End Literal

Literal Rectangle                              36
    Line 4 Column 3
    Line 13 Column 54
End Literal

Literal Text                                   37
    Line 15
    Column 3
    Value "Selection: "
End Literal

Literal Rectangle                              38
    Line 7 Column 57
    Line 9 Column 66
End Literal
```

(continued on next page)

Example 8-3 (Cont.) File Selection Dialog Box IFDL Code Example

```
Field FILEBOX_SELECTION_FIELD                39
  Line 15
  Column 14
End Field

Icon FILEBOX_FILTER_BUTTON                  40
  Literal Rectangle
    Line 4 Column 57
    Line 6 Column 66
  End Literal

  Literal Text
    Line 5
    Column 58
    Value " Filter "
  End Literal

End Icon

Icon FILEBOX_OK_BUTTON                      41
  Function Response SELECT                  42
    If (FILEBOX_SELECTION_FIELD = " ") Then
      Signal
      Message "You must make a selection."
      Position To Field FILEBOX_SELECTION_FIELD On PRINT_FILEBOX_PANEL
    Else
      Deactivate
      Panel PRINT_FILEBOX_PANEL
      Remove
      PRINT_FILEBOX_VIEWPORT
      Position To Previous Item
    End If
  End Response

  Active Highlight
  Reverse

  Literal Text
    Line 8
    Column 58
    Value " OK "                            43
  End Literal

End Icon
```

(continued on next page)

Example 8–3 (Cont.) File Selection Dialog Box IFDL Code Example

```
Icon FILEBOX_CANCEL_BUTTON          44
  Function Response SELECT           45
    Deactivate
      Panel PRINT_FILEBOX_PANEL
    Remove
      PRINT_FILEBOX_VIEWPORT
  End Response

  Literal Rectangle                  46
    Line 10 Column 57
    Line 12 Column 66
  End Literal

  Literal Text                       47
    Line 11
    Column 58
    Value " Cancel "
  End Literal

End Icon

End Panel

End Layout
```

- 1 A form data group, `FILEBOX_LIST`, is declared.
- 2 `FILEBOX_LIST` is an array of 64 form data items.
- 3 The `CURRENT` clause specifies the variable `CURRENT_FILE_INDEX` to contain the current index of the form data item where the cursor is positioned. If the cursor is positioned to `FILEBOX_LIST(31).FILE_SPEC`, then `CURRENT_FILE_INDEX` is 31.
- 4 The first item in `FILEBOX_LIST` is a form data item named `FILE_SPEC`. `FILE_SPEC` has a `CHARACTER` data type, and is 48 characters long.
- 5 A form data item, `FILEBOX_LIST_TOP_INDICATOR`, is declared to have a `CHARACTER` data type, and to be one character long.
- 6 A form data item, `FILEBOX_LIST_BOTTOM_INDICATOR`, is declared to have a `CHARACTER` data type, and to be one character long.
- 7 A form data item, `FILEBOX_LIST_TOP_CONTROL`, is declared to have a `UNSIGNED LONGWORD` data type.
- 8 A form data item, `FILEBOX_FILTER_FIELD`, is declared to have a `CHARACTER` data type, and to be 48 characters long. The value of `FILEBOX_FILTER_FIELD` is set to blanks.

- 9 A form data item, `FILEBOX_SELECTION_FIELD`, is declared to have a `CHARACTER` data type, and to be 48 characters long.
- 10 A form record, `FILEBOX_LIST_RECORD`, is declared. This form record contains a `FILEBOX_LIST` record group that corresponds to the `FILEBOX_LIST` data group.
- 11 A panel, `PRINT_FILEBOX_PANEL`, is declared.
- 12 The viewport in which `PRINT_FILEBOX_PANEL` is displayed is `PRINT_FILEBOX_VIEWPORT`. `PRINT_FILEBOX_VIEWPORT` was specified earlier in the form, as follows:

```
Viewport PRINT_FILEBOX_VIEWPORT
  Lines 5 Through 20
  Columns 7 Through 74
End Viewport
```

- 13 If the F10 key is pressed while the cursor is in the panel, and the value in `FILEBOX_SELECTION_FIELD` is set to blanks, a message saying "You must make a selection" is displayed, and the cursor is positioned to the `FILEBOX_SELECTION_FIELD` on `PRINT_FILEBOX_PANEL`.
- 14 If the value in `FILEBOX_SELECTION_FIELD` is not set to blanks, the `PRINT_FILEBOX_PANEL` is deactivated and removed from the display. (Removing the viewport accomplishes this.) The cursor is positioned to the previous item. (Printing is not yet implemented.)
- 15 The `DISCARD` function response specifies that the `PRINT_FILEBOX_PANEL` is deactivated and removed from the display when the F8 key or the PF1-Q key sequence is pressed.
- 16 This `LITERAL` declaration specifies the rectangle that is drawn around the file box.
- 17 This `LITERAL TEXT` declaration places the phrase "Print to File" at line 1, column 2, in the file box. The text is displayed in reverse video.
- 18 This `LITERAL TEXT` declaration places the phrase "Filter:" at line 3, column 3, in the file box.
- 19 The `FILEBOX_FILTER_FIELD` is specified at line 3, column 11. (This places the field next to the label, "Filter:".)
- 20 The `SELECT` function response is declared for the field. If the Select key is pressed while the field is active, the internal response `DO_FILEBOX_FILTER` occurs.

- 21 This EXIT RESPONSE specifies that the internal response DO_FILEBOX_FILTER occurs when the field is exited. DO_FILEBOX_FILTER was specified earlier in the form, as follows:

```
Internal Response DO_FILEBOX_FILTER
  Call "FILL_FILEBOX" Using
    By Reference FILEBOX_FILTER_FIELD
    By Reference FILEBOX_LIST_RECORD
    By Value FILE_SPEC_LENGTH
    By Value FILE_ARRAY_SIZE
  Giving FILE_COUNT
  Message FILE_COUNT " files located."
  Let FIRST_FILE_INDEX = 1
  Include COMPUTE_UNSEEN_FILE_COUNT
end response
```

DO_FILEBOX_FILTER makes a call to a procedural escape; "fill_filebox" updates the file index by one and then returns to the program.

- 22 The panel group FILEBOX_LIST is a vertical group (it scrolls) and it displays eight fields at a time.
- 23 The FILE_SPEC field is displayed at line 5, column 4. The contents of FILE_SPEC are the data items in 4.
- 24 This entry response specifies that an internal response, COMPUTE_UNSEEN_FILE_COUNT, occur when the field is entered. COMPUTE_UNSEEN_FILE_COUNT was declared earlier in the form, as follows:

```
Internal Response COMPUTE_UNSEEN_FILE_COUNT
  Let UNSEEN_FILE_COUNT = FILE_COUNT - FILE_LIST_WINDOW_SIZE -
    FIRST_FILE_INDEX + 1
End Response
```

This response computes a value, UNSEEN_FILE_COUNT.

- 25 The MOVE_DOWN function response is specified for the panel group. If the down arrow key is pressed in the panel, and the value in FILEBOX_LIST(current_file_index+1).FILE_SPEC equals blank, a message saying "No more files" is displayed. Otherwise, the cursor moves to the next position down.
- 26 The MOVE_UP function response is specified for the panel group. If the up arrow key is pressed in the panel, the cursor moves to the next item up.
- 27 The MOVE_LEFT function response is specified for the panel group. If the left arrow key is pressed in the panel, a message saying "No more items in that direction" is displayed.

- 28 The MOVE_RIGHT function response is specified for the panel group. If the right arrow key is pressed in the panel, the cursor is positioned to the OK button on PRINT_FILEBOX_PANEL.
- 29 The SELECT function response is declared for the group. If the Select key is pressed while the cursor is on the icon, the value of FILEBOX_SELECTION_FIELD is set to FILEBOX_LIST(current_file_index).FILE_SPEC.
- 30 The JUMP_UP function response is specified for the group. The JUMP_UP function was specified earlier in the form, as follows:

```
Function JUMP_UP
  Is (%PF1 %UP)
End Function
```

When the PF1 and up arrow keys are pressed, the cursor is positioned to the file box filter field on PRINT_FILEBOX_PANEL.

- 31 The JUMP_DOWN function response is specified for the group. The JUMP_DOWN function was specified earlier in the form, as follows:

```
Function JUMP_DOWN
  Is (%PF1 %DOWN)
End Function
```

When the PF1 and up arrow keys are pressed, the cursor is positioned to the file box selection field on PRINT_FILEBOX_PANEL.

- 32 This LITERAL TEXT declaration puts a blank space at line 5, column 52.
- 33 The FILEBOX_LIST_TOP_INDICATOR field is declared. It is displayed at line 5, column 53, in the PRIVATE_RULE character set. The indicator is displayed as a T (⌈) when the value in FIRST_FILE_INDEX is equal to 1, and as a colon (:) when the value in FIRST_FILE_INDEX is not equal to 1.
- 34 The FILEBOX_LIST_BOTTOM_INDICATOR field is declared. It is displayed at line 12, column 53, in the PRIVATE_RULE character set. The indicator is displayed as a reversed T (⌋) when the value in UNSEEN_FILE_COUNT is less than or equal to 0, and as a colon (:) when the value in UNSEEN_FILE_COUNT is greater than 0.
- 35 This LITERAL declaration specifies the line between the top indicator (⌈) and the bottom indicator (⌋.)
- 36 This LITERAL declaration specifies the rectangle around the list indicator.
- 37 This LITERAL TEXT declaration puts the Selection label in the File Filter box.

- 38 This LITERAL declaration puts the box around the selection field.
- 39 The FILEBOX_SELECTION_FIELD field is declared. It is displayed at line 15, column 14.
- 40 The FILEBOX_FILTER_BUTTON icon is specified. A rectangle is placed on the icon with the LITERAL declaration, and the button is labeled "Filter" by the LITERAL TEXT declaration.
- 41 The FILEBOX_OK_BUTTON icon is specified.
- 42 If the Select key is pressed while the cursor is on the icon, and the value of FILEBOX_SELECTION_FIELD is blank, the message, "You must make a selection" is displayed and the cursor is positioned to the FILEBOX_SELECTION_FIELD on PRINT_FILEBOX_PANEL. Otherwise, the PRINT_FILEBOX_PANEL is deactivated and removed from the display. (The REMOVE response step does this.)
- 43 This LITERAL TEXT declaration places the OK on the icon.
- 44 The FILEBOX_CANCEL_BUTTON icon is specified.
- 45 If the Select key is pressed while the cursor is on the icon, PRINT_FILEBOX_PANEL is deactivated and removed from the display. (The REMOVE response step does this by removing the viewport in which PRINT_FILEBOX_PANEL is displayed.)
- 46 This LITERAL declaration places a rectangle around the word "Cancel".
- 47 This LITERAL TEXT declaration places the Cancel within the rectangle on the icon.

A

Track and Field Registration Form

The IFDL that creates the Track and Field Registration form follows.

Form TAFR_FORM

Form Data

```
MODIFIED Unsigned Byte
  Value 0
N1 Unsigned Longword
N2 Unsigned Longword
N3 Unsigned Longword
COUNTRY_OPTIONS Character(13)
Group PRINT_FORMAT_RADIOBOX
  Occurs 3
  Current N1
  TOGGLE Unsigned Byte
  TAG Character(18)
End Group
Group COUNTRY_LIST_OPTIONS
  Occurs 12
  Current N2
  STRING Character(13)
  CONTROL Unsigned Byte
End Group
FILE_ENTRY_CONTROL Unsigned Byte
REGISTRATION_ENTRY_CONTROL Unsigned Byte
SCHEDULE_ENTRY_CONTROL Unsigned Byte
OPTIONS_ENTRY_CONTROL Unsigned Byte
HELP_ENTRY_CONTROL Unsigned Byte
LIST_ENTRY_CONTROL Unsigned Byte
COACHING_ENABLE Unsigned Byte
Group LANGUAGE_RADIOBOX
  Occurs 3
  Current N3
  TOGGLE Unsigned Byte
  TAG Character(21)
  PROTECT Unsigned Byte
End Group
REGISTRATION_NUMBER Integer(9)
FIRST_NAME Character(32)
LAST_NAME Character(32)
```

```

COUNTRY Character(13)
CALL_FROM Character(16)
TOGGLE_SHOT_PUT Unsigned Byte
TOGGLE_HIGH_JUMP Unsigned Byte
TOGGLE_JAVELIN Unsigned Byte
TOGGLE_POLE_VAULT Unsigned Byte
TOGGLE_DISCUS Unsigned Byte
TOGGLE_LONG_JUMP Unsigned Byte
TOGGLE_100_METER Unsigned Byte
TOGGLE_400_METER Unsigned Byte
TOGGLE_5000_METER Unsigned Byte
TOGGLE_10000_METER Unsigned Byte
TOGGLE_4X4_RELAY Unsigned Byte
WORKING_DELAY Unsigned Longword
    Value 5
FIRST_FILE_INDEX Longword Integer
CURRENT_FILE_INDEX Longword Integer
FILE_COUNT Longword Integer
UNSEEN_FILE_COUNT Longword Integer
FILE_ARRAY_SIZE Longword Integer value 64
FILE_SPEC_LENGTH Longword Integer value 48
FILE_LIST_WINDOW_SIZE Longword Integer value 8
Group FILEBOX_LIST
    Occurs 64
    Current CURRENT_FILE_INDEX
    FILE_SPEC Character(48)
End Group
FILEBOX_LIST_TOP_INDICATOR Character(1)
FILEBOX_LIST_BOTTOM_INDICATOR Character(1)
FILEBOX_LIST_TOP_CONTROL Unsigned Longword
FILEBOX_FILTER_FIELD Character(48)
    Value "
FILEBOX_SELECTION_FIELD Character(48)
End Data

Form Record FILEBOX_LIST_RECORD
    Group FILEBOX_LIST
        Occurs 64
        FILE_SPEC Character(48)
    End Group
End Record

Layout VT_LAYOUT
    Device
        Terminal
            Type %VT100
    End Device
    Size 24 Lines by 80 Columns

    Viewport BANNER_VIEWPORT
        Lines 1 Through 1
        Columns 1 Through 80
    End Viewport

```

Viewport BAR_MENU_VIEWPORT
Lines 2 Through 2
Columns 1 Through 80
End Viewport

Viewport MESSAGE_VIEWPORT
Lines 23 Through 24
Columns 1 Through 80
End Viewport

Viewport FILE_PULLDOWN_VIEWPORT
Lines 3 Through 6
Columns 1 Through 8
End Viewport

Viewport REGISTRATION_PULLDOWN_VIEWPORT
Lines 3 Through 9
Columns 10 Through 42
End Viewport

Viewport LIST_PULLDOWN_VIEWPORT
Lines 7 Through 12
Columns 41 Through 54
End Viewport

Viewport CAUTION_BOX_VIEWPORT
Lines 7 Through 18
Columns 16 Through 64
End Viewport

Viewport COUNTRY_LIST_DIALOG_VIEWPORT
Lines 6 Through 19
Columns 25 Through 55
End Viewport

Viewport COUNTRY_LIST_OPTION_VIEWPORT
Lines 2 Through 15
Columns 40 Through 56
End Viewport

Viewport OPTIONS_PULLDOWN_VIEWPORT
Lines 3 Through 16
Columns 35 Through 63
End Viewport

Viewport ADD_REGISTRATION_VIEWPORT
Lines 2 Through 20
Columns 11 Through 67
End Viewport

Viewport WORK_IN_PROGRESS_VIEWPORT
Lines 8 Through 12
Columns 30 Through 49
End Viewport

```

Viewport PRINT_FILEBOX_VIEWPORT
  Lines 5 Through 20
  Columns 7 Through 74
End Viewport

Function SELECT
  Is %SELECT
  %KP_PERIOD
End Function

Function SPECIAL_FUNCTION
  Is %DO
End Function

Function MOVE_UP
  Is %UP
End Function

Function MOVE_DOWN
  Is %DOWN
End Function

Function MOVE_LEFT
  Is %LEFT
End Function

Function MOVE_RIGHT
  Is %RIGHT
End Function

Function JUMP_UP
  Is (%PF1 %UP)
End Function

Function JUMP_DOWN
  Is (%PF1 %DOWN)
End Function

Function JUMP_LEFT
  Is (%PF1 %LEFT)
End Function

Function JUMP_RIGHT
  Is (%PF1 %RIGHT)
End Function

Function DISCARD
  Is %F8
  (%PF1 %CAPITAL_Q)
  (%PF1 %SMALL_Q)
End Function

Function NEXT_SCREEN
  Is %NEXT_SCREEN
End Function

```

```

Function PREV_SCREEN
  Is %PREV_SCREEN
End Function

Function ADD_REGISTRATION
  Is %CONTROL_A
End Function

Function MAGIC
  Is %F20
End Function

Function MNEMONIC_A
  Is (%PF4 %CAPITAL_A)
  (%PF4 %SMALL_A)
End Function

Function MNEMONIC_C
  Is (%PF4 %CAPITAL_C)
  (%PF4 %SMALL_C)
End Function

Function MNEMONIC_D
  Is (%PF4 %CAPITAL_D)
  (%PF4 %SMALL_D)
End Function

Function MNEMONIC_E
  Is (%PF4 %CAPITAL_E)
  (%PF4 %SMALL_E)
End Function

Function MNEMONIC_F
  Is (%PF4 %CAPITAL_F)
  (%PF4 %SMALL_F)
End Function

Function MNEMONIC_L
  Is (%PF4 %CAPITAL_L)
  (%PF4 %SMALL_L)
End Function

Function MNEMONIC_O
  Is (%PF4 %CAPITAL_O)
  (%PF4 %SMALL_O)
End Function

Function MNEMONIC_Q
  Is (%PF4 %CAPITAL_Q)
  (%PF4 %SMALL_Q)
End Function

Function MNEMONIC_R
  Is (%PF4 %CAPITAL_R)
  (%PF4 %SMALL_R)
End Function

```

```

Function MNEMONIC_S
  Is (%PF4 %CAPITAL_S)
    (%PF4 %SMALL_S)
End Function

Internal Response NO_FUNCTION
  Signal
  Message
    "That key has no function at this level."
End Response

Internal Response BORDER_PATROL
  Signal
  Message
    "There are no items in that direction."
End Response

Internal Response QUIT_APPLICATION
  Signal
  Message
    "Quitting application. Data discarded."
  Return Immediate
End Response

Internal Response CANCEL_QUIT
  Signal
  Message
    "Quit function cancelled."
  Deactivate
    Panel CAUTION_BOX_PANEL
  Remove
    CAUTION_BOX_VIEWPORT
  Position To Previous Item
End Response

Internal Response EXIT_APPLICATION
  Message
    "Exiting application..."
  Return
End Response

Internal Response COMPUTE_UNSEEN_FILE_COUNT
  Let UNSEEN_FILE_COUNT = FILE_COUNT - FILE_LIST_WINDOW_SIZE -
    FIRST_FILE_INDEX + 1
End Response

```

```

Internal Response DO_FILEBOX_FILTER
  Call "FILL_FILEBOX" Using
    By Reference FILEBOX_FILTER_FIELD
    By Reference FILEBOX_LIST_RECORD
    By Value FILE_SPEC_LENGTH
    By Value FILE_ARRAY_SIZE
    Giving FILE_COUNT
  Message FILE_COUNT " files located."
  Let FIRST_FILE_INDEX = 1
  Include COMPUTE_UNSEEN_FILE_COUNT
End Response

Internal Response PERFORM_LIST_FUNCTION
  If (PRINT_FORMAT_RADIOBOX(1).TOGGLE = 1) Then
    Signal
    Message
      "DISPLAY SCREEN LIST function not yet implemented."
    Position to previous item
  End If
  If (PRINT_FORMAT_RADIOBOX(2).TOGGLE = 1) Then
    Message
      "Printing list..."
    Display
      WORK_IN_PROGRESS_BOX
    Call "pause_interface" Using
      By Value WORKING_DELAY
    Remove
      WORK_IN_PROGRESS_VIEWPORT
    Message
      "List printed."
    Signal
    Position To Previous Item
  End If
  If (PRINT_FORMAT_RADIOBOX(3).TOGGLE = 1) Then
    Activate
      Panel PRINT_FILEBOX_PANEL
    Position To Panel PRINT_FILEBOX_PANEL
    If (FILEBOX_FILTER_FIELD = " ") Then
      Let FILEBOX_FILTER_FIELD = " *.*"
      Include DO_FILEBOX_FILTER
    End If
  End If
End Response

```

```

Enable Response
  Let PRINT_FORMAT_RADIOBOX(1).TAG = " Display on Screen"
  Let PRINT_FORMAT_RADIOBOX(2).TAG = " Print List"
  Let PRINT_FORMAT_RADIOBOX(3).TAG = " Print to File"
  Let COUNTRY_LIST_OPTIONS(1).STRING = "Australia"
  Let COUNTRY_LIST_OPTIONS(2).STRING = "Canada"
  Let COUNTRY_LIST_OPTIONS(3).STRING = "China"
  Let COUNTRY_LIST_OPTIONS(4).STRING = "Denmark"
  Let COUNTRY_LIST_OPTIONS(5).STRING = "Egypt"
  Let COUNTRY_LIST_OPTIONS(6).STRING = "England"
  Let COUNTRY_LIST_OPTIONS(7).STRING = "France"
  Let COUNTRY_LIST_OPTIONS(8).STRING = "Germany"
  Let COUNTRY_LIST_OPTIONS(9).STRING = "Japan"
  Let COUNTRY_LIST_OPTIONS(10).STRING = "Sweden"
  Let COUNTRY_LIST_OPTIONS(11).STRING = "United States"
  Let COUNTRY_LIST_OPTIONS(12).STRING = "Soviet Union"
  Let LANGUAGE_RADIOBOX(1).TAG = " English"
  Let LANGUAGE_RADIOBOX(2).TAG = " French"
  Let LANGUAGE_RADIOBOX(3).TAG = " Hebrew"
  Display
    BANNER_PANEL
  Activate
    Panel BAR_MENU_PANEL
  Position To Panel BAR_MENU_PANEL
  Let MODIFIED = 1
End Response

Function Response MOVE_UP
  If ( NOT UPPERMOST ITEM) Then
    Position To Up Item
  Else
    Include BORDER_PATROL
  End If
End Response

Function Response MOVE_DOWN
  If ( NOT LOWERMOST ITEM) Then
    Position To Down Item
  Else
    Include BORDER_PATROL
  End If
End Response

Function Response MOVE_LEFT
  If ( NOT LEFTMOST ITEM) Then
    Position To Left Item
  Else
    Include BORDER_PATROL
  End If
End Response

```



```

Function Response MOVE_RIGHT
  If ( NOT RIGHTMOST ITEM) Then
    Position To Right Item
  Else
    Include BORDER_PATROL
  End If
End Response

Function Response NEXT ITEM
  If ( NOT PANEL LAST ITEM) Then
    Position To Next Item
  Else
    Include BORDER_PATROL
  End If
End Response

Function Response PREVIOUS ITEM
  If ( NOT PANEL FIRST ITEM) Then
    Position To Previous Item
  Else
    Include BORDER_PATROL
  End If
End Response

Function Response TRANSMIT
  Include EXIT_APPLICATION
End Response

Function Response DISCARD
  If (MODIFIED = 1) Then
    Activate
    Panel CAUTION_BOX_PANEL
    Position To Icon QUIT_NO_BUTTON On CAUTION_BOX_PANEL
  Else
    Include QUIT_APPLICATION
  End If
End Response

Function Response ADD_REGISTRATION
  Activate
  Panel ADD_REGISTRATION_DIALOG
  Position To Panel ADD_REGISTRATION_DIALOG
End Response

Function Response SELECT
  Signal
  Message
  "Feature not yet implemented."
End Response

Field Default BUTTON_DEFAULTS
  Active Highlight
  Reverse
End Default

```

```

Field Default MENU_DEFAULTS
  Active Highlight
  Reverse
End Default

Field Default TEXT_DEFAULTS
  Active Highlight
  Underlined
  Display
  Underlined
End Default

Message Panel MESSAGE_PANEL
  Viewport MESSAGE_VIEWPORT
End Panel

Panel BANNER_PANEL
  Viewport BANNER_VIEWPORT
  Literal Text
  Line 1
  Column 1
  Value " Track and Field Registration V1.0      "
  Display
  Font Size Double Wide
  Underlined
  End Literal
End Panel

Panel BAR_MENU_PANEL
  Viewport BAR_MENU_VIEWPORT
  Function Response DISCARD
  If (MODIFIED = 1) Then
    Activate
    Panel CAUTION_BOX_PANEL
    Position To Icon QUIT_NO_BUTTON On CAUTION_BOX_PANEL
  Else
    Include QUIT_APPLICATION
  End If
End Response

Function Response MNEMONIC_F
  Let FILE_ENTRY_CONTROL = 1
  Activate
  Panel FILE_PULLDOWN_PANEL
  Position To Panel FILE_PULLDOWN_PANEL
End Response

Function Response MNEMONIC_R
  Let REGISTRATION_ENTRY_CONTROL = 1
  Activate
  Panel REGISTRATION_PULLDOWN_PANEL
  Position To Panel REGISTRATION_PULLDOWN_PANEL
End Response

```

```

Apply Field Default MENU_DEFAULTS
Icon FILE_CASCADE_BUTTON
  Function Response SELECT
    Let FILE_ENTRY_CONTROL = 1
    Activate
      Panel FILE_PULLDOWN_PANEL
      Position To Panel FILE_PULLDOWN_PANEL
    End Response

  Highlight
    Reverse
    When (FILE_ENTRY_CONTROL = 1)
  Literal Text
    Line 1
    Column 2
    Value " File "
    Display
      Underlined
  End Literal
End Icon

Icon REGISTRATION_CASCADE_BUTTON
  Function Response SELECT
    Let REGISTRATION_ENTRY_CONTROL = 1
    Activate
      Panel REGISTRATION_PULLDOWN_PANEL
      Position To Panel REGISTRATION_PULLDOWN_PANEL
    End Response

  Highlight
    Reverse
    When (REGISTRATION_ENTRY_CONTROL = 1)
  Literal Text
    Line 1
    Column 10
    Value " Registration "
    Display
      Underlined
  End Literal
End Icon

Icon SCHEDULE_CASCADE_BUTTON
  Function Response SELECT
    Signal
    Message
      "SCHEDULE menu not yet implemented."
  End Response

```

```

Highlight
    Reverse
    When (SCHEDULE_ENTRY_CONTROL = 1)
Literal Text
    Line 1
    Column 25
    Value " Schedule "
    Display
        Underlined
End Literal
End Icon
Icon OPTIONS_CASCADE_BUTTON
    Function Response SELECT
        Let OPTIONS_ENTRY_CONTROL = 1
        Activate
            Panel OPTIONS_PULLDOWN_PANEL
            Position To Panel OPTIONS_PULLDOWN_PANEL
        End Response
    Highlight
        Reverse
        When (OPTIONS_ENTRY_CONTROL = 1)
    Literal Text
        Line 1
        Column 36
        Value " Options "
        Display
            Underlined
    End Literal
End Icon
Icon HELP_CASCADE_BUTTON
    Highlight
        Reverse
        When (HELP_ENTRY_CONTROL = 1)
    Literal Text
        Line 1
        Column 74
        Value " Help "
        Display
            Underlined
    End Literal
End Icon
Literal Text
    Line 1
    Column 1
    Value " "
End Literal

```

```

Literal Text
  Line 1
  Column 1
  Value "
      "
  Display
    Underlined
End Literal

End Panel

Panel FILE_PULLDOWN_PANEL
  Viewport FILE_PULLDOWN_VIEWPORT
  Function Response DISCARD
    Deactivate
      Panel FILE_PULLDOWN_PANEL
    Remove
      FILE_PULLDOWN_VIEWPORT
    Position To Previous Item
    Let FILE_ENTRY_CONTROL = 0
  End Response

  Function Response MNEMONIC_E
    Let FILE_ENTRY_CONTROL = 0
    Include EXIT_APPLICATION
  End Response

  Function Response MNEMONIC_Q
    If (MODIFIED = 1) Then
      Activate
        Panel CAUTION_BOX_PANEL
        Position To Icon QUIT_NO_BUTTON On CAUTION_BOX_PANEL
    Else
      Include QUIT_APPLICATION
    End If
    Deactivate
      Panel FILE_PULLDOWN_PANEL
    Remove
      FILE_PULLDOWN_VIEWPORT
    Let FILE_ENTRY_CONTROL = 0
  End Response

  Apply Field Default MENU_DEFAULTS
  Literal Rectangle
    Line 1 Column 1
    Line 4 Column 8
  End Literal

  Icon EXIT_ENTRY
    Function Response SELECT
      Let FILE_ENTRY_CONTROL = 0
      Include EXIT_APPLICATION
    End Response

```

```

Literal Text
  Line 2
  Column 2
  Value " Exit "
End Literal

Literal Text
  Line 2
  Column 3
  Value "E"
  Display
    Underlined
End Literal

End Icon

Icon QUIT_ENTRY
  Function Response SELECT
    If (MODIFIED = 1) Then
      Activate
        Panel CAUTION_BOX_PANEL
        Position To Icon QUIT_NO_BUTTON On CAUTION_BOX_PANEL
    Else
      Include QUIT_APPLICATION
    End If
    Deactivate
      Panel FILE_PULLDOWN_PANEL
    Remove
      FILE_PULLDOWN_VIEWPORT
    Let FILE_ENTRY_CONTROL = 0
  End Response

  Literal Text
    Line 3
    Column 2
    Value " Quit "
  End Literal

  Literal Text
    Line 3
    Column 3
    Value "Q"
    Display
      Underlined
  End Literal

End Icon

End Panel

Panel CAUTION_BOX_PANEL
  Viewport CAUTION_BOX_VIEWPORT
  Function Response TRANSMIT
    Include CANCEL_QUIT
  End Response

```

```

Function Response DISCARD
  Include NO_FUNCTION
End Response

Apply Field Default MENU_DEFAULTS
Literal Rectangle
  Line 1 Column 1
  Line 12 Column 49
End Literal

Literal Text
  Line 1
  Column 2
  Value " Quit Caution Box
  Display
  Reverse
End Literal

Literal Text
  Line 3
  Column 4
  Value "Modifications made during this application"
End Literal

Literal Text
  Line 4
  Column 5
  Value "session will be discarded. Do you really"
End Literal

Literal Text
  Line 5
  Column 19
  Value "wish to quit?"
End Literal

Icon QUIT_YES_BUTTON
  Function Response SELECT
  Include QUIT_APPLICATION
  End Response

  Literal Text
  Line 9
  Column 17
  Value " Yes "
  End Literal

End Icon

Literal Rectangle
  Line 8 Column 16
  Line 10 Column 22
End Literal

```

```

Literal Rectangle
  Line 8 Column 29
  Line 10 Column 34
End Literal

Icon QUIT_NO_BUTTON
  Function Response SELECT
    Include CANCEL_QUIT
  End Response

  Literal Text
    Line 9
    Column 30
    Value " No "
  End Literal

End Icon

End Panel

Panel REGISTRATION_PULLDOWN_PANEL
  Viewport REGISTRATION_PULLDOWN_VIEWPORT
  Function Response DISCARD
    Deactivate
      Panel REGISTRATION_PULLDOWN_PANEL
    Remove
      REGISTRATION_PULLDOWN_VIEWPORT
    Position To Previous Item
    Let REGISTRATION_ENTRY_CONTROL = 0
  End Response

  Function Response MNEMONIC_A
    Deactivate
      Panel REGISTRATION_PULLDOWN_PANEL
    Remove
      REGISTRATION_PULLDOWN_VIEWPORT
    Let REGISTRATION_ENTRY_CONTROL = 0
    Activate
      Panel ADD_REGISTRATION_DIALOG
    Position To Panel ADD_REGISTRATION_DIALOG
  End Response

  Function Response MNEMONIC_C
    Signal
    Message
      "CHANGE REGISTRATION action not yet implemented."
  End Response

  Function Response MNEMONIC_D
    Signal
    Message
      "DELETE REGISTRATION action not yet implemented."
  End Response

```



```

Function Response MNEMONIC_L
  Let LIST_ENTRY_CONTROL = 1
  Activate
    Panel LIST_PULLDOWN_PANEL
  Position To Panel LIST_PULLDOWN_PANEL
End Response

Apply Field Default MENU_DEFAULTS
Literal Text
  Line 5
  Column 2
  Value "....."
End Literal

Icon ADD_ENTRY
  Function Response SELECT
    Deactivate
      Panel REGISTRATION_PULLDOWN_PANEL
    Remove
      REGISTRATION_PULLDOWN_VIEWPORT
    Let REGISTRATION_ENTRY_CONTROL = 0
    Activate
      Panel ADD_REGISTRATION_DIALOG
    Position To Panel ADD_REGISTRATION_DIALOG
  End Response

  Display
    Underlined
  Literal Text
    Line 2
    Column 2
    Value " Add Registration...  Ctrl+A"
    Display
      Nounderlined
  End Literal

  Literal Text
    Line 2
    Column 3
    Value "A"
    Display
      Underlined
  End Literal

End Icon

Icon CHANGE_ENTRY
  Function Response select
    Signal
    Message
      "CHANGE REGISTRATION action not yet implemented."
  End Response

```

```

Display
    Underlined
Literal Text
    Line 3
    Column 2
    Value " Change Registration...      "
    Display
        Nounderlined
End Literal

Literal Text
    Line 3
    Column 3
    Value "C"
    Display
        Underlined
End Literal

End Icon

Icon DELETE_ENTRY
    Function Response select
    Signal
    Message
        "DELETE REGISTRATION action not yet implemented."
    End Response

    Display
        Underlined
    Literal Text
        Line 4
        Column 2
        Value " Delete Registration...    "
        Display
            Nounderlined
    End Literal

    Literal Text
        Line 4
        Column 3
        Value "D"
        Display
            Underlined
    End Literal

End Icon

Icon LIST_CASCADE_BUTTON
    Function Response MOVE_RIGHT
    Let LIST_ENTRY_CONTROL = 1
    Activate
        Panel LIST_PULLDOWN_PANEL
    Position To Panel LIST_PULLDOWN_PANEL
    End Response

```

```

Function Response SELECT
  Let LIST_ENTRY_CONTROL = 1
  Activate
    Panel LIST_PULLDOWN_PANEL
    Position To Panel LIST_PULLDOWN_PANEL
  End Response

  Display
    Underlined
  Highlight
    Reverse
    When (LIST_ENTRY_CONTROL = 1)
  Literal Text
    Line 6
    Column 2
    Value " List Registrants      ->"
    Display
      Nounderlined
  End Literal

  Literal Text
    Line 6
    Column 3
    Value "L"
    Display
      Underlined
  End Literal

End Icon

Literal Rectangle
  Line 1 Column 1
  Line 7 Column 33
End Literal

End Panel

Panel LIST_PULLDOWN_PANEL
  Viewport LIST_PULLDOWN_VIEWPORT
  Function Response DISCARD
    Let LIST_ENTRY_CONTROL = 0
    Deactivate
      Panel LIST_PULLDOWN_PANEL
    Remove
      LIST_PULLDOWN_VIEWPORT
    Position To Previous Item
  End Response

  Apply Field Default MENU_DEFAULTS
  Literal Rectangle
    Line 1 Column 1
    Line 6 Column 14
  End Literal

```

```

Icon BY_COUNTRY_ENTRY
  Function Response SELECT
    Deactivate
      Panel LIST_PULLDOWN_PANEL
    Remove
      LIST_PULLDOWN_VIEWPORT
    Deactivate
      Panel REGISTRATION_PULLDOWN_PANEL
    Remove
      REGISTRATION_PULLDOWN_VIEWPORT
    Activate
      Panel COUNTRY_LIST_DIALOG_PANEL
    Position To Panel COUNTRY_LIST_DIALOG_PANEL
    Let LIST_ENTRY_CONTROL = 0
    Let REGISTRATION_ENTRY_CONTROL = 0
  End Response

  Literal Text
    Line 2
    Column 2
    Value " By Country "
  End Literal

End Icon

Icon BY_EVENT_ENTRY
  Function Response select
    Signal
    Message
      "LIST BY EVENT function not yet implemented."
  End Response

  Literal Text
    Line 3
    Column 2
    Value " By Event "
  End Literal

End Icon

Icon BY_NAME_ENTRY
  Function Response select
    Signal
    Message
      "LIST BY NAME function not yet implemented."
  End Response

  Literal Text
    Line 4
    Column 2
    Value " By Name "
  End Literal

End Icon

```

```

Icon BY_NUMBER_ENTRY
  Function Response select
    Signal
    Message
      "LIST BY NUMBER function not yet implemented."
    End Response

  Literal Text
    Line 5
    Column 2
    Value " By Number "
  End Literal

End Icon

End Panel

Panel COUNTRY_LIST_DIALOG_PANEL
  Viewport COUNTRY_LIST_DIALOG_VIEWPORT
  Function Response TRANSMIT
    Deactivate
      Panel COUNTRY_LIST_DIALOG_PANEL
    Remove
      COUNTRY_LIST_DIALOG_VIEWPORT
    Include PERFORM_LIST_FUNCTION
  End Response

  Function Response DISCARD
    Deactivate
      Panel COUNTRY_LIST_DIALOG_PANEL
    Remove
      COUNTRY_LIST_DIALOG_VIEWPORT
    Position To Previous Item
  End Response

  Literal Text
    Line 4
    Column 3
    Value "Country:"
  End Literal

  Literal Rectangle
    Line 3 Column 12
    Line 5 Column 28
  End Literal

  Field COUNTRY_OPTIONS
    Line 4
    Column 13
    Function Response SELECT
      Let CALL_FROM = "LIST_REGISTRANTS"
      Activate
        Panel COUNTRY_LIST_OPTION_MENU
      Position To Panel COUNTRY_LIST_OPTION_MENU
    End Response

```

```

Active Highlight
  Reverse
Output Picture ' 'X(13)
No Data Input
Output "      ...      "
  When (COUNTRY_OPTIONS = " ")
End Field

Literal Text
  Line 6
  Column 3
  Value "Print Format:"
End Literal

Literal Text
  Line 7
  Column 3
  Value "< >"
End Literal

Literal Text
  Line 8
  Column 3
  Value "< >"
End Literal

Literal Text
  Line 9
  Column 3
  Value "< >"
End Literal

Group PRINT_FORMAT_RADIOBOX
  Vertical
  Field TOGGLE
    Line 7
    Column 4
    Display
      Character Set Private_Rule
      Output Picture X
      Output " "
      When (PRINT_FORMAT_RADIOBOX(**).TOGGLE = 0)
      Output ""
      When (PRINT_FORMAT_RADIOBOX(**).TOGGLE = 1)
      Protected
    End Field
  End Field

```

```

Field TAG
  Line 7
  Column 6
  Function Response SELECT
    Reset
      PRINT_FORMAT_RADIOBOX(*).TOGGLE
    Let PRINT_FORMAT_RADIOBOX(N1).TOGGLE = 1
  End Response

  Active Highlight
  Reverse
  No Data Input
End Field

End Group

Literal Rectangle
  Line 11 Column 8
  Line 13 Column 13
End Literal

Literal Rectangle
  Line 11 Column 15
  Line 13 Column 24
End Literal

Literal Rectangle
  Line 1 Column 1
  Line 14 Column 31
End Literal

Literal Text
  Line 1
  Column 2
  Value " List Registrants by Country "
  Display
  Reverse
End Literal

Icon LIST_DIALOG_OK_BUTTON
  Function Response SELECT
    Deactivate
      Panel COUNTRY_LIST_DIALOG_PANEL
    Remove
      COUNTRY_LIST_DIALOG_VIEWPORT
    Include PERFORM_LIST_FUNCTION
  End Response

  Active Highlight
  Reverse

```

```

Literal Text
  Line 12
  Column 9
  Value " OK "
End Literal

End Icon

Icon LIST_DIALOG_CANCEL_BUTTON
  Function Response SELECT
  Message
    "List Operation cancelled."
  Signal
  Deactivate
    Panel COUNTRY_LIST_DIALOG_PANEL
  Remove
    COUNTRY_LIST_DIALOG_VIEWPORT
  Position To Previous Item
End Response

Active Highlight
  Reverse
Literal Text
  Line 12
  Column 16
  Value " Cancel "
End Literal

End Icon

End Panel

Panel COUNTRY_LIST_OPTION_MENU
  Viewport COUNTRY_LIST_OPTION_VIEWPORT
  Apply Field Default MENU_DEFAULTS
  Group COUNTRY_LIST_OPTIONS
  Vertical
  Field STRING
    Line 2
    Column 3
    Output Picture X(14)
    Highlight
      Reverse
      When (COUNTRY_LIST_OPTIONS(**).CONTROL = 1)
    Protected
  End Field

  Field CONTROL
    Line 2
    Column 2
    Entry Response
      Let COUNTRY_LIST_OPTIONS(N2).CONTROL = 1
    End Response

```



```

Exit Response
  Let COUNTRY_LIST_OPTIONS(N2).CONTROL = 0
End Response

Function Response SELECT
  If CALL_FROM = "LIST_REGISTRANTS" Then
    Let COUNTRY_OPTIONS = COUNTRY_LIST_OPTIONS(N2).STRING
  End If

  If CALL_FROM = "ADD_REGISTRANTS " Then
    Let COUNTRY = COUNTRY_LIST_OPTIONS(N2).STRING
  End If

  Deactivate
    Panel COUNTRY_LIST_OPTION_MENU
  Remove
    COUNTRY_LIST_OPTION_VIEWPORT
End Response

Output Picture X
No Data Input
Output " "
  When 1 = 1
End Field

End Group

Literal Rectangle
  Line 1 Column 1
  Line 14 Column 17
End Literal

End Panel

Panel OPTIONS_PULLDOWN_PANEL
  Viewport OPTIONS_PULLDOWN_VIEWPORT
  Function Response MAGIC
    If (LANGUAGE_RADIOBOX(3).PROTECT = 0) Then
      Let LANGUAGE_RADIOBOX(3).PROTECT = 1
      Let LANGUAGE_RADIOBOX(3).TAG = "{Hebrew}"
    Else
      Let LANGUAGE_RADIOBOX(3).PROTECT = 0
      Let LANGUAGE_RADIOBOX(3).TAG = " Hebrew"
    End If
  End Response

  Function Response discard
    Let OPTIONS_ENTRY_CONTROL = 0
    Deactivate
      Panel OPTIONS_PULLDOWN_PANEL
    Remove
      OPTIONS_PULLDOWN_VIEWPORT
    Position To Previous Item
  End Response

```

```

Apply Field Default MENU_DEFAULTS
Icon PRINT_OPTIONS_ENTRY
  Function Response SELECT
    Signal
    Message
      "PRINT OPTIONS function not yet implemented."
    End Response
  Literal Text
    Line 2
    Column 6
    Value " Print Options...  "
  End Literal
End Icon

Icon PROCESS_OPTIONS_PULLDOWN
  Function Response SELECT
    Signal
    Message
      "PROCESS OPTIONS function not yet implemented."
    End Response
  Literal Text
    Line 3
    Column 6
    Value " Process Options  ->"
  End Literal
End Icon

Literal Text
  Line 4
  Column 2
  Value "....."
End Literal

Literal Text
  Line 5
  Column 3
  Value "[ ]"
End Literal

```

```

Field COACHING_ENABLE
  Line 5
  Column 4
  Display
    Character Set Private_Rule
  Output Picture X
  Output " "
    When (COACHING_ENABLE = 0)
  Output ""
    When (COACHING_ENABLE = 1)
  Protected
End Field

Icon COACHING_ENTRY
  Function Response SELECT
    If (COACHING_ENABLE = 0) Then
      Let COACHING_ENABLE = 1
    Else
      Let COACHING_ENABLE = 0
    End If
  End Response

  Literal Text
    Line 5
    Column 6
    Value " Coaching "
  End Literal

End Icon

Literal Text
  Line 6
  Column 2
  Value "....."
End Literal

Group LANGUAGE_RADIOBOX
  Vertical
  Literal Text
    Line 7
    Column 3
    Value "< >"
  End Literal

```

```

Field TOGGLE
  Line 7
  Column 4
  Display
    Character Set Private_Rule
  Output Picture X
  Output " "
    When (LANGUAGE_RADIOBOX(**).TOGGLE = 0)
  Output ""
    When (LANGUAGE_RADIOBOX(**).TOGGLE = 1)
  Protected
End Field

Field TAG
  Line 7
  Column 6
  Function Response SELECT
    Reset
      LANGUAGE_RADIOBOX(*).TOGGLE
    Let LANGUAGE_RADIOBOX(N3).TOGGLE = 1
  End Response

  Active Highlight
    Reverse
  Output Picture X(22)
  No Data Input
  Protected
    When (LANGUAGE_RADIOBOX(**).PROTECT = 1)
End Field

End Group

Literal Text
  Line 10
  Column 2
  Value "....."
End Literal

Icon SAVE_SETTINGS_ENTRY
  Function Response SELECT
    Signal
    Message
      "SAVE SETTINGS function not yet implemented."
  End Response

  Literal Text
    Line 11
    Column 6
    Value " Save Settings      "
  End Literal

End Icon

```

```

Icon RESTORE_SETTINGS_ENTRY
  Function Response SELECT
    Signal
    Message
      "RESTORE SETTINGS function not yet implemented."
    End Response
  Literal Text
    Line 12
    Column 6
    Value " Restore Settings      "
  End Literal
End Icon

Icon USE_DEFAULTS_ENTRY
  Function Response SELECT
    Signal
    Message
      "USE DEFAULT SETTINGS function not yet implemented."
    End Response
  Literal Text
    Line 13
    Column 6
    Value " Use System Settings  "
  End Literal
End Icon

Literal Rectangle
  Line 1 Column 1
  Line 14 Column 29
End Literal

End Panel

Panel ADD_REGISTRATION_DIALOG
  Viewport ADD_REGISTRATION_VIEWPORT
  Function Response DISCARD
    Deactivate
      Panel ADD_REGISTRATION_DIALOG
    Remove
      ADD_REGISTRATION_VIEWPORT
    Position To Previous Item
  End Response

```

```

Function Response TRANSMIT
  Message
    "Recording registrant data."
  Deactivate
    Panel ADD_REGISTRATION_DIALOG
  Remove
    ADD_REGISTRATION_VIEWPORT
  Position To Previous Item
End Response

Literal Rectangle
  Line 1 Column 1
  Line 19 Column 57
End Literal

Literal Text
  Line 1
  Column 2
  Value " Add Registration           "
  Display
    Reverse
End Literal

Literal Text
  Line 3
  Column 3
  Value "Registration Number:"
End Literal

Field REGISTRATION_NUMBER
  Line 3
  Column 24
  Apply Field Default TEXT_DEFAULTS
  Output Picture 9(9)
End Field

Literal Text
  Line 4
  Column 3
  Value "First Name:"
End Literal

Field FIRST_NAME
  Line 4
  Column 24
  Apply Field Default TEXT_DEFAULTS
End Field

Literal Text
  Line 5
  Column 3
  Value "Last Name:"
End Literal

```

```

Field LAST_NAME
  Line 5
  Column 24
  Apply Field Default TEXT_DEFAULTS
End Field

Literal Text
  Line 6
  Column 3
  Value "Country...:"
End Literal

Field COUNTRY
  Line 6
  Column 24
  Apply Field Default TEXT_DEFAULTS
  Function Response SELECT
    Let CALL_FROM = "ADD_REGISTRANTS "
    Activate
      Panel COUNTRY_LIST_OPTION_MENU
      Position To Panel COUNTRY_LIST_OPTION_MENU
    End Response
End Field

Literal Text
  Line 8
  Column 3
  Value "Events"
End Literal

Literal Text
  Line 9
  Column 3
  Value "[ ]"
End Literal

Field TOGGLE_SHOT_PUT
  Line 9
  Column 4
  Display
    Character Set Private_Rule
    Output Picture X
    Output ""
      When (TOGGLE_SHOT_PUT = 1)
    Output " "
      When (TOGGLE_SHOT_PUT <> 1)
    Protected
End Field

```

```

Icon EVENT_SHOT_PUT
  Apply Field Default BUTTON_DEFAULTS
  Function Response SELECT
    If (TOGGLE_SHOT_PUT = 0) Then
      Let TOGGLE_SHOT_PUT = 1
    Else
      Let TOGGLE_SHOT_PUT = 0
    End If
  End Response

  Literal Text
    Line 9
    Column 6
    Value " Shot Put"
  End Literal

End Icon

Literal Text
  Line 9
  Column 24
  Value "[ ]"
End Literal

Field TOGGLE_100_METER
  Line 9
  Column 25
  Display
    Character Set Private_Rule
  Output Picture X
  Output ""
    When (TOGGLE_100_METER = 1)
  Output " "
    When (TOGGLE_100_METER <> 1)
  Protected
End Field

Icon EVENT_100_METER
  Apply Field Default BUTTON_DEFAULTS
  Function Response SELECT
    If (TOGGLE_100_METER = 0) Then
      Let TOGGLE_100_METER = 1
    Else
      Let TOGGLE_100_METER = 0
    End If
  End Response

  Literal Text
    Line 9
    Column 27
    Value " 100 Meter"
  End Literal

End Icon

```



```

Literal Text
  Line 10
  Column 3
  Value "[ ]"
End Literal

Field TOGGLE_HIGH_JUMP
  Line 10
  Column 4
  Display
    Character Set Private_Rule
  Output Picture X
  Output ""
    When (TOGGLE_HIGH_JUMP = 1)
  Output " "
    When (TOGGLE_HIGH_JUMP <> 1)
  Protected
End Field

Icon EVENT_HIGH_JUMP
  Apply Field Default BUTTON_DEFAULTS
  Function Response SELECT
    If (TOGGLE_HIGH_JUMP = 0) Then
      Let TOGGLE_HIGH_JUMP = 1
    Else
      Let TOGGLE_HIGH_JUMP = 0
    End If
  End Response

  Literal Text
    Line 10
    Column 6
    Value " High Jump"
  End Literal

End Icon

Literal Text
  Line 10
  Column 24
  Value "[ ]"
End Literal

```

```

Field TOGGLE_400_METER
  Line 10
  Column 25
  Display
    Character Set Private_Rule
  Output Picture X
  Output ""
    When (TOGGLE_400_METER = 1)
  Output " "
    When (TOGGLE_400_METER <> 1)
  Protected
End Field

Icon EVENT_400_METER
  Apply Field Default BUTTON_DEFAULTS
  Function Response SELECT
    If (TOGGLE_400_METER = 0) Then
      Let TOGGLE_400_METER = 1
    Else
      Let TOGGLE_400_METER = 0
    End If
  End Response

  Literal Text
    Line 10
    Column 27
    Value " 400 Meter"
  End Literal

End Icon

Literal Text
  Line 11
  Column 3
  Value "[ ]"
End Literal

Field TOGGLE_JAVELIN
  Line 11
  Column 4
  Display
    Character Set Private_Rule
  Output Picture X
  Output ""
    When (TOGGLE_JAVELIN = 1)
  Output " "
    When (TOGGLE_JAVELIN <> 1)
  Protected
End Field

```

```

Icon EVENT_JAVELIN
  Apply Field Default BUTTON_DEFAULTS
  Function Response SELECT
    If (TOGGLE_JAVELIN = 0) Then
      Let TOGGLE_JAVELIN = 1
    Else
      Let TOGGLE_JAVELIN = 0
    End If
  End Response

  Literal Text
    Line 11
    Column 6
    Value " Javelin"
  End Literal

End Icon

Literal Text
  Line 11
  Column 24
  Value "[ ]"
End Literal

Field TOGGLE_5000_METER
  Line 11
  Column 25
  Display
    Character Set Private_Rule
  Output Picture X
  Output ""
    When (TOGGLE_5000_METER = 1)
  Output " "
    When (TOGGLE_5000_METER <> 1)
  Protected
End Field

Icon EVENT_5000_METER
  Apply Field Default BUTTON_DEFAULTS
  Function Response SELECT
    If (TOGGLE_5000_METER = 0) Then
      Let TOGGLE_5000_METER = 1
    Else
      Let TOGGLE_5000_METER = 0
    End If
  End Response

  Literal Text
    Line 11
    Column 27
    Value " 5,000 Meter"
  End Literal

End Icon

```

```

Literal Text
  Line 12
  Column 3
  Value "[ ]"
End Literal

Field TOGGLE_POLE_VAULT
  Line 12
  Column 4
  Display
    Character Set Private_Rule
  Output Picture X
  Output ""
    When (TOGGLE_POLE_VAULT = 1)
  Output " "
    When (TOGGLE_POLE_VAULT <> 1)
  Protected
End Field

Icon EVENT_POLE_VAULT
  Apply Field Default BUTTON_DEFAULTS
  Function Response SELECT
    If (TOGGLE_POLE_VAULT = 0) Then
      Let TOGGLE_POLE_VAULT = 1
    Else
      Let TOGGLE_POLE_VAULT = 0
    End If
  End Response

  Literal Text
    Line 12
    Column 6
    Value " Pole Vault"
  End Literal

End Icon

Literal Text
  Line 12
  Column 24
  Value "[ ]"
End Literal

```

```

Field TOGGLE_10000_METER
  Line 12
  Column 25
  Display
    Character Set Private_Rule
  Output Picture X
  Output ""
    When (TOGGLE_10000_METER = 1)
  Output " "
    When (TOGGLE_10000_METER <> 1)
  Protected
End Field

Icon EVENT_10000_METER
  Apply Field Default BUTTON_DEFAULTS
  Function Response SELECT
    If (TOGGLE_10000_METER = 0) Then
      Let TOGGLE_10000_METER = 1
    Else
      Let TOGGLE_10000_METER = 0
    End If
  End Response

  Literal Text
    Line 12
    Column 27
    Value " 10,000 Meter"
  End Literal

End Icon

Literal Text
  Line 13
  Column 3
  Value "[ ]"
End Literal

Field TOGGLE_DISCUS
  Line 13
  Column 4
  Display
    Character Set Private_Rule
  Output Picture X
  Output ""
    When (TOGGLE_DISCUS = 1)
  Output " "
    When (TOGGLE_DISCUS <> 1)
  Protected
End Field

```

```

Icon EVENT_DISCUS
  Apply Field Default BUTTON_DEFAULTS
  Function Response SELECT
    If (TOGGLE_DISCUS = 0) Then
      Let TOGGLE_DISCUS = 1
    Else
      Let TOGGLE_DISCUS = 0
    End If
  End Response

  Literal Text
    Line 13
    Column 6
    Value " Discuss"
  End Literal

End Icon

Literal Text
  Line 13
  Column 24
  Value "[ ]"
End Literal

Field TOGGLE_4X4_RELAY
  Line 13
  Column 25
  Display
    Character Set Private_Rule
  Output Picture X
  Output ""
    When (TOGGLE_4X4_RELAY = 1)
  Output " "
    When (TOGGLE_4X4_RELAY <> 1)
  Protected
End Field

Icon EVENT_4X4_RELAY
  Apply Field Default BUTTON_DEFAULTS
  Function Response SELECT
    If (TOGGLE_4X4_RELAY = 0) Then
      Let TOGGLE_4X4_RELAY = 1
    Else
      Let TOGGLE_4X4_RELAY = 0
    End If
  End Response

  Literal Text
    Line 13
    Column 27
    Value " 4x4 Relay"
  End Literal

End Icon

```

```

Literal Text
  Line 14
  Column 3
  Value "[ ]"
End Literal

Field TOGGLE_LONG_JUMP
  Line 14
  Column 4
  Display
    Character Set Private_Rule
  Output Picture X
  Output ""
    When (TOGGLE_LONG_JUMP = 1)
  Output " "
    When (TOGGLE_LONG_JUMP <> 1)
  Protected
End Field

Icon EVENT_LONG_JUMP
  Apply Field Default BUTTON_DEFAULTS
  Function Response SELECT
    If (TOGGLE_LONG_JUMP = 0) Then
      Let TOGGLE_LONG_JUMP = 1
    Else
      Let TOGGLE_LONG_JUMP = 0
    End If
  End Response

  Literal Text
    Line 14
    Column 6
    Value " Long Jump"
  End Literal

End Icon

Literal Rectangle
  Line 16 Column 19
  Line 18 Column 24
  Display
    Bold
End Literal

Literal Rectangle
  Line 16 Column 26
  Line 18 Column 35
End Literal

```

```

Icon ADD_REG_OK_BUTTON
  Apply Field Default BUTTON_DEFAULTS
  Function Response SELECT
    Message
      "Recording registrant data."
    Deactivate
      Panel ADD_REGISTRATION_DIALOG
    Remove
      ADD_REGISTRATION_VIEWPORT
    Position To Previous Item
  End Response

  Literal Text
    Line 17
    Column 20
    Value " OK "
    Display
      Bold
  End Literal

End Icon

Icon ADD_REG_CANCEL_BUTTON
  Apply Field Default BUTTON_DEFAULTS
  Function Response SELECT
    Deactivate
      Panel ADD_REGISTRATION_DIALOG
    Remove
      ADD_REGISTRATION_VIEWPORT
    Position To Previous Item
  End Response

  Literal Text
    Line 17
    Column 27
    Value " Cancel "
  End Literal

End Icon

End Panel

Panel WORK_IN_PROGRESS_BOX
  Viewport WORK_IN_PROGRESS_VIEWPORT
  Literal Rectangle
    Line 1 Column 1
    Line 5 Column 20
  End Literal

```



```

Literal Text
  Line 1
  Column 2
  Value " Work in Progress "
  Display
    Reverse
End Literal

Literal Text
  Line 3
  Column 3
  Value "Printing List..."
  Display
    Blinking
End Literal

End Panel

Panel PRINT_FILEBOX_PANEL
  Viewport PRINT_FILEBOX_VIEWPORT
  Function Response TRANSMIT
    If (FILEBOX_SELECTION_FIELD = " ") Then
      Signal
      Message "You must make a selection."
      Position To Field FILEBOX_SELECTION_FIELD on PRINT_FILEBOX_PANEL
    Else
      Deactivate
      Panel PRINT_FILEBOX_PANEL
      Remove
      PRINT_FILEBOX_VIEWPORT
      Position To Previous Item
    End If
  End Response

  Function Response DISCARD
    Deactivate
    Panel PRINT_FILEBOX_PANEL
    Remove
    PRINT_FILEBOX_VIEWPORT
    Position To Previous Item
  End Response

  Literal Rectangle
    Line 1 Column 1
    Line 16 Column 68
  End Literal

```

```

Literal Text
  Line 1
  Column 2
  Value " Print to File           "_
  "      "
  Display
  Reverse
End Literal

Literal Text
  Line 3
  Column 3
  Value "Filter:"
End Literal

Field FILEBOX_FILTER_FIELD
  Line 3
  Column 11

  Function Response SELECT
    Include DO_FILEBOX_FILTER
  End Response

  Exit Response
    Include DO_FILEBOX_FILTER
  End Response
End Field

Group FILEBOX_LIST
  Vertical
  Displays 8 First FIRST_FILE_INDEX
  Field FILE_SPEC
    Line 5
    Column 4

    Entry Response
      Include COMPUTE_UNSEEN_FILE_COUNT
    End Response

    Function Response MOVE_DOWN
      If FILEBOX_LIST(CURRENT_FILE_INDEX+1).FILE_SPEC = " " Then
        Signal
        Message "No more files."
      Else
        Position To Down Occurrence
      End If
    End Response

    Function Response MOVE_UP
      Position To Up Occurrence
    End Response
  End Field
End Group

```

```

Function Response MOVE_LEFT
    Message "No items in that direction."
    Signal
End Response

Function Response MOVE_RIGHT
    Position To Icon FILEBOX_OK_BUTTON On PRINT_FILEBOX_PANEL
End Response

Function Response SELECT
    Let FILEBOX_SELECTION_FIELD =
        FILEBOX_LIST(current_file_index).FILE_SPEC
End Response

Function Response JUMP_UP
    Position To Field FILEBOX_FILTER_FIELD On PRINT_FILEBOX_PANEL
End Response

Function Response JUMP_DOWN
    Position To Field FILEBOX_SELECTION_FIELD On PRINT_FILEBOX_PANEL
End Response

No Data Input
Active Highlight Reverse
End Field

End Group

Literal Text
    Line 5
    Column 52
    Value " "
End Literal

Field FILEBOX_LIST_TOP_INDICATOR
    Line 5
    Column 53 Protected
    Display
        Character Set Private_Rule
    Output "w" when first_file_index = 1
    Output ":" when first_file_index <> 1
End Field

Field FILEBOX_LIST_BOTTOM_INDICATOR
    Line 12
    Column 53 Protected
    Display
        Character Set Private_Rule
    Output "v" when unseen_file_count <= 0
    Output ":" when unseen_file_count > 0
End Field

Literal Polyline
    Line 6 Column 53
    Line 11 Column 53
End Literal

```

```

Literal Rectangle
  Line 4 Column 3
  Line 13 Column 54
End Literal

Literal Text
  Line 15
  Column 3
  Value "Selection: "
End Literal

Literal Rectangle
  Line 7 Column 57
  Line 9 Column 66
End Literal

Field FILEBOX_SELECTION_FIELD
  Line 15
  Column 14
End Field

Icon FILEBOX_FILTER_BUTTON
  Literal Rectangle
    Line 4 Column 57
    Line 6 Column 66
  End Literal

  Literal Text
    Line 5
    Column 58
    Value " Filter "
  End Literal

End Icon

Icon FILEBOX_OK_BUTTON
  Function Response SELECT
    If (FILEBOX_SELECTION_FIELD = " ") Then
      Signal
      Message "You must make a selection."
      Position To Field FILEBOX_SELECTION_FIELD On PRINT_FILEBOX_PANEL
    Else
      Deactivate
      Panel PRINT_FILEBOX_PANEL
      Remove
      PRINT_FILEBOX_VIEWPORT
      Position To Previous Item
    End If
  End Response

  Active Highlight
  Reverse

```

```
    Literal Text
      Line 8
      Column 58
      Value " OK "
    End Literal
  End Icon
  Icon FILEBOX_CANCEL_BUTTON
    Function Response select
      Deactivate
        Panel PRINT_FILEBOX_PANEL
      Remove
        PRINT_FILEBOX_VIEWPORT
    End Response
    Literal Rectangle
      Line 10 Column 57
      Line 12 Column 66
    End Literal
    Literal Text
      Line 11
      Column 58
      Value " Cancel "
    End Literal
  End Icon
End Panel
End Layout
End Form
```


B

Track and Field Registration Application

The program that follows is the DECforms Track and Field Registration program. The application is written in C.

```
/*
 *
 *   © Copyright 2005 Hewlett-Packard Development Company, L.P.
 *
 *   This software is furnished under a license and may be used and copied
 *   only in accordance with the terms of such license and with the
 *   inclusion of the above copyright notice. This software or any other
 *   copies thereof may not be provided or otherwise made available to any
 *   other person. No title to and ownership of the software is hereby
 *   transferred.
 *
 *   The information in this software is subject to change without notice
 *   and should not be construed as a commitment by Hewlett-Packard Development
 *   Company.
 *
 *   HP assumes no responsibility for the use or reliability of its
 *   software on equipment which is not supplied by HP.
 *
 * Facility: tafr - track and field registration demo
 *
 * Abstract: This module contains the main driver and general purpose routines.
 *
 * Environment: VAX/VMS and RISC/ULTRIX
 */
#include <stdio.h>
#include <forms_def.h>

#ifdef vms
#include <descrip.h>
#else
#include <sys/dir.h>
#endif
```

```

/*
 * Table of contents
 */
int main();
void pause_interface();
unsigned fill_filebox();
void check_status ();

#ifdef vms
unsigned vms_fill_filebox();
#else
unsigned ultrix_fill_filebox();
#endif

/*
 * Variables used by the Portable API request calls.
 */
Forms_Request_Options enable_options[2];
Forms_Session_Id      session_id;
Forms_Form_Object     TAFR_FORM;
Forms_Status          status;

/*
 * Platform dependent information.
 */

#ifdef vms
#define device_name "SYS$OUTPUT"

/*
 * VMS Descriptors
 */
typedef struct {
    unsigned short length;
    unsigned char dtype;
    unsigned char class;
    char *pointer;
} Descriptor;

#define _dx_Empty {0, DSC$K_DTYPE_T, DSC$K_CLASS_S, NULL}
#else

/*
 * Constants for ULTRIX
 */

#define device_name "/dev/tty"
#define MAX_COMMAND_LENGTH 256
#define MAX_STRING_LENGTH 48
#endif

```



```

/*
* Routine: main
*
* Functional Description:
*
*   Executive routine for the tafr program. This routine controls the high-
*   level execution which consists of enabling and disabling the form. All
*   user interaction takes place as a result of the enable response. The
*   enable request does not terminate until the user terminates the
*   interactive session.
*
* Formal Parameters:
*
*   none (the argc and argv C parameters are not used by this program)
*
* Routine Value:
*
*   true
*
*/

int main ()
{
    char display_device[11];

    /* Set up display device name for the Enable call */
    strcpy (display_device, device_name);

    enable_options[0].option = forms_c_opt_form;
    enable_options[0].form.object = TAFR_FORM;

    enable_options[1].option = forms_c_opt_end;

    status = forms_enable (session_id,
        display_device,
        NULL,
        "TAFR_FORM",
        enable_options);

    check_status (status, "Enable");

/*
* Disable the form
*/
    status = forms_disable (session_id, NULL);
    check_status (status, "Disable");
}

void check_status (Forms_Status status, char *request_name)
{
    char msg_text[256];
    Forms_Status disable_status;

    /*
    /* If request status is not normal then call forms_errormsg
    /* to translate fims error number into message text.
    /*
    forms_errormsg (status, msg_text);

```

```

if ((status != forms_s_normal) &&
    (status != forms_s_return_immed) &&
    (status != forms_s_converr)) {
    /*
    * Disable the form
    */
    disable_status = forms_disable (session_id, NULL);

    /* print out the fims error number */
    fprintf(stderr, "%s FIMS error number is %d or in hex %x \n", request_name, status, status);

    /* print out the corresponding message text */
    fprintf(stderr, "%d : %s", status, msg_text);

    if (disable_status != forms_s_normal)
        fprintf (stderr, "Failure disabling the form. Status is %d \n.", disable_status);

    exit (0);
}
}

/*
* Routine: pause_interface
*
* Functional Description:
*
* This procedural escape routine pauses for the specified number of
* seconds. It is used as a "stub" for a function that, if actually
* implemented, might take a significant length of time. This routine
* could be used, for example, to display a "work in progress" box for
* a specified period of time.
*
* Formal Parameters:
*
* none
*
* Routine Value:
*
* none
*/
void pause_interface (delay)
    unsigned delay;
{
    sleep (delay);
    return;
}

```

```

/*
* Routine: fill_filebox
*
*
* Functional Description:
*
*   This jacket routine determines the platform and calls the appropriate
*   fill_filebox routine.
*
* Formal Parameters:
*
*   filter      (by ref)  - A file specification which may consist of directory spec,
*                          file spec, file type and wildcards.
*   file_array  (by ref)  - An array buffer to store all the file names which match the
*                          criteria specified by the filter.
*   string_length (by value) - Maximum length of the file filter.
*   array_length (by value) - Maximum length of the array buffer.
*
* Routine Value:
*
*   file_count - number of files that match the given filter.
*
*/
unsigned fill_filebox (
    char *filter,
    char *file_array,
    int  string_length,
    int  array_length)
{
    unsigned file_count;

#ifdef vms
    file_count = vms_fill_filebox (filter, file_array, string_length, array_length);
#else
    file_count = ultrix_fill_filebox (filter, file_array, string_length, array_length);
#endif

    return file_count;
}
#ifdef vms

```

```

/*
* Routine: vms_fill_filebox
*
*
* Functional Description:
*
*   This VMS specific routine sets up VMS descriptors and make a call to Lib$Find_File
*   using the given filter and load all the file names satisfying the filter criteria
*   into the array buffer.
*
* Formal Parameters:
*
*   filter      (by ref)  - A file specification which may consist of directory spec,
*                          file spec, file type and wildcards.
*   file_array  (by ref)  - An array buffer to store all the file names which match the
*                          criteria specified by the filter.
*   string_length (by value) - Maximum length of the file filter.
*   array_length (by value) - Maximum length of the array buffer.
*
* Routine Value:
*
*   n - number of files that match the given filter.
*/

unsigned vms_fill_filebox (
char *filter,
char *file_array,
int string_length,
int array_length)
{
    unsigned n, ok, context = 0;
    Descriptor result_dx = _dx_Empty;
    Descriptor filter_dx = _dx_Empty;
    char *ptr;

    /* Initialize the file array buffer with blank spaces */
    memset (file_array, ' ', string_length * array_length);

    /* Set up VMS descriptors for Lib$Find_File */

    filter_dx.length = string_length;
    filter_dx.pointer = filter;

    result_dx.length = string_length;
    result_dx.pointer = malloc (string_length);

    /* Locate all files which matches the given filter criteria */
    /* and load their names into the file array buffer.          */

    for (n = 0; n < array_length; n++) {
        ok = Lib$Find_File (&filter_dx, &result_dx, &context);

        /* VMS status divisible by 2 indicates failure. */
        /* Bail out if this is the case.                */

        if (ok % 2 == 0) break;
    }
}

```

```

ptr = file_array + (n * string_length);
    strncpy (ptr, result_dx.pointer, string_length);
}

/* clean up before we return */

Lib$Find_File_End (&context);
free (result_dx.pointer);
return n;
}
#endif

#ifdef vms

/*
 * Routine: ultrix_fill_filebox
 *
 * Functional Description:
 *
 * This ULTRIX specific routine sets up a pipe to execute a "ls" command with the given
 * filter and stores the result listing into the array buffer.
 *
 * Formal Parameters:
 *
 * filter          (by ref)  - A file specification which may consist of directory spec,
 *                            file spec, file type and wildcards.
 * file_array      (by ref)  - An array buffer to store all the file names which match the
 *                            criteria specified by the filter.
 * string_length  (by value) - Maximum length of the file filter.
 * array_length   (by value) - Maximum length of the array buffer.
 *
 * Routine Value:
 *
 * n - number of files that match the given filter.
 */

unsigned ultrix_fill_filebox (
    char *filter,
    char *file_array,
    int string_length,
    int array_length)

{
    char buf[MAX_STRING_LENGTH];
    char *file_array_ptr;
    char *command_string;
    FILE *popen_ptr;
    size_t trim_blanks;
    int n;

    /* Initialize variables */

    command_string = (char *) malloc (MAX_COMMAND_LENGTH);
    memset (file_array, ' ', string_length * array_length);
    n = 0;

    /* Construct the command string to be executed */

```

```

strcpy (command_string, "/bin/ls ");
trim_blanks = strcspn (filter, " ");
strncat (command_string, filter, trim_blanks);

/* Establish a pipe for reading the result of the ls command */

popen_ptr = popen (command_string, "r");
if (popen_ptr == NULL) return 0;

/* Locate all files which matches the given filter criteria */
/* and load their names into the file array buffer.          */

while ( fscanf(popen_ptr, "%48[^\n]", buf) != EOF ) {
    fgetc (popen_ptr);
    file_array_ptr = file_array + (char *) (n++ * string_length);
    strcpy(file_array_ptr, buf);
    *buf = 0;
}

/* clean up before we leave */

pclose (popen_ptr);
free (command_string);
return n;
}
#endif

```

Glossary

accelerator

A key or key sequence that provides a shortcut to access an application function quickly. Also called a *keyboard accelerator*.

bar menu

A rectangular area at the top of the screen that contains the names of pull-down menus for an application.

button

An on-screen control that allows users to choose actions or operations. See also push button.

cascade indicator

An arrow character (→) that appears to the right of a menu item to indicate that the menu item is a cascade item.

cascade item

A menu item that displays a cascade menu.

cascade menu

A pull-down menu evoked from another menu that provides selections that amplify the parent menu item. A cascade indicator associated with the parent menu item indicates the availability of a cascade menu. Also called a *submenu*.

caution box

A standard informational dialog box that informs the user of the consequences of carrying out an action. When the box appears, application activity stops, and user input is required for application activity to proceed.

check field

A control used to choose options that are not mutually exclusive. A check field consists of a label describing an option and a check indicator to show if the field is on or off. Compare with *radio field*.

check indicator

A graphic symbol used to show whether a check field is on or off. The indicator is made of square brackets surrounding a diamond character. The diamond character disappears when the check field is turned off.

command item

A choice on a menu that initiates an action or operation directly, without calling a submenu.

command line

A field in which users can enter typed commands.

control

An on-screen object that allows users to provide input to applications. See also button, check field, list group, option field, push button, radio field, text-entry field.

control panel

A permanently displayed dialog box containing controls that are used often during a work session.

cursor

An on-screen symbol that indicates the current object.

default push button

The push button that provides the user with the most likely response to a dialog box query.

dialog box

A secondary panel that displays messages to the user and solicits input from the user. Usually, the user must take an appropriate action (as indicated by the choices presented in the dialog box) to continue application activity.

dismiss

To remove a menu or dialog box from the screen without changing any settings.

file selection dialog box

A specialized dialog box that allows the user to specify a file name within the application.

flush left

The beginning of the text or graphic object is lined up evenly with the left margin with no indentation.

flush right

The end of the text or graphic object is lined up evenly with the right margin with no indentation.

highlight

A visual indication of the current object. Typically the highlight is accomplished through reverse video.

icon

A form element that is composed entirely of background literals. An icon is similar to a field, but it cannot have data input. An icon can have function key input.

indicator

A symbol that designates the status or presence of a particular object. See also cascade indicator, check indicator, list indicator, radio indicator.

informational dialog box

A specialized dialog box that is used to display important messages that require acknowledgement.

label

The text that identifies a control.

left-justify

To align a group of text elements by lining up the beginning character of each element in the same column.

list group

A displayed list of items, such as available files, from which the user can select. The user scrolls through the list by using navigational keys to display a portion of the list in the scroll area.

list indicator

Graphic symbols used with a list group to indicate whether or not there are more items in the underlying list in either scroll direction.

menu

A list from which users can choose one or more items. See also bar menu, cascade menu, pop-up menu, pull-down menu, submenu.

menu item

A choice on any type of menu. See also cascade item, command item, toggle item.

menu name

The title of a menu listed in the bar menu.

message panel

The two-line area on the bottom of the screen used by the application to display messages not requiring immediate action.

mnemonic

A single character in a menu item, indicated by an underscore, that provides a shortcut for choosing that menu item. Users can press the PF4 key and the mnemonic character to choose the corresponding menu item without using the navigation keys.

navigation key

One of the keys defined for moving the cursor on the screen.

object

An entity on the screen, such as a button, control, menu, text, and so on.

obscure

To conceal all or part of the screen that would otherwise be visible.

option field

A field that the user completes by choosing from a list of options. The list of options is presented in either an associated pop-up menu or in a list group in a dialog box. Once an option is chosen, the option becomes the value of that field.

pop-up menu

A context-sensitive menu that appears whenever the user presses a special key sequence.

pull-down menu

A menu that is displayed when the user selects a menu item from the menu bar or a cascade item from another menu.

push button

A control that consists of a rectangular box surrounding a label that indicates a command to be performed. The user chooses the command by selecting the push button.

question dialog box

A specialized dialog box used to ask the user a brief question. Question dialog boxes are used typically to caution the user and to confirm an action.

radio box

A set of radio fields. Only one radio field in a radio box can be on at one time.

radio field

A control used to choose among mutually exclusive options. Radio fields consist of a label describing an option and an indicator made of angle brackets surrounding a diamond character. The diamond character disappears when the radio field is turned off. Within a set of radio fields, only one can be on at a time. Compare with *check field*.

radio indicator

A graphic symbol used to show whether a radio field is on or off. The indicator is made of angle brackets surrounding a diamond character. The diamond character disappears when the radio field is turned off.

reverse video

A video display characteristic that is used to highlight an object on the screen. If the default video display is white characters on a black background, reverse video displays black characters on a white background.

right-justify

To align a group of text elements by lining up the last character of each element in the same column.

select

To initiate an action by positioning the cursor to a screen object and pressing the Select key or the keypad period key.

scroll area

A window behind which the items in a list group can be scrolled.

submenu

A menu, associated with a pull-down or pop-up menu, that expands on the choices offered by the menu and is displayed in response to selecting the name of the submenu. A cascade indicator (→) at the right of the submenu name indicates the availability of a submenu. Also called a *cascade menu*.

text entry field

A field into which the user can type information.

toggle

To switch a two-state option to its opposite state.

toggle item

A menu item that is either a check field or one of a group of radio fields. Also called a *toggle field*.

unavailable menu item

A disabled menu item that is currently visible but cannot be selected. When a menu item is unavailable, it is surrounded by braces ({}).

work area

The panel in which users perform most application tasks.

work in progress dialog box

A specialized dialog box used to display information about a current operation.

Index

A

- Accelerators, 4–3
 - placement, 4–1
 - when unavailable, 4–5
- Application title bar
 - description, 2–2

B

- Bar menu
 - description, 2–3, 4–1, 4–6
 - example, 2–2, 2–3
 - example with pull-down menus, 4–7
 - standard items, 4–7
 - two-line, 4–7

C

- Cascade indicator
 - description, 4–3
 - example, 4–2, 4–4
 - location relative to pull-down menu, 4–8
 - placement, 4–1
 - when highlighted, 4–4
- Cascade item
 - description, 4–3
 - example, 4–2
 - placement, 4–1
 - when unavailable, 4–5
- Caution
 - in dialog box, 5–8
- Check field
 - appearance, 3–5
 - as toggle item, 4–3
 - description, 3–5
 - example, 3–5
 - label, 3–5
 - labeling guidelines, 2–9
 - selection, 3–5
 - when to use, 2–6
- Check indicator, 3–5
- Color
 - use, 1–5
- Column heading
 - with list group, 3–8
- Command item
 - description, 4–3
 - example, 4–2
- Confirmation
 - in dialog box, 5–8
- Consistency
 - definition, 1–5
 - purpose, 1–5
- Control panel, 2–1
 - when to use, 2–7
- Controls, 3–1 to 3–13
 - check fields, 3–5
 - list groups, 3–8 to 3–11
 - option fields, 3–12 to 3–13
 - push buttons, 3–1 to 3–3
 - radio fields, 3–3 to 3–4
 - text entry fields, 3–6 to 3–8
- Ctrl/H keys
 - in screen navigation, 2–10

Ctrl/J keys
in text editing, 3–7

Ctrl/Z keys
in dialog box, 5–3
in file selection dialog box, 5–11
in screen navigation, 2–10

D

Default push button
alternate appearance of, 3–2
appearance of, 3–2
example, 3–2
when to use, 5–4

Delete key
in text editing, 3–7

Design
common pitfalls, 1–7

Dialog box
appearance, 5–1
arranging text entry fields in, 5–5
banner, 5–1
cancelling, 5–3
chaining, 5–3
description, 2–5
example, 2–5
file selection, 5–9
grouping controls, 5–3
grouping push buttons, 5–3
informational, 5–7
labeling guidelines, 2–8
labeling objects within, 2–9
linking to a text entry field, 3–7
location, 5–2
purpose, 5–1
question, 5–8
sample, 5–2
size and placement, 5–2
specialized, 5–6
standard, 5–6
title, 5–1
when to use, 2–5
with command item, 4–3
with option field, 3–13
Work in Progress, 5–6

Direct manipulation, 1–4

Dismissing menus, 4–6

Down arrow key
in list group, 3–10
in screen navigation, 2–10

E

Editing text
in text entry field, 3–7

Edit menu item, 4–8

Ellipsis
in menus, 4–3
in option field, 3–12
in push button, 3–3
in text entry field, 3–7
use in menus, 5–1
use in text entry field, 5–1

Entering text
in text entry field, 3–7

Error messages
displayed in dialog box, 5–7

Errors
guidelines for anticipating, 1–6

Explicit destruction
use, 1–7

Extending this style guide, 1–2

F

F10 key
in dialog box, 5–3
in file selection dialog box, 5–11
in screen navigation, 2–10

F12 key
in screen navigation, 2–10

F13 key
in text editing, 3–7

F8 key
in dialog box, 5–3
in file selection dialog box, 5–11
in menus, 4–6
in screen navigation, 2–10

File filter, 5–10
File menu item, 4–8
File selection box, 5–9
 example, 5–10
 keys used to navigate, 5–11
Filter push button, 5–10

H

Help menu item, 4–8

I

Informational dialog box, 5–7
 example, 5–8
Interface design
 elements of good, 1–3

K

Keyboard accelerators, 4–3
KP. key
 in list group, 3–11

L

Labeling guidelines, 2–7 to 2–9
 dialog boxes, 2–8
 general, 2–7
 menus, 2–8
 objects in dialog boxes, 2–9
 push buttons, 2–9
Left arrow key
 in list group, 3–10
 in screen navigation, 2–10
 in text editing, 3–7
Linefeed key
 in text editing, 3–7
List group
 appearance, 3–8
 description, 3–8
 example, 3–8
 example of two column, 3–9
 keys used to navigate within, 3–10
 navigation within, 3–10
 selection, 3–9

List group (cont'd)
 using text entry fields, 3–8
 when to use, 2–6
 with option field, 3–13
List indicator, 3–8, 3–9

M

Main screen
 different areas, 2–1
 example, 2–2
 when to use, 2–1
Menu items
 choosing, 4–4
 description, 4–3
 grouping guidelines, 4–12
 naming guidelines, 4–11
 when unavailable, 4–5
Menus
 appearance, 4–1
 bar menu, 2–3, 4–1, 4–6
 choosing an item, 4–4
 components, 4–3
 description, 2–4
 dismissing, 4–6
 guidelines for designing, 4–11
 keyboard accelerators, 4–3
 labeling guidelines, 2–8
 menu items, 4–3
 mnemonics, 4–4
 pop-up, 4–1, 4–10
 pull-down, 4–1, 4–8
 separators, 4–4, 4–12
 showing unavailable items, 4–5
 types, 4–6
 when to use, 2–4
Message panels
 description, 2–3
 example, 2–2
 use with Work in Progress dialog box,
 5–7
 when to use, 3–3, 5–6
Mnemonic introducer key, 4–4

Mnemonics
description, 4–4
for standard items, 4–7
Modifying this style guide, 1–2

N

Navigation
guidelines, 2–9
guidelines for, 1–5
keys used, 2–10
within a list group, 3–10
Next Screen key
in list group, 3–10

O

Option field
appearance, 3–12
description, 3–12
example, 3–12
example when highlighted, 3–12
example with pop-up menu, 3–13
example with undefined initial value,
3–12
label, 3–12
labeling guidelines, 2–9
selection, 3–13
when to use, 2–6
Options menu item, 4–8

P

PF1 B keys
in list group, 3–11
PF1 down arrow keys
in list group, 3–11
PF1 KP4 keys
in list group, 3–10
PF1 KP5 keys
in list group, 3–11
PF1 left arrow keys
in list group, 3–11

PF1 Next Screen keys
in list group, 3–11
PF1 Prev Screen keys
in list group, 3–11
PF1 Q keys
in menus, 4–6
in screen navigation, 2–10
PF1-Q keys
in dialog box, 5–3
in file selection dialog box, 5–11
PF1 right arrow keys
in list group, 3–11
PF1 T keys
used in list group, 3–11
PF1 up arrow keys
in list group, 3–11
PF3 key, 4–10
PF4 key, 4–4
Pop-up menu
description, 4–1, 4–10
example used with option field, 3–13
implementing, 4–11
location, 4–11
used with option field, 3–13
when to use, 2–7
Prev Screen key
in list group, 3–11
Progress indicator, 5–7
Progressive disclosure, 1–4
Pull-down menu
description, 4–1, 4–8
example, 2–4, 4–9
example when space is tight, 4–10
from bar menu, 4–6
location of, 4–9
when to use, 2–7
Push buttons
alternate appearance, 3–2
appearance, 3–1
default, 3–2, 5–4
description, 3–1
example, 3–2, 3–3
example of horizontal arrangement, 5–3
example of vertical arrangement, 5–4
grouping in dialog box, 5–3

Push buttons (cont'd)

- label, 3-3
- labeling guidelines, 2-9
- placed horizontally, 5-3
- selection, 3-3
- stacked vertically, 5-3

Q

- Question dialog box
- description, 5-8
 - example, 5-9

R

- Radio box
- arranged horizontally, 5-5
 - arranged vertically, 5-5
 - description, 5-5
 - example of horizontal arrangement, 5-5
 - example of vertical arrangement, 5-5
 - title, 5-5
- Radio field, 3-3
- appearance, 3-4
 - as toggle item, 4-3
 - example, 3-4
 - label, 3-4
 - labeling guidelines, 2-9
 - selection, 3-4
 - used in radio box, 5-5
 - when to use, 2-6
- Radio indicator, 3-4
- Response
- guidelines for, 1-4
- Return key
- in list group, 3-11
 - in screen navigation, 2-10
- Right arrow key
- in list group, 3-10
 - in screen navigation, 2-10
 - in text editing, 3-7

S

- Screen design, 2-1 to 2-10
- labeling screen objects, 2-7
 - when to use check fields, 2-6
 - when to use control panels, 2-7
 - when to use dialog boxes, 2-5
 - when to use list groups, 2-6
 - when to use main screen, 2-1
 - when to use menus, 2-4
 - when to use option fields, 2-6
 - when to use pop-up menus, 2-7
 - when to use pull-down menus, 2-7
 - when to use radio fields, 2-6
- Screen navigation, 2-9
- Screen object
- labeling guidelines, 2-7
- Scroll area, 3-8
- Selection field, 5-10
- Select key
- in list group, 3-11
- Separators
- in menus, 4-4, 4-12
- Style guide
- advantages of using, 1-1, 1-2
 - extending, 1-2
 - modifying, 1-2
- Submenu
- indication, 4-3
 - maximum number, 4-12
 - pull-down menus, 4-8

T

- Text entry field
- appearance, 3-6
 - arranging in dialog box, 5-5
 - description, 3-6
 - entering text, 3-6
 - example, 3-6
 - example of link to dialog box, 3-8
 - example of stacking, 5-6
 - in list group, 3-8
 - keys used to edit text, 3-7

Text entry field (cont'd)
label, 3-6
labeling guidelines, 2-9
linking to a dialog box, 3-7

Title bar

example, 2-2

Toggle indicator

example, 4-4

placement, 4-2

when highlighted, 4-4

Toggle item

description, 4-3

example, 4-2

example when highlighted, 4-5

placement, 4-2

when unavailable, 4-5

U

Unavailable menu items, 4-5

example, 4-6

Up arrow key

in list group, 3-10

in screen navigation, 2-10

V

View menu item, 4-8

W

Warning

displayed in dialog box, 5-7

Work area

description, 2-3

example, 2-2

Work in Progress dialog box, 5-6

example, 5-7

when to use, 3-3