COBOL sp2
User Interface Development System
Version 5.3 User Guide

Flexus
P.O. Box 640
Bangor PA 18013-0640
U.S.A.

Voice: 610-588-9400
Fax: 610-588-9475
E-Mail: info@flexus.com
WWW: http://www.flexus.com
Downloads: http://flexus.sharefile.com

TABLE OF CONTENTS

PART A

Getting Started

CHAPTER A1 - Before You Begin

WHAT IS COBOL sp2?

COBOL sp2 is a tool used by COBOL programmers to develop and maintain contemporary user interfaces for their programs. It allows the complexities of such an interface to be somewhat isolated from a COBOL program, yet still enables a COBOL programmer to have close control over the interface directly from a program using standard COBOL code.

The Panel Editor

COBOL sp2 provides a panel editor for designing and developing your interface. Development is done in a graphical mode, specifically, in Microsoft Windows. However, it is possible that you might deploy your interface in an alternative environment. These alternative environments can range all the way from a character mode terminal to a web browser running on a mobile device. For more on this topic, please see Chapter A3.

The Runtime System

For implementing your interface, COBOL sp2 provides a powerful runtime system (also used by the editor itself, incidentally) accessed through simple call statements. Sample call statements and a copy file containing interface data items can be generated using the panel editor's own code generator. Different versions of the runtime system allow the same code and panel definition to work in different environments.

HOW DOES COBOL sp2 COMPARE TO TRADITIONAL SCREEN DEVELOPMENT METHODS?

Traditionally, user interfaces or screens for COBOL systems have been developed using one of the following methods:

- Display/accept logic.
- Display/accept logic incorporating a screen section.
- A home-grown screen-handling subroutine.
- A vendor-supplied screen-handling tool.
- Hardware-specific screen-handling methodology.

DISPLAY/ACCEPT Logic

DISPLAY/ACCEPT logic imbeds the user interface entirely within a system's business logic. While this is adequate in a lot of cases, it can make program logic very complex as your user interface becomes more complex. Additionally, coding such interface code can be both tedious and time-consuming. With DISPLAY/ACCEPT logic it may be impossible to build an adequate user interface, especially in a graphical environment.

Screen Sections

Screen sections can greatly simplify display/accept coding because most of the details of the user interface can be laid out fairly intuitively within a program's working-storage section. You may find that many user interface features are not supported by the screen section because it takes time to incorporate new language elements to provide support for such features. This is especially true for graphical user interfaces.

Screen-Handling Subroutines

Home-grown screen-handling subroutines have been used effectively in the past to isolate screen-handling code from the business logic. The problem people have today, however, is that it may be difficult to maintain or enhance such a subroutine to provide features needed in a more contemporary user interface. Fortunately, it is usually fairly easy to convert from your own subroutine to COBOL sp2, which can be thought of as a subroutine itself. It is also possible to write your own import routines using the COBOL sp2 runtime, so that you can salvage any existing screen definitions that do not require too much change.

Screen-Handling Tools

Screen-handling tools supplied by other vendors may no longer satisfy your current requirements or be unsuited for use with COBOL. It is often not too difficult to convert to COBOL sp2. Import tools may already be available or can be made available, or you might be able to write one yourself. Once you have converted to using the COBOL sp2 runtime, you will have isolated the core of your user interface, yet will still have convenient access to user interface elements from your COBOL program, whenever necessary. You will be using standard COBOL call statements to control the interface and your code will be easier to enhance and maintain: some changes to your user interface may need no code changes at all. You will also be well positioned for dealing with the difficulties and intricacies of modern user interfaces, yet you will still be able to write your own unique business code using the language best suited for that purpose, COBOL.

Hardware-Specific Screen-Handling

Screen-handling methodology supplied by hardware manufacturers is a problem if you need to move to other hardware or you need to move from a proprietary operating system to an open-systems one. Some systems are converted fairly easily and the interface with the COBOL sp2 runtime is somewhat similar to a number of those systems. Moreover, an import tool may already be available for the system that you were using, allowing you to incorporate existing screen layouts without too much trouble. Some systems use coding techniques that are somewhat non-standard in COBOL terms and may be harder to convert. It may nevertheless be possible to import screen definitions into the panel editor so that design work is kept to a minimum.

INTRODUCING COBOL sp2 INTO YOUR ORGANIZATION

Introducing COBOL sp2 into your organization will probably require that you make some changes in the way that you develop your software. If you are developing new software it may be easier to make these changes, but there is no reason why development with COBOL sp2 cannot be phased in to your current development methodology.

New Software Development

If you are developing a brand new system or a new subsystem that has little or no interaction with existing system components, then you should have no problems using COBOL sp2 for developing your user interface.

You can quickly become familiar with the panel editor by working through Part B of the documentation, and Part C should give you enough information to start writing programs that use COBOL sp2 panel definitions.

Parts D and E are reference sections and would normally be accessed (at least initially) using the extensive Index at the back of the documentation. For instance, if you need to set a certain screen component from your program, you would look up that component in the Index, which would point you to various references, usually in Parts D or E. If the reference was in Part E (Data items), then you would also need to look up the programming function mentioned at the start of Part D.

Enhancing an Existing System

If you have an existing system that you need to move to a new operating system, or you need to enhance an older system to bring it in line with contemporary user interfaces, then there are clearly additional considerations involved. Hopefully, the task of becoming familiar with the COBOL sp2 panel editor and runtime, as discussed above, will go fairly quickly. The next sections discuss using COBOL sp2 methodology as a replacement for, or in conjunction with, your existing screen-handling code.

CONVERTING EXISTING CODE TO WORK WITH COBOL sp2

Replacing existing code may be necessary either because your existing screen-handling techniques do not support the features needed in your system's user interface, or those techniques are not supported by some new hardware or operating environment.

Is a Conversion Practical?

Converting to sp2 can be a lot easier than you might think, mainly due to the availability of conversion services from Flexus. Although there is usually a fee involved in this work it is certainly worth considering if you have a significan number of screens to convert. Of course it is also possible to do the conversion work yourself but you have to consider the time and manpower that may be involved.

Using Flexus Conversion Services

This process involves sending yor source code to Flexus and having Flexus convert it and then return it to you for testing. A small subset of the code might be sent initially to establish proof of concept. The turnaround for this work can be quite quick due to the development by Flexus of some automated conversion routines that can be adapted to different types of code. Currently we are able to convert the following types of code:

- Traditional in-line display/accept.
- Screen section (various compiler dialects)
- Screen section with GUI extensions
- Some proprietary tools (call for availability)

Please contact Flexus to discuss your particular situation so that we can come up with a plan to move yur conversion forward:

Converting Code Yourself

If you have ony a small number of screens or you have access to tools tat might help with the conversion then doing the conversion work yourself might be practical. It may also be worth considering using sp2 in conjunction with your existing code rather than as a complete replacement for it - see the topic on this later on in this section. Of course the downside to such a piecemeal approach is that this combination user interface will probably not run in all environments to which you want to deploy - such as a web browser.

Converting Screen Layouts

Most likely the first step is to convert your existing screen definitions into COBOL sp2 panel definitions. There are a number of possibilities here, as follows:

- Recreate screen layouts using the panel editor.
- Write an import tool.
- Dynamically define panel definitions from within your program.

Using the Panel Editor

Using the panel editor to recreate screens is the most straightforward approach but can become time-consuming if you have a lot of screens in your system. If you do just have a small number of screens, this approach may in fact be beneficial because you may find yourself reorganizing screen layouts given the ease with which this can be done using the panel editor.

Using an Import Tool

If you have a good number of screens and the definitions could be imported (you are using screen sections, for instance,) an import tool is definitely the way to go. Before making any decisions, contact your dealer to check if such a tool already exists. If not, ask your dealer what it would take to have one written for you. If this is not possible (this is usually the case only because of time constraints,) then consider writing an import tool yourself. The COBOL sp2 runtime allows panels to be defined from within a program and then saved to a panel file, essentially reproducing the functionality of the panel editor.

Defining Panels Dynamically

If it is not possible to import screen definitions, it may still not be necessary to use the panel editor to recreate the screens. Because COBOL sp2 allows panels to be defined from within your program, you may wish to consider substituting each in-line display statement with a COBOL sp2 call statement to create a text or field item. It may even be possible to use a text editor or some other tool to automate this effort to some extent. Refer to part C for an introduction to creating panels dynamically.

PROCESSING USER INPUT WITH COBOL sp2

Once you have converted your screen layouts to COBOL sp2 panel definitions you can use a single COBOL sp2 call statement (the CONVERSE-PANEL function) to display a panel and accept user input. This function call can automatically open a window, display a panel within that window, and then process user input into the fields of that panel.

Screens, Windows and Panels

The idea of opening a window rather than just displaying a panel on the screen may be somewhat of a new concept to you. A window is an object that serves as an interface between your panel definition and the video hardware (the screen.) Windows are primarily used so that multiple processes can all share the same screen in graphical environments. They are also useful in single process environments for displaying pop-up panels and automatically restoring underlying panel images. Windows can usually also be moved around the screen and resized, even in text mode. Of course, in text mode, you may wish to have your panel take up the whole screen (no one else is using it after all!) One of the text mode options allows you to specify that a panel should automatically take up the whole screen when running in text mode but be shown as a normal window in graphical mode.

Where Does User Input Go?

As input is keyed into each entry field by your user, that data is formatted and passed back to your program's field area. The field area for each panel is generated as part of a copy file when you invoke the panel editor's generator. If you are defining your panels dynamically you will effectively build your own field area when you set the program offset of each field. Refer to part C for more details of this.

Once a panel has been completed by your user, you can move data out of your program's fields area into a record area elsewhere in working-storage, prior to writing the data to a file. If you wish to avoid this intermediate moving logic, it is possible to format a panel's field area to match exactly a file record layout. This is done by specifying the program offset of each field yourself rather than have the editor set a default offset which relates to the order of the fields within the panel.

As might be expected, your program can also move data into the field area so that you can display new data within a panel's fields. This might be done prior to the initial display of the panel or on a subsequent display after a record has been read. Each display is usually done using the same CONVERSE-PANEL function call - a switch set only on the first call causes a new window to be opened to contain the panel.

Monitoring User Requests

As well as keying data in entry fields, your user may request that your program perform a particular task. For instance, he may press a function key that requests a file look-up.

It was mentioned above that the field area is only part of the copy file that is generated for a panel. The rest of the file holds data items that allow your program to detect what action has been performed by the user, and to control the subsequent display of your panel. For example, a code is set indicating the key that the user pressed to cause control to be returned to your program.

Other data items hold such things as the current cursor position, and the next cursor position (which could of course be reset by your program.)

Field-by-Field Processing

Your user interface code may demand more detailed interaction than merely displaying a panel and waiting for the user to complete that panel. Many programs are set up to interact with the user on a field-by-field basis. For instance, as soon as the user keys in an account number, the program may need to verify this number against a file. Some programmers are hesitant about using a product that might prevent them from using this field-by-field logic. Indeed, by default, the COBOL sp2 runtime would not normally return control to your program until the user has gone through all the fields on a panel and pressed some termination key (say Enter or Return.) Fortunately it is easy to modify this default behavior. The normal way to do this is to define a field as a "control" field. As soon as the user exits such a field, control will be returned to your program so that you can perform further processing. See part C for more details on control fields.

By incorporating control fields in a panel (every field in a panel can be a control field if you like) you can match existing field based processing very closely. One advantage that panel/control field processing has over traditional field-by-field processing is that the next field to be processed is normally under the control of the runtime rather than your program. Your program can always override this processing sequence if necessary but it usually means that you no longer have to process TAB/BACKTAB keys and other keys that move the cursor from field to field.

Field Input Sequence

Remember also that the sequence in which fields are processed is no longer fixed based on the use of keys. This is because you should allow a user to use the mouse to move around your panels if one is present and he wishes to do so. Of course, if you need to prevent your user from moving to a certain field before an earlier field is completed, this is quite possible by making that earlier field a control field and resetting the default next cursor location.

Keyboard Usage

Keyboard usage is often a concern for people converting existing systems: they do not want their users to have to learn too many new keystrokes, especially for such rudimentary things as moving from field to field. Traditionally the key to do this has often been the return key, as opposed to the tab key which is used in a lot of modern user interfaces. COBOL sp2 allow you the flexibility to use either key (or any other key, for that matter.)

Entry Fields

Another potential problem for existing users is keyboard usage within entry fields under graphical environments. The default system entry field may behave quite differently from the entry field created as a result of an accept statement. COBOL sp2 provides a custom data entry field that is a little more typical of traditional data entry fields.

Pushbuttons

Apart from entry fields, the other most common field type in a modern user interface is probably the pushbutton. A pushbutton is usually pressed (clicked on) by the user when it is time for the program to do some more processing, to save some information just entered, for example. This is exactly how function keys have traditionally been used, and for this reason, COBOL sp2 allows you to equate pushbuttons to function keys so that your program logic does not have to be changed to deal with them.

Menus

In modern user interfaces, menus often form only part of a window, rather than occupying a whole screen as in traditional systems. This allows a user to quickly perform different functions and access other parts of your system even in the midst of a regular data entry process.

These menus normally take the form of a Menu Bar across the top of a window, with optional pulldown submenus. Menu selections are returned into a data item within the generated copy file just like function key codes.

Of course this does not mean that you cannot use more traditional standalone menus to control processing flow within your system. Indeed it may be very easy to enhance an existing vertical list menu using a column of pushbuttons to replace the numbered menu options that you are perhaps using now.

USING COBOL sp2 IN CONJUNCTION WITH EXISTING CODE

As mentioned above, it may not be practical to convert all your existing screens to COBOL sp2. However, it may be very desirable to use COBOL sp2 to process additions and enhancements to the core interface. These additions might take the form of a new front-end logo and menuing system, or pop-up windows over existing data entry screens. These new features may be enough to satisfy your users' immediate needs, giving you time to implement a more thorough conversion.

A New Front-End

Developing a new front-end for your system is much like developing a completely new system: the new code will interact with the old code only in order to invoke portions of that old code. The old code will then operate as if the new code were not even present.

This technique works better in some situations than others. Although the new portion of the system will look different than the old system, it is important that it not look TOO different. Such a situation may confuse your users even though initial impressions may be good.

Pop-Up Windows

Using pop-up windows over your existing screens may be a really good option for you. Again, it is important that the new windows function in a somewhat similar manner to the old screens. However, because their use can be optional, the user is not forced to change his work methods unless he or she wants to. At the same time, you can demonstrate the new windows as evidence of your commitment to steadily improving and enhancing your software.

You will need to consult documentation supplied with each COBOL sp2 version to see what steps must be taken to ensure that COBOL sp2 can operate seamlessly with existing screens. In some cases, a window can be displayed and processed as if your old screen were not even present, in other cases some configuration work may have to be done.

Pop-up windows with COBOL sp2 are so easy because, in simple terms, every window is a pop-up window. A window is opened (usually automatically as a result of a converse-panel function call), user input into the panel is processed and then the window is closed (using the close-window function call.) Closing the window automatically restores any underlying screen data without further work on your part.

Choice Windows

One type of window that you could provide is a window that allows users to pick from a list of choices rather than key in the choice directly into a data entry field. This allows a user who is comfortable with the mouse to avoid too much unnecessary work with the keyboard. Perhaps you could enhance your existing accept code to detect a mouse click in a field and then pop-up a choice window using COBOL sp2. This choice window might take the form of a List Box with a fixed number of entries, a repeat group populated from an indexed file, or even just a very simple window containing a few radio buttons.

TERMINOLOGY

Here is a formal definition of the some of the terms used in the rest of the documentation. Some of these terms have already been mentioned above.

Panel file

The file in which COBOL sp2 stores panels. This file may be manipulated with regular operating system commands.

Panel

The form-type object through which your program may display and accept data. Panels are defined using the COBOL sp2 Panel Editor or by making calls to the runtime.

Window

An object defined in memory used to process panels. If a window is defined as visible it will be displayed on the hardware video device together with any panels assigned to it. Windows are created by making calls to the runtime. Normally a window will contain just one panel and so the window can be created automatically based on the appropriate panel definition.

Field

An object defined within a panel for the purpose of displaying, accepting or identifying an item of data. Fields are defined using the COBOL sp2 Panel Editor or by making calls to the runtime. Fields are also referred to as controls in some environments. Static fields are used for non-enterable panel text, labels for non-static fields and other non-enterable text display. Non-static fields are used for accepting user input and displaying non-static data.

Group

An object used to logically group non-static fields within a panel. Groups are defined using the COBOL sp2 Panel Editor or by making calls to the runtime.

Repeat Group

An object used to allow a non-static field or group of fields to be defined once and used multiple times within a panel. Used to create vertical and horizontal arrays. A repeat group may scroll if there are more occurrences than can fit into the area assigned to it within the panel. Repeat Groups are defined using the COBOL sp2 Panel Editor or by making calls to the runtime.

Color

The field, panel or window property controlling the foreground and background colors of the object. Colors are defined using the COBOL sp2 Panel Editor or by making calls to the runtime.

Font

The field or panel property controlling the display of text for that object. Fonts are defined using the COBOL sp2 Panel Editor or by making calls to the runtime.

Property

A specific attribute of an sp2 object that controls some aspect of the object. Properties can usually be set in the Panel Editor or at runtime.

WHAT'S IN THE REST OF THE DOCUMENTATION?

The COBOL sp2 User Documentation is divided into 2 volumes, the User Guide and the Reference Manual. You'll find the Table of Contents for both volumes at the beginning of the User Guide and the Appendices and Index for both volumes at the end of the Reference Manual. The User Guide consists of 3 parts and the Reference Manual consists of 2 parts. Each part has been assigned a letter and represents a major documentation topic. The remaining 4 parts are:

User Guide, Part B, The COBOL sp2 Panel Editor
User Guide, Part C, Programming with COBOL sp2
Reference Manual, Part D, COBOL sp2 Programming Functions
Reference Manual, Part E, COBOL sp2 Data Items

A Summary of the User Guide, Part B

The User Guide, Part B, The COBOL sp2 Panel Editor, provides a basic introduction to the Panel Editor and how to use the editor in general. If you encounter difficulty in creating, deleting, moving or changing the size of fields in the Panel Editor, you should refer to Chapter B2. This chapter explains the standard procedures to use when working with fields in the panel you are creating.

It also provides a series of step-by-step instructions on how to create several standard panels and panel fields which you will eventually use in your program. We highly recommend that you spend time reviewing Part B before attempting to develop user interface screens with COBOL sp2.

Part B will help significantly reduce your learning curve when using the Panel Editor.

A Summary of the User Guide, Part C

The User Guide, Part C, Programming with COBOL sp2, provides an overview of the concepts involved when programming with COBOL sp2. Part C is helpful when you first start using COBOL sp2.

This section of the documentation explains in general terms the standard procedures you will use when programming with COBOL sp2. Topics covered include how to display panels, pass data to panels, process data from panels and eventually close panels and end the interactive portion of your program.

Part C also provides step-by-step methods for enhancing the program which was generated from the panels painted in Part B.

A Summary of the Reference Manual, Part D

The Reference Manual, Part D, COBOL sp2 Programming Functions, lists the various COBOL sp2 functions used in controlling the panels from your COBOL program.

This section of the documentation explains those functions which must be used as well as those functions which may be used to achieve the results you desire. Part D provides information on functions used to control a default panel definition as well as those functions which may be used to dynamically change the default panel appearance and behavior. Also covered are those functions which allow you to dynamically create and maintain panels without using the Panel Editor.

A Summary of the Reference Manual, Part E

The Reference Manual, Part E, COBOL sp2 Data Items, lists the various COBOL sp2 data variables passed to COBOL sp2 by your program. These data items may be used by your program to dynamically detect and control panel events.

If you need to set or control a particular field or panel property and you're not sure how to proceed you should do the following:

First refer to the Index to obtain the reference location for the specific item.

The Index will direct you to a Data Item Definition which is located in the Chapters contained in Part E.

The entries in the various chapters of Part E will list the Data Item as well as the Function needed to set the item in your program. A brief description of the Data Item as well as the location in the Panel Editor where you can set the item as a default panel property is also included for each entry.

Appendix A

Appendix A lists the Decimal Values for keys used to control your panels from a COBOL application. The listing includes the actual decimal value along with a textual description of the key it represents.

Decimal key values are passed to your program by COBOL sp2 to allow you to determine the action desired by the operator of your program. In addition, the decimal key value is passed by your program to COBOL sp2 so that you may dynamically enable and disable specific keys during the operation of your program.

Appendix B

Appendix B lists the Environment Variables used to used to define the sort of processing that is needed.

Appendix C

Appendix C provides techniques for organizing panels within your panel files. It also describes the SP2CHECK utility which is used to compress and update panel files.

Appendix D

Appendix D lists the various error codes returned by the programming functions used with COBOL sp2.

Appendix E

Appendix E describes the COBOL sp2 Code Generator template file, called SP2.CBX. This template file can be modified to produce code that may be more suited to your own needs.

Appendix F

Appendix F lists the property ids used with the GET-PROPERTY and SET-PROPERTY functions and the property keys used with the Code Generator.

INSTALLATION OF THE COBOL sp2 DEVELOPMENT SYSTEM

COBOL sp2 files are shipped in a compressed, self-extracting installation executable file. The self-extracting installation executable allow you to install the files in an existing directory or a new directory which is created during the installation process. If you do not specify a directory, the installation executable will install the software in the default directory.

SYSTEM REQUIREMENTS

COBOL sp2 files will require approximately 10 megabytes of hard disk space.

GETTING HELP

The COBOL sp2 Documentation should provide answers to most of your questions. Please study the documentation carefully before calling for assistance. If after reviewing the User Guide and Reference Manual, you are still unsure of the solution to your problem, please send an email to the address on the firrst page of this User Guide.

CHAPTER A2 - An Introduction To Graphical User Interface Systems

What is a Graphical User Interface?

Graphical User Interfaces, also referred to as "Windowing Systems" or "GUI's" are user interface handling environments designed to make application programs easier to use for an individual with very little computer experience. Depending upon the specific application, GUI's can also give the computer operator greater productivity. GUI's can also help the experienced computer operator by allowing multiple tasks to be performed at one time.

Today, the traditional version of the GUI is commonly referred to as the "Desktop GUI". This is in contrast with a Web Browser user interface which is also of course a type of "GUI" but one which is under the control of a browser rather than running directly on the Desktop. A browser interface may run as a window within the desktop or as an full screen application on a mobile device.

Desktop vs. Browser

The Desktop allows applications to create windows in order to interact with the user - an application may use one or several windows with some open at the same time. The Browser allows an applications to create pages to interact with the user - an application will normally use multiple pages but only one page will be dislayed at any one time. In order to allow compatibility between these two different environments, sp2 (via the sp2 Web Client/X add-on product) allows pseudo-windows to be created within a page that mimic in many ways the behavior of regular desktop windows. This means that a COBOL program can essentially treat both interfaces in the same way which of course makes development a whole lot easier!

Refer to the Web Client/X documentation for more details on deploying to the Web and running in a web browser including how to customize the appearance of your application to give it a true Web look-and-feel.

Because most of the elements of the two environments are treated as one and the same by sp2, the rest of this manual will use the term GUI or Graphical User Interface to refer to both desktop and browser interfaces.

Advantages of a Graphical User Interface

Some of the ways in which GUI's help make applications easier to use are as follows:

- Use of a pointing device such as a mouse to allow the user to select an item in a window by pointing to it and clicking with a single button
- Use of small pictures, called "icons" to start programs instead of requiring the operator to remember a text-based command to initiate programs
- Use of menus which are highly standardized so that the application learning curve is greatly reduced

GUI's add to operator productivity for both novice and experienced operators by incorporating the following features:

- Use of single choice and multiple choice selection fields which force the operator to choose from a limited set of pre-validated options instead of requiring the operator to type in the text value of the various options
- Use of icons within an application to initiate a task which is complex or one which normally would require multiple steps for initiation
- Use of printing programs which operate independently of the application so that the operator can resume working before a printing task has been completed
- Use of multiple application windows to switch from one application to another quickly instead of starting and exiting programs continually

When an operator or customer requests such a user interface, many COBOL programmers are not quite sure where to begin. The GUI represents a radical change in the way user interfaces are designed and controlled by a COBOL application. Depending upon how the GUI is implemented in the program, it may also represent a completely different way of programming for the developer.

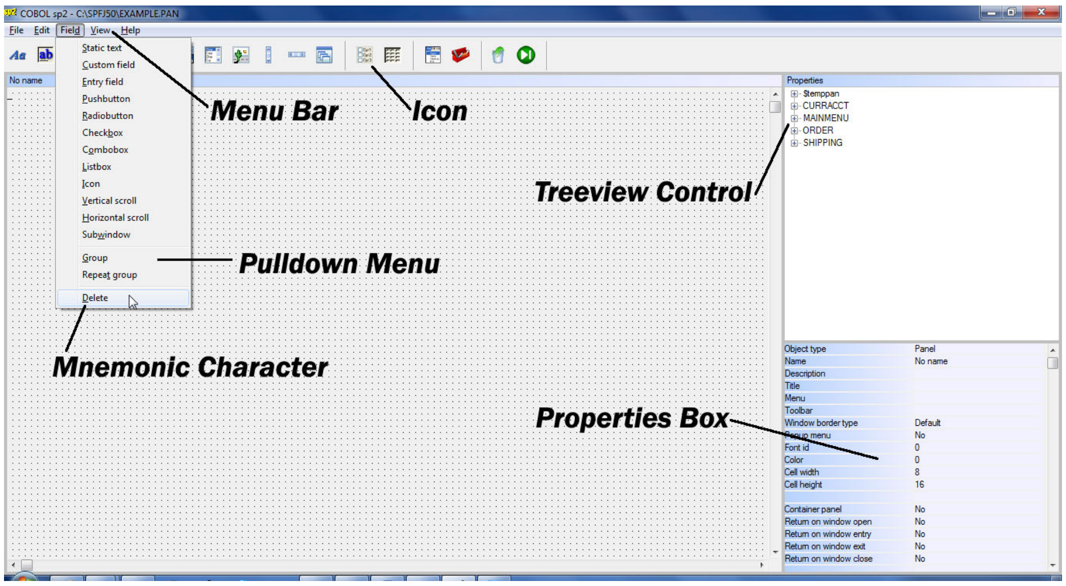
Perhaps the most positive aspect of the GUI environment for the programmer and the operator is the level of consistency in user interface design and usage. GUI environments generally share a similar set of field types and incorporate visual selection fields such as icons. This can make new applications easier to learn for the user. In addition, because most GUI applications have a similar appearance, system user interfaces can be easier to design for the programmer.

This is not so much the case with browser interfaces but because sp2 (via the Web Client/X product) for the most part replicates the behavior of the desktop GUI, some sense of consisitency is preserved even though it is not necessarily the rule.

New Terminology to Learn

One of the issues involved in providing GUI support for an application is the new terminology associated with the various environments. For example, in most GUI environments, fields are referred to as "controls." In order to provide the smoothest possible transition for the COBOL programmer, COBOL sp2 refers to all "controls" as "fields." Use of the traditional terminology should help to ease the transition for COBOL programmers, therefore throughout this documentation, "field" will be used instead of "control."

FIGURE 1



Object type	Panel
Name	No name
Description	
Title	
Menu	
Toolbar	
Window border type	Default
Popup menu	No
Font id	0
Color	0
Cell width	8
Cell height	16
Container panel	No
Return on window open	No
Return on window entry	No
Return on window exit	No
Return on window close	No

Menu Bar (See Figure 1)

The Menu Bar is a group of menu options arranged in a horizontal fashion at the top of the window. Menu Bar options may be selected using the keyboard keys, a pointing device (such as a mouse) or by pressing mnemonic keys on the keyboard.

Pull Down Menus (See Figure 1)

Pull Down Menus are a group of additional menu options which are subordinate to the Menu Bar options. Pull Down Menus display in a vertical fashion directly below the corresponding Menu Bar option. A Pull Down Menu is simply a sub menu to the Menu Bar option directly above it.

Mnemonics (See Figure 1)

Mnemonics are characters which help speed the movement of the cursor to a selection field or are used to initiate the choice in a selection field. Mnemonics are generally identified with an underline or with a highlighted attribute. Mnemonics can be used in Menu Bars, Pulldown Menus, Push Buttons, Check Boxes and Radio Buttons. Mnemonics can also be used in Field Labels to move the cursor quickly to a entry field.

Icons (See Figure 1)

Icons consist of a graphical image and an optional text label. Icons represent an object or initiate an action when selected with the mouse.

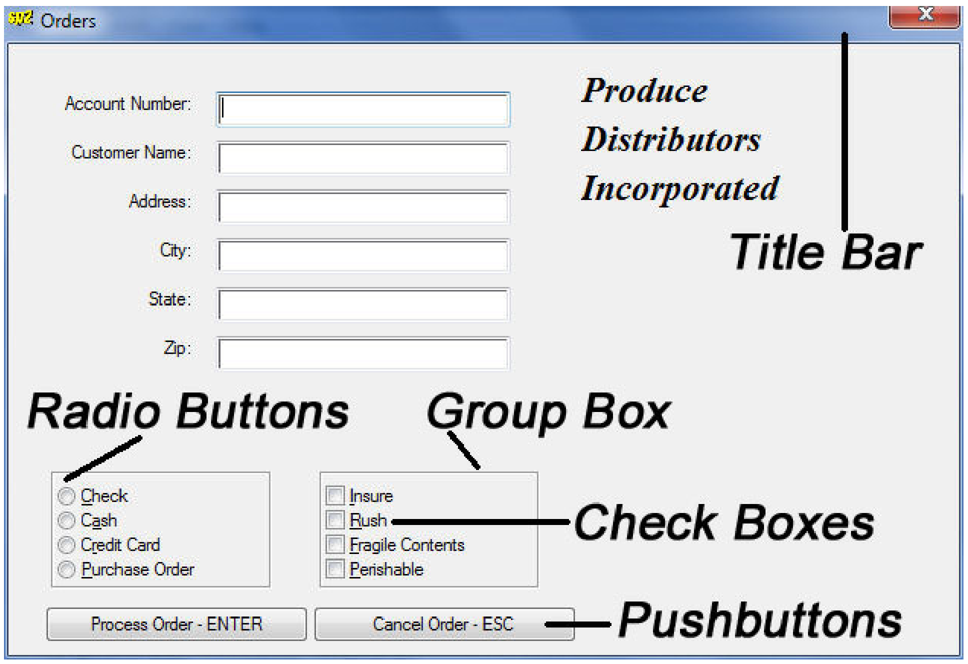
Treeview Controls (See Figure 1)

Treeview controls consist of a hierarchical set of directories, files or objects contained within a file. These objects could be records in a data file or as in the case of COBOL sp2, panel components, such as fields, push buttons, bitmap icons and other panel components.

Properties Boxes (See Figure 1)

A Properties Box shows the properties of the object which currently has focus. In the case of COBOL sp2, the Properties Box displays the properties of the panel or panel field which currently has focus.

FIGURE 2



Push Buttons (See Figure 2)

Push Buttons are fields labeled with text, a graphic or both. Push Buttons represent an action to be initiated when selected by the operator. Push Buttons may be selected with the keyboard or with a mouse.

Check Boxes (See Figure 2)

Check Boxes are square boxes with associated text representing a choice. If there is more than one Check Box, multiple selections may be made. When the Check Box is selected, the box is filled with an "x" or a check mark, indicating that the choice has been selected.

Radio Buttons (See Figure 2)

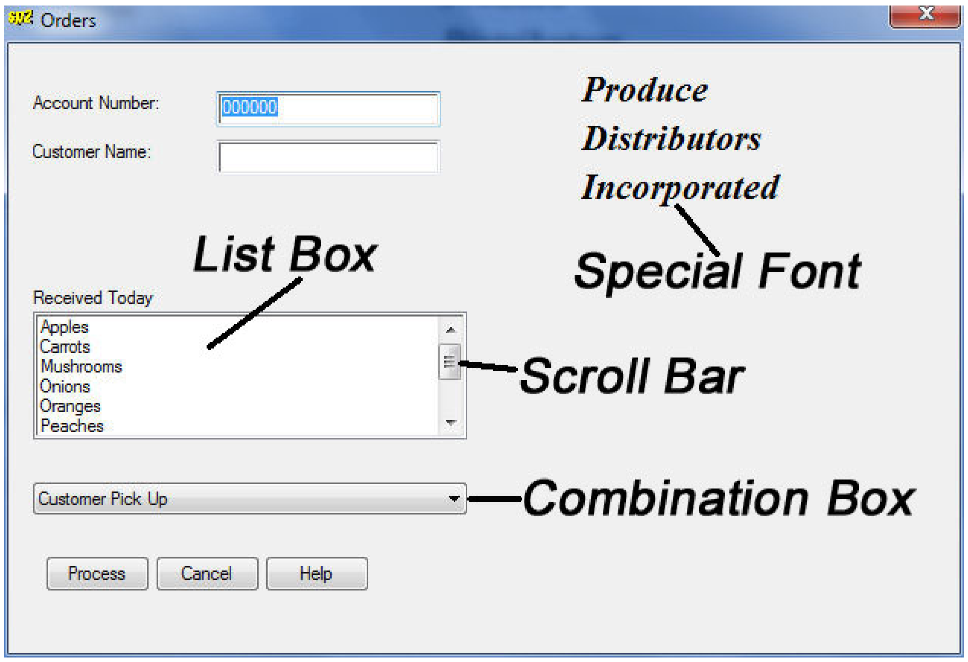
Radio Buttons are circles with associated text representing a choice. When the Radio Button is selected, the circle is partially filled, indicating that the choice has been selected. Radio Buttons only allow for a single choice, unlike Check Boxes which allow multiple choice selection.

Group Boxes (See Figure 2)

A Group Box is used to combine similar fields into a set. By default, Group Boxes have borders which visually separate each Field Group. For example, a set of Radio Buttons which represent various customer payment options may be grouped together to separate them from other Field Groups used to collect other types of information in the window.

Once the fields are combined into a group, the cursor tabbing behavior changes. The cursor will now tab from the first field in one group to the next field outside the field group.

FIGURE 3



Scroll Bars (See Figure 3)

Scroll Bars indicate to the operator that a data entry array or selection list contains more information than is visible in the window. Depending upon whether a horizontal or vertical Scroll Bar is displayed, the data will scroll either left and right or up and down. It is also possible to have both horizontal and vertical scroll bars for data which can be scrolled in any direction. Scroll Bars are controlled with the mouse, although they are active when the data is being scrolled with the keyboard.

List Boxes (See Figure 3)

List Boxes are selection fields which display a vertical arrangement of possible choices for the operator of the application. List Boxes include a scroll bar if the list of choices exceeds the displayable area in the window.

List Boxes also provide a convenient method for the operator of the application to search for the desired option within the List Box. List Boxes automatically sort the data in numeric and alphabetic order. In addition, when the first character (or a number) is typed through the keyboard, the List Box will automatically jump to the first occurrence of the character which was typed.

An example of a possible use for List Boxes is a State Code Selection Box. This List Box provides a list of two digit State Codes for selection by the operator. The selected State Code would then be transferred to a State Code Field.

Combination Boxes (See Figure 3)

A Combination Box, also referred to as a "Drop Down List" is a field which allows the operator of the application to display a list a of valid options under the field. Combination Boxes include a downward pointing arrow to the right of the field. If the operator wishes to change the default option displayed, the mouse pointer may be used in conjunction with the downward pointing arrow to display a list under the field.

If all of the contents of a Combination Box can be displayed in the Drop Down List simultaneously, the Scroll Bar in the drop down list will automatically disappear. When there are more options in the Drop Down List than can be displayed at one time, the Scroll Bar will be present.

Combination boxes may be selection fields which only allow the operator to choose a valid option or they may be enterable by the operator. When the field is enterable, the downward pointing arrow is not attached to the enterable field.

Fonts (See Figure 3)

A Font is a specific style of typeface used on the panel. Fonts can have a number of attributes including:

- typeface name
- size of the font
- font pitch
- weight
- italics
- strikethrough
- underline

For more information on Fonts, please refer to Chapter B20, Changing the Default Font Size and Font Type, for a more detailed description of Font attributes.

Pointer (See Figure 3)

The Pointer is the small arrow or other symbol which works in conjunction with the Pointing Device. The Pointer can change in appearance depending on its location within the GUI environment. The Pointer is typically an arrow, but may change to an "I-beam" when located in a text entry field. It will also change to a double arrow symbol when it is on a window border. This indicates that the Pointer can be used to re-size the window. When the application is running through a lengthy process, the Pointer will generally take the form of an hourglass (Windows) or a clock (OS/2).

Pointing Device

A Pointing Device is a device such as a mouse or trackball. The Pointing Device is used to move a pointer in the window for the purpose of cursor movement. The Pointing Device is also used to activate other windows within the GUI environment. The Pointing Device is used to make choices on selection fields, such as Radio Buttons. Pointing Devices are generally used in conjunction with Menu Bars, Pulldown Menus, Radio Buttons, Check Boxes, Push Buttons, Scroll Bars and Icons.

Graphical User Interface Development in COBOL

Many programmers assume that it is impossible to have a GUI in a COBOL application. This is simply not true. Most COBOL compilers support CALL statements from the COBOL application directly to the external Application Programming Interface (API) used by the GUI environments. By invoking API CALL statements within a GUI environment, such as Microsoft Windows, the COBOL program can display GUI windows.

What is Application Programming Interface (API) Code?

GUI interface windows are typically created and controlled through the use of Application Programming Interface (API) Code. The main problem associated with making API calls from a COBOL application, is that the API call syntax is not very "COBOL-oriented."

A typical GUI API call is much more like a C language program than a COBOL program. This creates several problems. The obvious problem with using API calls is that your programming staff must learn an entire new programming language to display GUI windows. In a large company, the training may be quite substantial in terms of classroom costs and staff time.

An additional concern is that the new language to be learned is far more cryptic than traditional COBOL syntax. COBOL was designed to be similar to a spoken language, not an abbreviated "code-oriented language." COBOL language programs are easier to maintain by large programming groups than C language programs, because the COBOL programming language is self-documenting. C language programs are not. In a programming environment with a large number of programmers, a specific program may be maintained by many programmers over the course of several years. Maintaining programs with embedded API calls will necessarily be more difficult which means more time spent on maintenance. This directly equates to a higher maintenance cost than if the programs were maintained entirely in COBOL.

Portability of API Code

Another problem with API code, is that API CALL statements designed for one GUI environment are not necessarily the same in another GUI environment. In order to maintain a high degree of flexibility for the future, many companies want their source programs to be hardware independent. The many different GUI environments available include Microsoft Windows, Microsoft Windows NT, OS/2 Presentation Manager, Motif, Open Look and DEC Windows. Their is no industry standard set of API CALL statements for the many GUI environments and therefore using API code to display GUI windows will limit the portability of an application program.

API CALL statements cannot be used if the application must run in a character-based environment, such as MS-DOS as well as a GUI environment. If it is acceptable to maintain one source program for character-based environments and four or five source code versions for the various GUI environments, then embedded API code for the GUI based interface is a good solution. If the objective is to keep the source code maintenance task as simple as possible, and the source code as portable as possible, the API code solution is a poor alternative.

Supporting International Versions Of An Application

Because many applications are distributed throughout the world, it is highly desirable to avoid inclusion of user interface definitions in the COBOL source program. If the user interface is developed using API code, multiple versions of the source program must be maintained in order to support more than one spoken language.

Multiple source code versions create maintenance problems. When modifying user interfaces defined within the source code, program bugs can be easily introduced. This can increase the amount of time needed to maintain the program user interface. In addition, the burden of the software support staff is increased if different versions of the software are released as user interfaces are translated to each spoken language.

Maintaining GUI window definitions that are external to the COBOL source application is therefore highly desirable for purposes of supporting an application to be distributed internationally.

Adding a GUI to Your COBOL Programs

It would appear that implementing GUI windows in your COBOL program is highly undesirable. Using API CALL statements to implement GUI windows does create a number of significant problems. How does the COBOL programmer accomplish the task of implementing GUI windows without creating additional problems?

Traditional PC COBOL uses DISPLAY and ACCEPT verbs to control user interface display. The COBOL language does not currently support an ANSI standard syntax for displaying GUI windows. DISPLAY and ACCEPT verbs are capable of only displaying character-based user interfaces for COBOL applications. This necessitates a different approach to displaying a GUI window from COBOL. The best approach for the display and control of GUI windows is through the use of an external program dedicated to COBOL user interface handling.

COBOL sp2 is dedicated to adding a Graphical User Interface to your COBOL application.

COBOL sp2 provides a "Panel Editor" to quickly develop and test individual windows for the COBOL user interface. COBOL sp2 uses an ANSI standard COBOL "CALL USING..." statement to display windows from the COBOL application.

CHAPTER A3 - Deploying Your Interface to Other Environments

Although user interface development with COBOL sp2 is done in a regular graphical environment (i.e. Microsoft Windows), your interface can be deployed to other environments that sp2 supports. There are currently three options:

- Internet/Intranet
- Web browser
- Character-mode

Internet/Intranet

This option, called Thin Client, allows you to run your interface on a Windows client but run your application on a server connected by TCP/IP and running Windows or Unix. This is the simplest of the alternative environments to deploy to because, using the Thin Client package, your sp2 application can be run in this environment without any change i.e. it will run across the Internet or an intranet as if it were running on a standalone Windows machine. Because of this, there are really no restrictions or special techniques to consider when developing for this environment.

Contact Flexus in order to purchase the Thin Client package or to find out more about it.

Web Browser

Two options are provided here. The first is called Web Client and allows you to run your interface as a regular Web interface using a combnation of HTML and Javascript. The HTML is generated from the panels which you define and a special runtime is used to communicate between your application and the HTML code.

The second option is called Web Client/X and is the preferred option because it provides a very high level of compatibility between your application running in a desktop and in a browser. It uses Javascript to dynamically render the interface rather than generating HTML to do this.

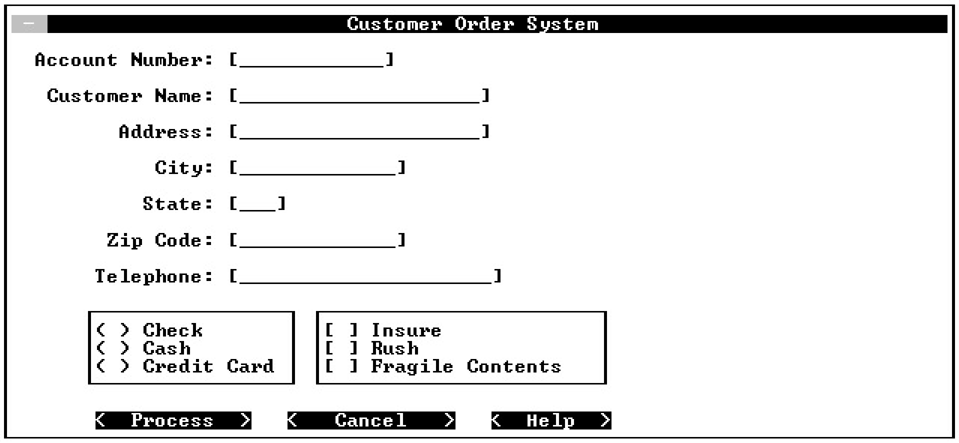
Contact Flexus in order to purchase the Web Client or Web Client/X package or to find out more about them.

Character Mode

Almost all the features available in graphical mode are faithfully reproduced by a COBOL sp2 text mode runtime. This includes such features as moveable, resizable windows, pulldown menus, standard controls and extensive mouse support.

Of course, not all features can be reproduced: graphical icons and fonts are not supported; colors are not supported if you have a monochrome terminal; mouse support may not be available. Nevertheless, these shortcomings do not prevent the development of a text mode user interface that has a graphical user interface feel. Moreover, even if you do develop a graphical user interface containing some of these unsupported features, the same interface will probably at least be operable in text mode.

Following is a text mode equivalent of one of the screen images shown in the previous section:



```
 -                      Customer Order System
 Account Number: [____________]
  Customer Name: [_____________________]
        Address: [_____________________]
           City: [_____________]
          State: [___]
       Zip Code: [_____________]
      Telephone: [______________________]

       ( ) Check           [ ] Insure
       ( ) Cash            [ ] Rush
       ( ) Credit Card     [ ] Fragile Contents

       <  Process  >    <   Cancel   >    <  Help  >
```

Developing For Character Mode

If you need to develop an interface that works in both graphical and text modes, develop it in graphical mode and get it looking as fancy as you want. The only things you should avoid are using non-standard fonts and altering the standard cell size. These two items are discussed later in this section.

After you have developed the graphical version, check out how it looks in text mode by running the generated test program with a text mode runtime. The one problem you may encounter if you have been using the standard proportional font (the default) is that some of your static text items may be truncated. Try switching the panel font to a fixed pitch font in the editor to get a pretty good impression of what needs to be done.

In order to avoid the proportional font pitfall, you may want to consider initially developing your panels using the standard fixed pitch font to ensure that your fields will be spaced out sufficiently. Once this has been done, you can switch back to a proportional font to achieve the appropriate graphical look-and-feel. If you use this technique, avoid using too many upper-case characters because upper-case characters actually take up less space when displayed with a fixed font than if they had been displayed with a proportional one.

The following is a list of things that you should bear in mind when developing for text mode:

Menu Bar

The menu bar can be accessed with the mouse or using the F10 or ALT keys. If these keys are not available on your terminal, use the CTRL+A key instead.

Combination boxes

To display the Combination Box pulldown list using the keyboard, use the CTRL-DOWN ARROW key. If this key is not available on your terminal, use the CTRL-O key instead.

Pushbutton mnemonics

To activate a pushbutton using a mnemonic (the highlighted character within the pushbutton text), use the ALT-MNEMONIC key. If this key is not available on your terminal, press the CTRL-B key first and then press the mnemonic key itself. (This rule also applies to running your own program if it uses pushbutton mnemonics.)

Fonts

Special fonts are not supported in text mode. All text will be displayed in the default text mode fixed font. If panels have been developed in a graphical environment using proportional fonts, some static text items may appear truncated.

If you have designed a panel in graphical mode using a small font, there may not be enough room to display all the text. In this case the panel may have to be redesigned.

Cell size

Whatever the cell size selected, all fields will be positioned in text mode with respect to the regular character cells. For this reason, it is best to use the default cell size (16 pixels by 8 pixels) if you are developing in graphical mode and plan to use a panel in text mode.

Colors

Only the standard 16 text-mode colors are supported. If an RGB color has been defined in graphical mode, the nearest text mode color will be used but this is not normally a problem.

Graphical environments usually have color schemes that can be set by the user outside of any particular application. COBOL sp2 emulates this concept in text mode by providing default colors for screen-based objects. If your video hardware has no color capability, an alternative highlighting scheme is used. The defaults are as follows:

Object	Color	Monochrome

Screen	White-on-black	Normal
Window	White-on-blue	Normal
Window border *	White	Normal
Title Bar	Blue-on-cyan	Reverse
Button	Black-on-white	Reverse
Highlighted button	White-on-black	Normal
Menu Bar	White-on-red	Normal
Menu mnemonic *	Yellow	Bright
Menu highlight	White-on-black	Reverse
Scroll Bar	Black-on-white	Reverse
Scroll Bar slider	Black-on-cyan	Normal
Selected item	Black-on-white	Bright
Highlighted item	White-on-black	Reverse
Pushbutton	Blue-on-cyan	Normal
Highlighted pushbutton	White-on-black	Reverse
Highlighted slider	White-on-black	Normal
Button mnemonic *	Yellow	Bright

* indicates that only the foreground color is relevant here

If you or your user prefer a different default color scheme, it can be changed by creating a file called SYSCOLOR.SP2 or SYSMONO.SP2 depending whether you are running in color or monochrome. The file is a regular text file with one line in it containing 3-digit color codes for each of the above screen objects. The codes must be in the correct order and have one blank between them. The following is a sample entry reproducing the default color scheme:

```
007 023 023 049 112 007 071 014 007 112 048 112 007 049 007 007 014
```

The codes are formed by adding together one number from each of the foreground and background columns:

Color	**Foreground**	**Background**
Black	000	000
Blue	001	016
Green	002	032
Cyan	003	048
Red	004	064
Magenta	005	080
Brown	006	096
White	007	112
Grey	008	128
Bright-blue	009	144
Bright-green	010	160
Bright-cyan	011	176
Pink	012	192
Bright-magenta	013	208
Yellow	014	224
Bright-white	015	240

Icons

Graphical icons are not supported in text mode. An icon field will by default be displayed as periods in text mode. You can alter this behavior in various ways. The first way is by setting the first 8 bytes of the initial value of the field If the value is set to "NULL", nothing will be displayed. If any other text is keyed in, that text will be displayed centered within the icon. This is how the panel editor icons are handled.

An alternative method for controlling icon display in text mode is to use one of the panel-level text mode options, discussed later in this section. Using one of these options, you can automatically cause all display of icons in text mode to be suppressed.

System-Default Entry Fields

This type of field is handled in text mode as if it were a custom field.

Multi-line Entry Fields

This type of field is not currently supported in text mode so its use should be avoided if you need to run in text mode.

Text Mode Options

In order to simplify running in both graphical and text modes using the same panel or set of panels, a number of options are provided that only apply when running in text mode - in graphical mode, these options are ignored.

The options are set as follows:

No field borders

It is normal in graphical mode for entry fields to have a border around them. This border is simulated in text mode using square brackets at the start and end of each field. This option automatically suppresses the display of these symbols.

No pushbutton borders

A pushbutton is simulated in text mode using less than and greater than signs at the start and end of the field. This option automatically suppresses the display of these symbols.

No icon display

This will suppress any display of simulated icons.

Auto-size window to fill whole screen

This will cause a panel to fill up the whole screen rather than be sized as it was originally defined.

No window border

This is usually used with the previous option to suppress the window border which is not essential if the panel takes up the whole screen.

No system menu

This suppresses the system menu and minimize/maximize icons found at the start and end of the Title Bar. This essentially suppresses resizing of the window.

No Title Bar

This suppresses the Title Bar completely. The previous option is not referenced if this option is set.

Current field color

This causes the current field (the field with the cursor) to be highlighted in text mode only - no highlighting will be done in graphical mode.

PART B

The COBOL sp2 Panel Editor

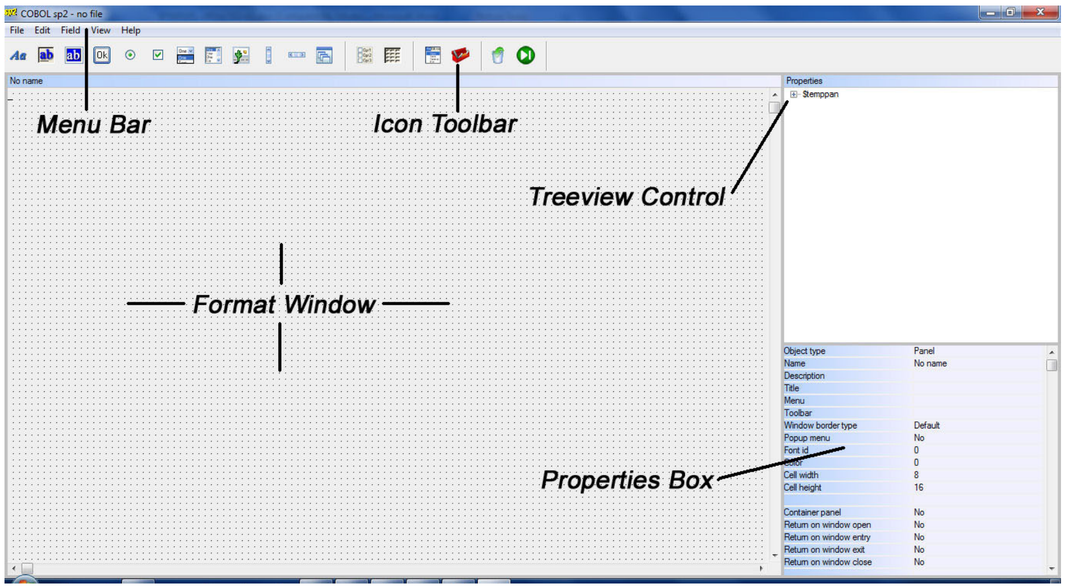
CHAPTER B1 - An Introduction to the COBOL sp2 Panel Editor

Launch the COBOL sp2 Panel Editor

Invoke the COBOL sp2 program by positioning the mouse pointer on the COBOL sp2 icon and double clicking the left mouse button as you would invoke any program from a GUI environment.

The COBOL sp2 logo window will display for a few moments. The COBOL sp2 Panel Development Tool will then display.

Components of the COBOL sp2 Panel Editor



Object type	Panel
Name	No name
Description	
Title	
Menu	
Toolbar	
Window border type	Default
Popup menu	No
Font id	0
Color	0
Cell width	8
Cell height	16
Container panel	No
Return on window open	No
Return on window entry	No
Return on window exit	No
Return on window close	No

As you can see in the panel image on the previous page, there are five main components of the COBOL sp2 Panel Editor. The components are as follows:

- The Format Window
- The Icon Tool Bar
- The Menu Bar
- The Treeview Control
- The Properties Box

The Interrelationships between the Five Components

Each of the five components will be addressed individually later.

The Format Window is your work area. This is the area where you:

- specify the location of your panel fields,
- resize the default panel fields to suit your needs and
- combine various panel fields to complete the appearance of the user interface for your COBOL application.

The Icon Tool Bar is used to select various panel fields for creation and placement in the Format Window. The Icon Tool Bar should significantly speed completion of the user interface design if you prefer to use a mouse.

For example, assume that you would like to place a data entry field on the proposed panel. The first step to accomplish this task would be to move the mouse pointer to the third icon on the Tool Bar and click once. You would then move the pointer to the desired location in the Format Window and click again. A default data entry field will then appear in the Format Window automatically.

As mentioned previously, you could then easily change the location or size of the default field within the Format Window.

The Menu Bar is also available to create panel fields in the Format Window. The Menu Bar also provides access to those panel design tools which are not available on the Icon Tool Bar.

For creation of standard panel fields, it is suggested that you use the Icon Tool Bar for faster panel design, but if you prefer not to use a mouse when designing the panel, the Menu Bar provides you with all of the same panel design facilities.

The Menu Bar is also used for accessing panel file management tasks and tools not available on the Icon Tool Bar such as opening panel files, saving your work to disk and modifying the size and position of the panel.

The Treeview Control can be used for several purposes. When a panel file is opened, panels listed on the Treeview Control may be opened in the Format Window of the COBOL sp2 Panel Editor by clicking on the panel name listed in the Treeview Control and "dragging" the panel into the Format Window.

In addition, when updating the properties of fields in a panel, it may be easier for you to place focus on a specific field by selecting it from the Treeview Control than selecting it from within the Format Window.

The Properties Box allows you to specify the properties of the panel fields once they are placed in the Format Window. In the example of the data entry field just created, the developer could easily specify:

a) a COBOL field name,
b) initial display value for the data entry field,
c) COBOL picture clause and
d) whether the field is an alphanumeric, numeric or date field.

This is all accomplished by typing the appropriate values within the appropriate property in the Properties Box.

The properties within the Properties Box will dynamically change depending upon the specific type of field on which you are setting the properties.

In order to enter or select a property for a specific field or field group, the cursor in the Format Window must be on the field or field group for which you want to establish the property.

Using the COBOL sp2 Panel Editor

The COBOL sp2 Panel Editor may be navigated with the keyboard, the mouse or both the keyboard and the mouse. Although the keyboard must be used when specifying field captions, field formats and initial values, using the mouse as much as possible will greatly speed the process of screen development.

At any point during the development process, you may minimize, maximize or restore the COBOL sp2 Panel Editor using the icons in the upper right corner of the main COBOL sp2 window. You may also use the system icon in the upper right to perform these functions.

Because the COBOL sp2 Panel Editor has been designed using a "Windowing approach", it is also possible to move any window or change the size of any window in order to suit your specific needs. Please bear in mind, however, COBOL sp2 does not change the size of the text when the size of the window is changed.

Accessing the COBOL sp2 Menu Bar

With the Mouse pointer

Click on the desired menu bar option. The corresponding pull down will appear, allowing you to click on the desired choice.

With the Keyboard

Use the (ALT) or (F10) key to activate the menu bar. Press the (DOWN ARROW) or (ENTER) key to display the corresponding pull down menu. Use the (ARROW) keys to move to the desired choice and select the choice using the (ENTER) key.

Accessing the COBOL sp2 Icon Tool Bar

The Icon Tool Bar is used for several purposes in the COBOL sp2 Panel Editor. The Icon Tool Bar can be used to create fields and field groups in the Format Window as well as delete them. In addition, menu bars and toolbars may be created using the Icon Toolbar. Accessing the Icon Tool Bar can be accomplished as follows:

With the Mouse pointer

Position the mouse pointer on the desired icon and click the left mouse button once. The pointer symbol may change, depending upon the specific icon selected. See the section on Pointer Symbols in the COBOL sp2 Icon Tool Bar below for more information on those icons which change the pointer symbol.

With the Keyboard

The Icon Tool Bar must be activated with the mouse. All tools available on the Icon Tool Bar are also accessible through the Menu Bar, if you prefer to use the keyboard instead of the mouse.

Pointer Symbols in the COBOL sp2 Icon Tool Bar

The Icon Tool Bar uses two different pointer symbols. The default pointer symbol, which is a diagonally pointing arrow, is used to initially select the icon.

After the Icon has been selected, the pointer will change to an action pointer for those icons which are used to:

a) create a field in the Format Window, or
b) select a field in the Format Window for further definition.

When the pointer symbol changes to the action pointer (vertically pointing arrow or a four-way pointing arrow, depending upon your operating environment), it is an indication that action is needed in the Format Window. The action to be taken depends on the specific icon selected.

The pointer will remain as a diagonally pointing arrow for those icons which require further definition of field or group property in the Properties Box. These properties, or rules, are set in the appropriate property of the Properties Box. The pointer will also remain as the default symbol when the Menu Bar Icon is selected. Selection of this Icon will automatically display the Menu definition pop-up window with a set of default menu options or the specific menu options assigned to the current panel.

Actions to be Taken After Selection of Specific Icons

The icons which change to an action pointer when selected and the specific action to be taken include:

Icon	**Action to Be Taken**
Static Text	Create Static Text in Format Window
Custom Field	Create Custom Field in Format Window
System Default Field	Create System Default Field in Format Window
Push Button	Create Push Button in Format Window
Radio Button	Create Radio Button in Format Window
Check Box	Create Check Box in Format Window
Combination box	Create Combination box in Format Window
List Box	Create List Box in Format Window
Icon	Create Icon in Format Window
Vertical Scroll Bar	Create Vertical Scroll Bar in Format Window
Horizontal Scroll Bar	Create Horizontal Scroll Bar in Format Window
Group Box	Create Field Group Box in Format Window
Repeat Field	Create Repeat Field Box in Format Window
Delete Field	Delete Field in Format Window

The icons which remain as the default pointer symbol when selected and the action to be taken, include:

Icon	**Action to be Taken**

Menu Bar	Create a New Menu Bar or Modify an Existing Menu Bar
Toolbar	Create Toolbar

Accessing the COBOL sp2 Properties Box

With the Mouse pointer

Using the mouse, position the pointer on the field in the Format Window for which you want to establish a specific property. Click the left button once to position the cursor. The Properties Box will then display the properties for that specific field.

If you are setting general panel-level properties, position the pointer in an area of the panel where no fields, groups or repeat groups exist and click the left button once to position the cursor. You may also double click or press the (ENTER) key on the panel description located in the Treeview Control. The Properties Box will then display panel-level properties. You may also double click or press the (ENTER) key on the panel description in the Treeview control.

Click on a specific property value to set that property. The values can either be typed in directly or the valid values are displayed in a drop down list - click on one of the options to set that option. In some cases multiple options can be set for a single property - in this case, clicking an option will toggle it on and off.

With the Keyboard

When the cursor is in the Format Window, you may access the Properties Box by pressing the (ESC) key once. The Panel Editor will attempt to maintain the cursor in the last property accessed when you toggle back and forth between the Format Window and the Properties Box. This works best when going from one field to next field of a similar type. When you access a series of dissimilar fields, the cursor may jump to the top property in the Properties Box because the properties in the Properties Box change depending upon the field which currently has focus in the Panel Editor.

To access other properties in the Properties Box, press the (UP ARROW) or (DOWN ARROW) key to move the cursor.

The property values can either be typed in directly or the valid values are displayed in a drop down list by pressing the space bar- press the space bar on one of the options to set that option. In some cases multiple options can be set for a single property - in this case, pressing space bar will toggle an option on and off.

Pointer Symbols in the COBOL sp2 Properties Box

The Properties Box uses two different pointer symbols. The default pointer symbol, which is the I-Beam pointer is used for data entry fields.

The second pointer symbol, the diagonally pointing arrow, is used to make selections in those properties within the Properties Box which provide two or more selectable options in a drop down list.

Accessing the COBOL sp2 Format Window

With the Mouse Pointer

Using the mouse, position the pointer in the desired location within the Format Window. Click the left mouse button once to position the cursor. To move the cursor to an area of the Format Window which is not currently visible, click the left button on the horizontal or vertical scroll bar within the Format Window.

You can also scroll the Format Window by clicking and holding the left mouse button while dragging the pointer left, right, up or down. The Format Window should automatically scroll when the pointer is dragged outside of the Format Window.

With the Keyboard

If the cursor is already within the Format Window, you may locate the cursor by using the (ARROW) keys. If the cursor is in the Properties Box, press the (ESC) key to locate the cursor in the Format Window. Please note that pressing the (ESC) key while the cursor is in the Properties Box will cause you to lose any changes you may have made in the last field in which the cursor was residing when the (ESC) key was pressed.

If the Menu Bar has been activated and a menu bar option is highlighted, pressing the (ESC) key twice will place the cursor in the Format Window if the cursor was in the Format Window prior to Menu Bar activation.
If a pulldown menu is displayed, the (ESC) key must be pressed three times to access the Format Window if the cursor was in the Format Window prior to Menu Bar activation.

When the cursor jumps to the Format Window, it will always be located in its prior location in the Format Window.

Pointer Symbols Used with the COBOL sp2 Format Window

The Format Window uses several different pointer symbols. The default pointer symbol, which is a diagonally pointing arrow, is used to position the cursor and select fields in the Format Window for moving, resizing or property modification.

The action pointer is used in the Format Window to locate a field or select a field in the Format Window for further definition. The action pointer is displayed as a result of clicking the mouse on the Icon Tool Bar.

The pointer will change to a double arrow symbol when a field has been selected and is displayed with a reverse video border, and when the pointer is directly on the reverse video border.

The double arrow symbol indicates that the mouse may be used to change the width and/or height of a field. There are three general types of double arrow symbols as follows:

Pointer Symbol	**Action to be Taken**
Horizontal Double Arrow	Change the width of the field
Vertical Double Arrow	Change the height of the field
Diagonal Double Arrow	Change both the width and height of the field

Accessing the COBOL sp2 Treeview Control

With the Mouse Pointer

Using the mouse, position the pointer on the item which you want to establish as the current object. Double click the left mouse button.

With the Keyboard

Press the (F2) key to access the Treeview window. Press (ENTER) to select an object.

CHAPTER B2 - How to Use the COBOL sp2 Panel Editor

What is a COBOL sp2 Panel File?

COBOL sp2 stores panel information in data files called panel files. These panel files contain the images of the panels you create as well as all of the information about the panel fields contained on each panel.

COBOL sp2 allows you to store one panel per panel file or many panels in a file. You can create any number of panel files you desire but it is quite normal to use just one file for a whole application. A panel file may contain both full size panel definitions as well as smaller pop-up window definitions. (While actively maintaining a panel file, it is useful to run it through the sp2check program every month or so to prevent a build-up of free space in the file - see Appendix C.)

These panel files will eventually be used and loaded into memory by your COBOL application when you make a CALL to COBOL sp2. In order to design your panel images, it is necessary to first create a panel file. Once the panel file is created, it may be opened later so that you may add additional panels or modify existing panels.

Before you begin designing your panels, you must first create a new panel file or open an existing panel file.

Create a New COBOL sp2 Panel File

To create a new panel file select the File/New file Menu Bar Option.

The COBOL sp2 Menu Bar may be accessed with the mouse by positioning the pointer on the menu options and clicking the left mouse button. If you prefer to use the keyboard to access the menu bar, press the (ALT) key or (F10) key to activate the menu bar. Use the (ARROW) keys to navigate the menu bar and pull down menus. When you have highlighted the New file pull down option, press the (ENTER) key.

The New File pop-up window will be displayed allowing you to establish the file name and directory in which the file will reside. There are no restrictions on file naming conventions for your panel files other than those limitations imposed by your operating system/environment.

At this time, create a panel file with the name, SAMPLE.PAN. Leave the Directory field blank so that the panel file will be created in the directory in which COBOL sp2 resides. Press the (ENTER) key or click on the OK Push Button to create the panel file.

Panels in COBOL Copy Files

If you are maintaining a system that is the result of a Flexus conversion project it is possible that your system is using panel definitions stored in copy files taher than in panel files. To access these panels in the editor you need to use the File/Import menu option and enter the name of the copy file to be imported. If you have not yet opened or created a panel file as explained above you will also be prompted for the name of a panel file. This panel file is needed to store the panel while it is being edited even though the file will not be used at runtime - you will regenerate the copy file for that purpose.

How to Use the COBOL sp2 Format Window

The following will describe how to access the COBOL sp2 Format Window and arrange panel fields to help you design your panels.

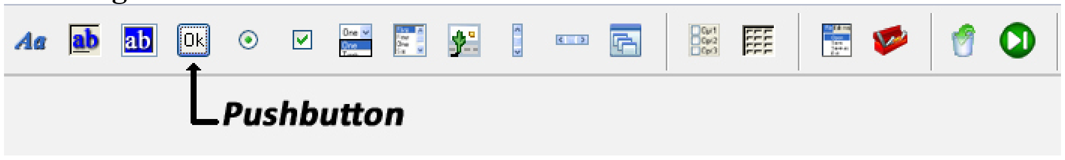
The Format Window is your work area. This is the area in which you arrange your panel fields for a window that will eventually be displayed in your COBOL application.

To scroll the format window, either click on the vertical and horizontal Scroll Bars or use the PGUP/PGDN keys to scroll up/down and the HOME/END keys to scroll left/right.

Working with Fields in the Format Window

Fields are easily created, sized and arranged in the Format Window with your mouse or the keyboard. When working with fields in the Format Window, avoid overlaying one field with another.

Creating a Field in the Format Window



Using the Mouse

Choose the Push Button icon (pictured above) from the Icon Toolbar and select it by positioning the pointer directly on the icon and clicking the left mouse button. Move the mouse pointer to bottom left corner of the Format Window. This is the location in which the Push Button is to appear. Click the left mouse button once more. A default representation of the selected field will appear in the Format Window.

Using the Keyboard

When creating a field with the keyboard, the cursor should be positioned to the desired location in the Format Window before the field is created. Otherwise, the field will be automatically created in the current Format Window cursor position.

Once the cursor is properly positioned in the Format Window, press the (ALT) or (F10) key to activate the Menu Bar. Next, press the "d" mnemonic key or use the (ARROW) keys to display the Field Pull Down Menu. Position the highlight on the desired field type and press the (ENTER) key to create the field.

Selecting a Field in the Format Window

Using the Mouse

To select a field in the Format Window, double click the left mouse button while the pointer is directly on the field. COBOL sp2 will then surround the field with a reverse video border, indicating that the field has been selected. If you wish to use the keyboard to select a field, first position the cursor on the field, then press the (ENTER) key to select it.

Using the Keyboard

To select a field in the Format Window with the keyboard, first position the cursor directly on the field.

Once the cursor is located on the field, press the (ENTER) key. COBOL sp2 will then surround the field with a reverse video border indicating that the field has been selected.

You may also select a field by double clicking on the desired field in the COBOL sp2 Treeview Control.

De-selecting a Field in the Format Window

Using the Mouse

When using the mouse to de-select a field, click once on the selected field. You may also move the mouse outside of the Format Window to de-select the field.

Using the Keyboard

A field which has been selected in the Format Window may be de-selected by pressing the (ESC) key.

Changing the Size of a Field or Group in the Format Window

Using the Mouse

Select the Push Button with the mouse. This is accomplished by double clicking the left mouse button while the pointer is directly on the field. COBOL sp2 will then surround the field with a reverse video border, indicating that the field has been selected.

Move the mouse pointer to one of the sides or corners of the reverse video border surrounding the field. As soon as the pointer changes to a double arrow symbol, you may change the height, width or both the height and width of the field. To change the width of the Push Button, position the pointer on the side of the selected Push Button until the pointer is a horizontal double arrow. Click and hold down the left mouse button.

Modify the width of the Push Button by moving the mouse pointer to the left or right while continuing to hold the left mouse button. When the desired Push Button width has been achieved, release the left button on the mouse.

Using the Keyboard

If you want to use the keyboard to change the size of fields in the Format Window, first locate the cursor directly on the field you wish to re-size with the (ARROW) keys.

Press the (ENTER) key to select the field. The field will be surrounded by a reverse video border, indicating that it was selected.

Changing the size of a field with the keyboard is a two step process. The first step is to press one of the four key combinations below, depending upon which side of the field you wish to change. When you press the key combination below, the "focus" changes to the appropriate side. You may then proceed to step two.

Key Combination	**Focus is on:**
(CTRL-UP ARROW)	Top of Field
(CTRL-DOWN ARROW)	Bottom of Field
(CTRL-RIGHT ARROW)	Right Side of Field
(CTRL-LEFT ARROW)	Left Side of Field

Step two allows you to actually increase or decrease the size of the field depending upon which key combination is pressed.

The following table indicates the result based upon the focus and the key combination pressed.

Focus is on the Top of the Field	
(CTRL-UP ARROW)	Increase the Height of the Field
(CTRL-DOWN ARROW)	Decrease the Height of the Field
Focus is on the Bottom of the Field	
(CTRL-UP ARROW)	Decrease the Height of the Field
(CTRL-DOWN ARROW)	Increase the Height of the Field
Focus is on the Right Side of the Field	
(CTRL-RIGHT ARROW)	Increase the Width of the Field
(CTRL-LEFT ARROW)	Decrease the Width of the Field
Focus is on the Left Side of the Field	
(CTRL-RIGHT ARROW)	Decrease the Width of the Field
(CTRL-LEFT ARROW)	Increase the Width of the Field

Using the Properties Box

If you want to use the Properties Box to change the size of fields in the Format Window, first select the field which you want to modify and then set the width and height properties located in the COBOL sp2 Properties Box.

PLEASE NOTE: Although it is possible to change the size of virtually every field in COBOL sp2, it may not be necessary in all cases. When the default caption for Static Text, Push Buttons, Radio Buttons and Check Boxes is increased in size, the field will increase in width automatically.

These fields will automatically increase in width to accommodate changes in the width of the caption or the width of the font used for the caption. This occurs when the caption exceeds the current field width. The fields will not automatically increase in height to accommodate a larger *font style. If you wish to use a larger font, you must manually increase the height of the field as appropriate.*

In addition, a decrease in the length of the caption will not automatically decrease the width of the field. In some cases, the appearance of the panel is improved if all fields in a field group (such as Push Buttons) are equal in length. If the field width were automatically decreased, it would be impossible to have equal width fields with variable length captions.

Working with Cell Rows and Cell Columns

The Format Window of COBOL sp2 contains a grid to help you change the location fields and change the size of fields appropriately.

Using the top reverse video border to change the size of a panel field may not produce the precise size that you may desire. After you have selected a field to change the size, you may wish to use the bottom border to change the height of a field.

The reason you may wish to use the bottom of the reverse video border to change the height is because COBOL sp2 uses the top of the field to establish the default row position for the field in a character based environment. When you use the top border to change the size of the field, it will always "snap" itself to the top of the cell row. By using the bottom border to change the size, you will be able to change the field height more precisely.

Moving a field to a new location on the panel is perfectly acceptable, because the top of the field will "snap" to a cell row in the grid. If you need finer control over field position that the default grid allows, you may change this grid size at any time using the Properties Box.

Aligning Fields

Once an area has been selected, the fields within that area may be aligned using the Edit/Align menu option. Fields can be aligned with respect to the first field in the selected area either horizontally (left/right/center) or vertically (top). Fields can also be resized to the first field and spaced horizontally or vertically by a specified amount.

Moving a Field in the Format Window

Using the Mouse

Select the Push Button field on the Format Window by double clicking the left mouse button while the pointer is directly on the Push Button field. COBOL sp2 will then surround the field with a reverse video border, indicting that the field has been selected.

Move the mouse pointer inside the reverse video border surrounding the field, then press and hold down the left mouse button. The mouse pointer must be inside the border in order to move the field. If the pointer is directly on the reverse video border, the field will be resized instead of moved.

Change the location of the Push Button by moving the mouse pointer while continuing to hold the left mouse button. The reverse video border will follow the pointer as long as the left mouse button is depressed. When the reverse video border for the field has been moved to the desired location, release the left mouse button. This process results in "dragging" the field to the desired location.

Using the Keyboard

If you want to use the keyboard to move fields on the screen, first locate the cursor directly on the field you wish to move with the (ARROW) keys.

Press the (ENTER) key to select the field. The field will be surrounded by a reverse video border, indicating that it was selected.

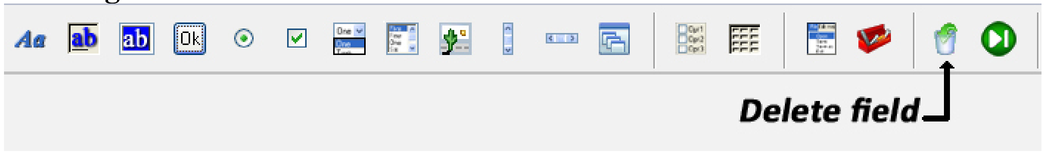
Move the field to the desired location by pressing the appropriate (ARROW) key.

It is possible to accidentally move a field past the left or top edge of the panel. To gain access to this "negative" area you have to use the ctrl-up and ctrl-left arrow keys.

Using the Properties Box

If you want to use the Properties Box to change the location of fields in the Format Window, first select the desired field and then modify the location by changing the values contained in the row and column properties in the Properties Box.

Deleting a Field in the Format Window



Using the Mouse

Move the mouse pointer to the "Delete" icon on the Icon Toolbar and click the left button once. The pointer symbol will change to an action pointer. Move the pointer so that it is directly on the Push Button field and click the left mouse button once more. The Push Button field will be deleted.

Using the Keyboard

Move the cursor to the field that you want to delete in the Format Window. Select the Field/Delete Menu Bar Option. The field will be automatically deleted. You should use caution when selecting a field to be deleted. When this method is used, there is no warning given before the field is deleted.

Manipulating Multiple Fields and Groups in the Format Window

You may find it useful to move a whole area within the format window rather than individually moving the fields and/or groups within that area. You can also copy fields and/or groups within the window to a save area and later copy those items back to a different place within the same panel or even to another panel. Finally, you can set the properties of all the fields in a selected area.

Selecting an area using the mouse

An area of the panel is selected by holding down the left mouse button and moving the mouse pointer down or to the right. After the mouse pointer has been moved a few pixels, the selection highlight will appear and will resize itself as the mouse pointer is moved further. When the required area has been highlighted, release the mouse button.

Selecting an area using the keyboard

Alternatively, you may position the cursor using the arrow keys and then use the CTRL-RIGHT and CTRL-DOWN keys to select an area (the R and D keys if CTRL-RIGHT and CTRL-DOWN are not supported on your keyboard).

Moving the selected area using the mouse

To move the items included in the selected area, move the mouse pointer into the selected area, hold down the left mouse button, move the selection highlight to a new location and then release the mouse button.

Moving the selected area using the keyboard

Alternatively, you may use the arrow keys to move the selection highlight and then press ENTER.

Copying the selected area

Once an area has been selected, the items within it may be copied to the editor clipboard using the Edit/Copy menu option.

Deleting the selected area

Once an area has been selected, the items within it may be deleted using the Edit/Clear menu option.

Deleting and copying the selected area

Once an area has been selected, the items within it may be copied to the editor clipboard and then deleted in one action using the Edit/Cut menu option.

Copying an area back to the Format window

Once items have been copied to the editor clipboard using the Edit/Copy or the Edit/Cut menu options, these items may be copied back to the current panel or a different panel using the Edit/Paste menu option. To do this position the cursor using the mouse or arrow keys and select the Edit/Paste option. The items will be copied to the area whose top left corner is defined by the cursor position. The Paste option will maintain all the IDs of selected items unless those IDs are already being used in the panel. This may occur if items are being moved to another panel or copied within the same panel. The Paste option will also adjust the size of items if items are being moved to a panel with a different cell size. The contents of the editor clipboard will be maintained (for use in Paste operations) until another Copy or Cut operation is performed or a selected area is moved (see Moving the Selected Area above).

Setting properties of fields within the selected area

Once an area has been selected, properties of all the fields within the area may be set in one action. Access the Properties Box as if just one field has been selected and the values in the Properties Box will go blank. You may now type in new values which will apply to all the fields in the selected area.

Overlaying One Field with Another in the Format Window

When creating or moving fields in the Format Window, it is possible to overlay one field with another. When a field is overlaid with another in the Format Window, it may cause the overlaid field to disappear from the Format Window. To restore the field back to its original state, move the top field. If the overlaid field does not appear in the Format Window after you have moved the top field, select the View/Test Menu Bar Option to display the panel in test mode. Press the (ESC) key to return to the Format Window. The overlaid field will then appear in the Format Window.

Undoing Your Last Change

If during editing, you make a change to the panel in error or a change that you are not satisfied with, you can undo the change by selecting the Edit/Undo menu option. Any edit action is considered a change, including adding and deleting fields, moving fields and changing attributes by selecting menu options. Only the last edit action can be undone.

Saving your Work

Use the File/Save menu option to save your work at any time. When you exit the editor, you will automatically be asked if you wish to save any changes you have made.

Once you are finished editing a panel you will most likely want to generate or regenerate the COBOL code that is used to interact with the panel at runtime. Use the File/Generate menu option for this purpose. If you are using COBOL copy files rather than panel files to hold your panel definitions at runtime, this generation step is particularly important because the generated copy file holds the definition of the panel as well as the code used to interact with it. Chapters C2 and C3 discuss the generated code.

CHAPTER B3 - Working With Menu Bars

Start COBOL sp2 and Open a Panel File

If you are not continuing from Chapter B2, invoke the COBOL sp2 Panel Editor from your operating system/environment as you would normally. If you are continuing from the last section, please proceed to the next section entitled, What is a COBOL sp2 Menu Bar?

When the COBOL sp2 Panel Editor is loaded, open the panel file created in Chapter B2 by selecting the Open file Pull Down Menu option from the File Menu Bar Option. If you have not created the panel file as outlined in Chapter B2 or you have not reviewed Chapter B2, we suggest that you do so at this time.

If you are not using panel files it is not necessary to open one. If you wish to open a copy file that you have generated you may do that also using the File/Open menu option. Reply "yes" when you are asked if you wish to import the copy file.

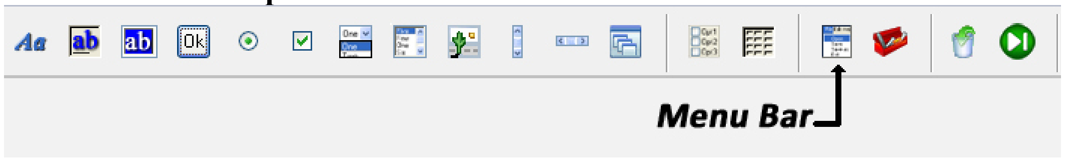
What is a COBOL sp2 Menu Bar?

COBOL sp2 allows you to design practically any type of menu system you desire. If you plan to run your application from within a World Wide Web browser at some point in the future, we suggest that you skip this step and design your menuing system using icon based menu options from within the panel definition.

If you plan to run your application or plan to run the application screens strictly in a native Windows environment, then it may be desirable for you to utilize a Menu Bar in your application.

Menu Bars are simply main menu options arranged in a horizontal fashion at the top of the panel. Submenu options are displayed in pulldown menus displayed directly under the corresponding menu bar option.

Create a COBOL sp2 Menu Bar



Set the Menu property to “New menu.” See Chapter B1 for details on setting properties.

Mnemonic Characters in the COBOL sp2 Menu Bar Options

You should notice that the "file" menu bar option is preceded by a tilde character "~." This character is used to identify the mnemonic character within the menu option text. The "File" menu option will therefore display with an underline beneath the "F" in File. In a character mode environment, the "F" character will be highlighted instead of underlined.

Please also note that in addition to the tilde, the "Open" pull down option is preceded by an equal sign "=." This character is used to signify that the menu option is a submenu option to the menu option immediately above it.

Assigning Menu Id's in a COBOL sp2 Menu Bar

The Id property in the Menu definition pop-up window contains an "001" for the File menu option and an "021" for the "Open" pulldown menu option. These three digit numbers refer to the option number for the menu bar.

If the "Open" pulldown menu option is selected by the operator of your program, COBOL sp2 will return an "021" menu id number to your program. This will allow you to detect which Menu Bar or Pulldown Menu option was selected by the operator of the program.

The menu id number which is assigned to Menu Bar and Pulldown Menu options may be any number you desire. The menu id numbers which are defined below assume a maximum number of Menu Bar options of 20. Therefore the Menu Bar options range from 001 to 020.

The first option in the first pulldown, therefore should be 021. Assuming a maximum of 20 Pulldown Menu Options, the range for the first Pulldown Menu would be 021 through 040.

The first option in the second pulldown, therefore should be 041. Assuming a maximum of 20 Pulldown Menu Options, the range for the second Pulldown Menu would be 041 through 060. This method of Menu id number assignment allows you to have an organized approach to Menu option number assignment and allows for future expansion of up to 20 options for the Menu Bar and each Pulldown Menu.

The following table provides an example of this method of Menu id assignment for Menu Bar processing:

Menu Bar Options

001	002	003	004	005	006	007	...	020

Pulldown Menu Options

021	041	061	081	101	121	141	...	401
022	042	062	082	102	122	142	...	402
023	043	063	083	103	123	143	...	403
. . .	. . .	. . .	. . .	. . .	. . .	. . .	. . .	. . .
040	060	080	100	120	140	160	...	420

Using the following table, enter the information into the Menu definition pop-up window to define the Menu Bar.

Text	Id	Key
~File	001	
=~Open	021	
=~Close	022	
=-	000	
=E~xit	023	
~Options	002	
=~Orders	041	
=~Shipping	042	
=~Balance	043	
==~Current Balance	501	
==~Account History	502	

You should note that in the above table, the Cascading Pull Down Menu options, "Current Balance" and "Account History" fall outside of the logical numbering arrangement, therefore they should be assigned special numbers. They have been assigned Menu ID Numbers of 501 and 502 respectively. Also please note the "-" (dash) between "Close" and "Exit" which defines a separator between the two options.

After entering the above information in the pop-up window, press the (ENTER) key to return to the Format Window.

Add A Title And 20 Character Description For The Menu Bar

Set the Title property to "Produce Distributors, Incorporated." See Chapter B1 for details on setting properties.

This is how you add a title for the window you are creating. This procedure is identical for all panels, whether they include a menu bar or not.

Set the Description property to "Sample System Menu." See Chapter B1 for details on setting properties.

This is how you add a description of up to 20 characters for your panel. This procedure is identical for all panels, whether they include a menu bar or not.

Test The Function Of The Menu Bar

Select the Test pull down option from the View menu bar option. The Menu Bar you just created will be displayed so that you may test it for appearance and function.

Save the Menu Bar

Save the menu bar panel by selecting the Save pull down option from the File menu bar option.

COBOL sp2 panel names are case sensitive in both the Panel Editor as well as your COBOL program. If you attempt to load a panel in the Panel Editor or display a panel in your program, and one or more letters are typed in the incorrect case, the panel will not be loaded or displayed.

Deleting an sp2 Menu Bar

Select the Menu property, highlight the menu which you want to delete and click the right mouse button to delete the menu.

CHAPTER B4 - Create the Client Area of the Main Menu Panel

What is the Client Area?

The Client Area of the panel is the area directly under the menu bar. The Client Area generally contains information that is universal throughout the application.

For example, in a typical order entry system, the customer's account number, name, address and telephone number would be appropriate data for display in the Client Area. This information is not only the typical starting point for processing a customer's order, it may also be desirable to display this information during the entire order entry process.

Opening a COBOL sp2 Panel

If you have just completed the previous section, you may continue working on the MAINMENU panel and ignore step 2 below. If not, you must first open the panel that you created previously.

Select the Open panel pull down option from the File menu bar option and select the panel named "MAINMENU". The panel may be selected and loaded by double clicking the left mouse button directly on the panel name. It may also be selected by highlighting the panel name and pressing the (ENTER) key.

Opening a COBOL sp2 Panel Using the Treeview Control

You can also open the panel by selecting the panel from the Treeview Control as follows:

First open the panel file named SAMPLE.PAN by selecting the Open file pulldown menu option from the File menu bar option.

The Treeview Control will display an expandable list of all panels and all panel fields.

Select the MAINMENU panel by positioning the mouse pointer on the panel name in the Treeview Control, click the left mouse button and continue to hold it down while dragging the mouse pointer to the Format Window of the Panel Editor.

The MAINMENU panel will automatically be loaded into the Format Window.

Default Look-and-Feel for the Panel

The default look-and-feel for the client area (in graphical mode) gives your panel a 3-dimensional appearance. This includes the following:

If a panel has the default color, the color used for pushbuttons will be used for the panel.

If a panel has the default font, a smaller, thinner font may be used.

If a field has the default border, the border will be drawn as a 3d border.

If a group has the default border, the border will be drawn as a 3d border.

If a repeat group has the default border, the border will be drawn as a 3d border and will be resized to wrap around any scroll bars present.

If you do not want this 3d look-and-feel but would prefer a more traditional, flatter look, position the cursor in the Format Window so that the Panel has focus and select the More options property in the Properties Box. The default setting is X”01” use 3d effects. If you click the left mouse button while maintaining the highlight on the default setting, the 3d effects option will be turned off. Options in this specific property can be toggled on or off.

CHAPTER B5 - Panel Size and Position

Change the Size of the Panel

Using the Mouse

Select the Size/Position pull down option from the View menu bar option. The Main Menu panel will then display as though you are testing the panel. You may now change the default size and position of the panel.

Using the Keyboard

If you prefer to use the keyboard instead of the mouse, press the (ALT) or (F10) key to activate the Menu Bar. Press the "V" mnemonic to display the View Pull Down Menu and then the "S" mnemonic to choose the Size/Position option. You may also use the (ARROW) keys to navigate the COBOL sp2 Menu Bar and the (ENTER) key to choose the Size/Position option from the View Pull Down Menu. The panel will appear as though it is in Test mode.

Using the Properties Box

Set the Width and Height properties to "74" and "20" respectively. See Chapter B1 for details on setting properties for objects in COBOL sp2.

Using the Mouse

Move the pointer to the bottom right corner of the Main Menu panel. When the pointer is on the border of the panel, you will see that the pointer symbol changes from the default pointer symbol to a diagonally pointing double arrow symbol.

Using the Keyboard

Press the (ALT) or (F10) key to activate the System Icon, then press the DOWN ARROW key to display the various options available on the System Icon. Use the UP ARROW or DOWN ARROW key to move the highlight to the Size option on the System Icon Menu. Press the ENTER key to select the Size option. You will see that all four sides of the panel border are now highlighted.

Using the Mouse

With the pointer directly on the border, click and hold the left button. Continue to hold the left button while dragging the pointer to the left and up. This will reduce the size of the Main Menu Panel. Keep the panel at least 90% to 95% of the original panel size so that you can add a few more fields to the client area of the window. When you have reached the desired size, release the left button.

Using the Keyboard

Set the focus on the bottom panel border by pressing the DOWN ARROW key once. After the focus has been established, press the UP ARROW key once or twice to reduce the height of the panel. Now press the RIGHT ARROW key to change the focus to the right panel border and press the LEFT ARROW key once or twice to reduce the width of the panel. When you have reached the desired panel size, press the ENTER key twice to return to the Panel Editor.

Change the Position of the Panel

Using the Mouse

To change the position, move the pointer to the title bar on the panel. Click and hold the left button. Drag the pointer to the right and down. This will change the position of the Main Menu Panel. When the panel is in the desired position, release the left button.

Press the (ENTER) key to return to the Format Window.

Using the Keyboard

If you prefer to use the keyboard instead of the mouse, press the (ALT) or (F10) key to activate the Menu Bar. Press the "V" mnemonic to display the View Pull Down Menu and then the "S" mnemonic to choose the Size/Position option. You may also use the (ARROW) keys to navigate the COBOL sp2 Menu Bar and the (ENTER) key to choose the Size/Position option from the View Pull Down Menu. The panel will appear as though it is in Test mode.

Press the (ALT) or (F10) key to activate the System Icon, then press the DOWN ARROW key to display the various options available on the System Icon. Use the UP ARROW or DOWN ARROW key to move the highlight to the Move option on the System Icon Menu. Press the ENTER key to select the Move option. You will see that all four sides of the panel border are now highlighted.

Press the RIGHT ARROW key twice, then press the DOWN ARROW key twice to move the panel to its new default location. Press the ENTER key twice to return to the Format Window.

Select the Test pull down option from the View menu bar option. The Menu Bar will be displayed so that you may test it for appearance and function.

Save the Main Menu by selecting the Save pull down option from the File menu bar option.

Using the Properties Box

Set the Row and Column properties to "11" and "13" respectively. See Chapter B1 for details on editing properties for objects in COBOL sp2.

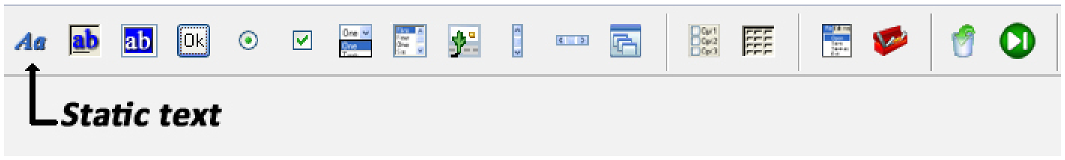
CHAPTER B6 - Add a Line of Static Text to the Panel

What is Static Text?

Text

In COBOL sp2, static text is generally used to represent a field label. Field labels are also sometimes referred to as field prompts. This is how it will be used in this example. Static text could also be used anytime you want to add information to the panel which normally wouldn't change.

Create a Line of Static Text in the Client Area



Using the Mouse

Click the mouse pointer on the "Static Text" Icon. The pointer will change from a diagonal arrow to a vertical arrow, indicating that the icon has been selected. Move the pointer to the center of the Format Window and click once to establish the static text. The default value is "Text."

Using the Keyboard

You may also select the Field/Static text Menu Bar Option to create Static Text in the Format Window. You should use caution when creating panel fields using the COBOL sp2 Menu Bar. The field will be automatically positioned where the cursor is located in the Format Window. It is important to first locate the cursor to the desired field location in the Format Window before activating the Menu Bar to create the field.

Move the pointer to the Text property in the Properties Box and click the left button. This is the property in which you will establish the actual text for the static text. Type "Account Number:" and delete any extra text characters from the original value of “Text” at the end of the property using the (DEL). Press the (ENTER) key. You will see that the text in the Format Window has been changed to the new value.

Using the Properties Box

Set the Width and Height properties to "74" and "20" respectively. See Chapter B1 for details on setting properties for objects in COBOL sp2.

Move the Line of Static Text in the Client Area

If you are not using a mouse to design your panels, please refer to the instructions, in Chapter B2, How to Use the COBOL sp2 Panel Editor for information on moving and re-sizing fields with the keyboard.

The only problem with our line of static text is that it should be repositioned to the top left corner of the Main Menu Window. Reposition the text by double clicking on the static text in the Format Window. A reverse video border will appear around the text. Move the pointer to the center of the static text, click once and hold the left mouse button down. Drag the mouse pointer to the left and up until the static text is in the upper left corner of the Format Window. Release the left mouse button to establish the new location.

Save the Main Menu by selecting the Save pull down option from the File menu bar option.

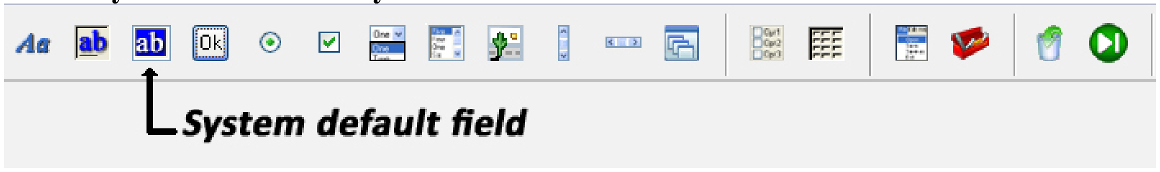
CHAPTER B7 - Working with Data Entry Fields

What is a Data Entry Field?

A Data Entry Field is simply a field on the panel into which the operator of your application can type information. COBOL sp2 provides two different types of Data Entry Fields, Custom Data Entry Fields and System Default Data Entry Fields.

Custom Data Entry Fields are special COBOL sp2 fields which behave more like a traditional input field. For more information on Custom Data Entry Fields, please refer to Chapter B14, Working with Custom Fields. System Default Fields behave like traditional GUI default data entry fields. These fields support a vertical cursor, I-Beam mouse pointer symbol and are always in insert mode. Unlike Custom Fields, System Default Fields allow more data to be typed into the field than allowed by the COBOL Field Format. This can cause confusion for the operator of your application in a typical COBOL application. You should note that when running in a text mode environment, there is no difference in the behavior of Custom Data Entry Fields and System Default Data Entry Fields. (See SP2EDT in Appendix B for information on changing the behavior of System Fields.)

Add a System Default Entry Field to the Panel



Now that you have created a field caption with the static text, you should add an entry field to correspond to the text. Click the mouse pointer on the "System Default Entry Field" icon, move the pointer to the right side of the field caption you just created in the Format Window and click once.

Using the Keyboard

You may also select the Field/Entry field Menu Bar Option to create a System Default Field in the Format Window. You should use caution when creating panel fields using the COBOL sp2 Menu Bar. The field will be automatically positioned where the cursor is located in the Format Window. It is important to first locate the cursor to the desired field location in the Format Window before activating the Menu Bar to create the field.

Assign a COBOL Field Name To The Entry Field

Set the Name property to "ACCOUNT-NUMBER.” See Chapter B1 for details on setting properties.

Assign an Initial Value to the Entry Field

Set the Value property to "000000." See Chapter B1 for details on setting properties.

Assign a Standard COBOL Field Format to the Entry Field

Set the Format property to “9(6).” See Chapter B1 for details on setting properties.

As you can see, the System Default Entry Field is not large enough to view all 6 digits of the account number at one time. Although this does not create any problems for data entry, you should increase the size of the field to display all 6 digits at one time.

Increase the Size of the System Default Entry Field

If you are not using a mouse to design your panels, please refer to the instructions, in Chapter B2, How to Use the COBOL sp2 Panel Editor for information on moving and re-sizing fields with the keyboard.

You should increase the field at least three cell columns to the right in this case. To increase the field size, double click the mouse directly on the System Default Entry Field you just created. A reverse video border will surround the System Default Field.

Move the pointer to the right side of the reverse video border until the pointer symbol changes to a horizontal double arrow.

Click and hold the left mouse button. Drag the mouse to the right at least 3 cell columns to increase the width of the field. Release the left mouse button when the field is the desired size.

If you would like more information on increasing the size of a field, please refer to Chapter B2, How to use the COBOL sp2 Panel Editor, for instructions on increasing field size.

Increasing the Size of a Field Using the Properties Box

Set the Width property to "85." See Chapter B1 for details on setting properties.

If you want all fields in a column to be identical in size, by using this approach to changing the field width, all fields can be made precisely the same width. In addition, when you use the (ESC) key to toggle between the Format Window and the Properties Box, COBOL sp2 maintains the focus on the same Field Property as you move from one field to the next.

Add Additional Captions and Entry Fields to the Client Area

Follow the previous example to add several more Captions and System Default Entry Fields to the Client Area. The COBOL Field Name is optional, but you should increase the size of the fields to an appropriate length. The additional fields should have the following properties:

Static Text	**COBOL Field Name**	**Format Property**
Customer Name:	CUSTOMER-NAME	X(25)
Address:	CUSTOMER-ADDRESS	X(25)
City:	CUSTOMER-CITY	X(15)
State	CUSTOMER-STATE	X(2)
Zip Code:	CUSTOMER-ZIP-CODE	9(5)

Before testing the panel, we must establish the Zip Code Field as a numeric field. Make sure that the cursor is on the Zip Code Field in the Format Window and click the mouse once on the Type property in the Properties Box. Select the Numeric (second) option from the drop down list and click the mouse once more in the Format Window.

Select the Test pull down option from the View menu bar option. The Menu Bar will be displayed so that you may test it for appearance and function. Press the (ESC) key to return to the Format Window

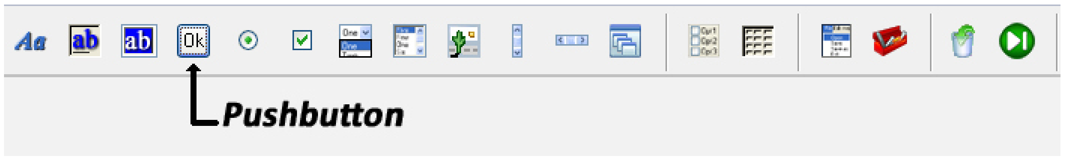
Save the Main Menu by selecting the Save pull down option from the File menu bar option.

CHAPTER B8 - Working with Push Button Fields

What is a Push Button?

Push Buttons are generally used to invoke a process in your application. These fields are accessible with the mouse, the keyboard or both. In COBOL sp2 you can assign a key value to the Push Button. When a Push Button is clicked with the mouse, COBOL sp2 sends the key value back to your program. This allows you to test for a key value which greatly simplifies the programming effort.

Add a Push Button Field to the Client Area



Scroll the Panel Down to Work at the Bottom of the Panel

If you are not using a mouse to design your panels, please refer to the instructions, in Chapter B2, How to Use the COBOL sp2 Panel Editor for information on scrolling through the panel with the keyboard.

Because the Push Button fields will be established at the bottom of the panel, you should scroll the panel image down to view the bottom of the panel. To scroll down, click the mouse on the downward pointing arrow on the vertical scroll bar. The panel will shift to reveal the lower portion.

It is also possible to scroll the panel without using the scroll bars. Click and hold the left mouse button, while the pointer is in the Format Window. Drag the mouse downward while continuing to hold the left button down. The panel will scroll when the mouse is dragged outside of the COBOL sp2 Panel Editor Window to one of the edges of the screen image on your monitor.

To create a Push Button field, click the mouse on the "Push Button" icon, move the pointer 4 cell rows below the Zip Code field in the bottom left corner of the Format Window and click once.

Using the Keyboard

You may also select the Field/Pushbutton Menu Bar Option to create a Push Button in the Format Window. You should use caution when creating panel fields using the COBOL sp2 Menu Bar. The field will be automatically positioned where the cursor is located in the Format Window. It is important to first locate the cursor to the desired field location in the Format Window before activating the Menu Bar to create the field.

Assign a Caption and Mnemonic to the Push Button

Set the Value property to "~Add Customer - F4." See Chapter B1 for details on setting properties. The tilde "~" character before the "A" represents the mnemonic character for the Push Button.

If the panel is too small for the Push Buttons to fit below the System Default Entry Fields, you can increase the default panel size with the Size/Position pulldown menu option on the View menu bar option.

Assign a Key Value to the Push Button

You may have noticed that when the cursor is located on the Push Button in the Format Window, the Help key property in the Properties Box contains a numeric value. This represents the decimal value of the key which will be passed back to your COBOL application when the Push Button is selected. Using the mouse, place the cursor on the Push Button which you just created.

Set the Help key property to "318" which represents the decimal value of the (F4) key. You can also simply press the (F4) key directly and then press the (UP ARROW) or (DOWN ARROW) key to establish the value of “318” in the Help key property. In this example, the (F4) key will be used to indicate that the operator of the application wants to add a new customer.

For a list of the decimal values which represent keyboard keys, please refer to Appendix A, Decimal Key Values.

Add an Additional Push Button to the Client Area

Using the same procedure outlined above, create one more Push Button. Locate the second Push Button immediately below the Push Button you just created. The additional Push Button should be given the following properties:

Value	Help key
~Delete Customer - F5	"319" (F5) key

Increase the Size of the Push Button Field

As you can see, the "Add Customer" Push Button Field is a bit shorter than the "Delete Customer" Push Button Field. You should increase the "Add Customer" Push Button Field at least three cell columns to the right in this case. To increase the field size, double click the mouse directly on the Add Customer Push Button Field you just created. A reverse video border will surround the Push Button.

Move the pointer to the right side of the reverse video border until the pointer symbol changes to a horizontal double arrow. Click and hold the left mouse button. Drag the mouse to the right at least 3 cell columns to increase the width of the field. Release the left mouse button when the field is the desired size.

If you would like more information on increasing the size of a field, please refer to Chapter B2, How to use the COBOL sp2 Panel Editor, for instructions on increasing field size.

Select the Test pull down option from the View menu bar option. The Menu Bar will be displayed so that you may test it for appearance and function. Press the (ESC) key to return to the Format Window

Save the Main Menu by selecting the Save pull down option from the File menu bar option.

Increasing the Size of a Push Button Field Using the Properties Box

An alternate method for increasing the pushbutton field size is to type the desired length of the field in pixels directly into the Width property in the Properties Box. First note the value in the Width property for the "Delete Customer" Push Button Field. Change the focus so that the cursor is on the "Add Customer Push Button Field" and type in the identical Width value. When you press the (ENTER) key, you will see the length of both Push Button Field will be identical.
If you want all fields in a column to be identical in size, by using this approach to changing the field width, all fields can be made precisely the same width. In addition, when you use the (ESC) key to toggle between the Format Window and the Properties Box, COBOL sp2 maintains the focus on the same Field Property as you move from one field to the next.

Default Push Button

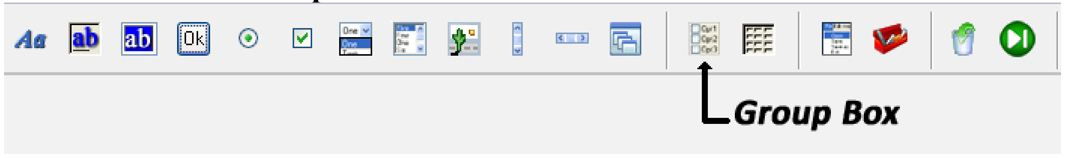
Set the Usage option property to the Default option.

CHAPTER B9 - Working with Field Groups

What is a Field Group?

A field group is used to combine similar fields into a set. Once the fields are combined into a group, by default, the cursor tabbing behavior changes. The cursor will now tab from the first field in one group to the next field outside the field group.

Establish a Field Group



Now that you have two Push Buttons on the panel, you should establish a field group. To accomplish this, position the pointer on the group icon and click the left mouse button once. Move the pointer to the top left corner of the first Push Button in the Format Window. Click once to establish the group box.

Using the Keyboard

You may also select the Field/Group Menu Bar Option to create a Field Group in the Format Window. You should use caution when creating Groups using the COBOL sp2 Menu Bar. The Group will be automatically positioned where the cursor is located in the Format Window. It is important to first locate the cursor to the left and immediately above the far left field to be included in the Group before activating the Menu Bar to create the field.

Include All Fields Within the Field Group

A box with a reverse video border will appear. If the box is a thin, line, the Group Box was "de-selected." To change the size of the Group Box, select it by double clicking the mouse on the thin line until it becomes a thick reverse video line. You may also select the Group with the keyboard by first positioning the cursor on the Group and pressing the (ENTER) key.

This box will not be large enough to include all Push Buttons in the group. To enclose all Push Buttons in the group move the pointer to the lower right corner of the reverse video border until the pointer changes to a diagonally pointing double arrow symbol. When the pointer becomes a double arrow, you can resize the reverse video border down and to the right to include all Push Buttons in the field group.

Using the Keyboard

If you want to use the keyboard to change the size of the Group in the Format Window, first locate the cursor directly on the Group you created with the (ARROW) keys.

If the Group isn't selected, press the (ENTER) key to select the field. The field will be surrounded by a reverse video border, indicating that it has been selected.

Change the size of the Group by pressing the (CTRL-RIGHT ARROW) key combination to increase the width and (CTRL-DOWN ARROW) key combination to increase the height.

If you used the mouse to re-size the group, release the left mouse button to establish the Field Group. The reverse video border will become a thin reverse video box around the fields in the group. If the box doesn't include all the fields in the group, you may delete the field group and start over or change the size to include all the fields in the group. For more information on deleting a field, please refer to Chapter B2, How to Use the COBOL sp2 Panel Editor.

Establish a "No Action" Selection Criteria for the Push Button Group

Push Buttons do not require any special selection criteria when combined in a group. However, by default, a group is created without any members even though fields may exist within the boundaries of the group. In this case, it would be useful to assign the Push Buttons as actual members of the group, so the group type should be changed from "No member" to "No action." In order to establish the "No Action" selection criteria, first make sure that the cursor is on the group box which you just established. If the cursor is on the group box, the first property in the Properties Box should read "Group."

Set the Select type property to "No = No action." See Chapter B1 for details on setting properties.

Establishing Tabbing for Push Buttons Groups

It is generally acceptable to allow the cursor to tab through all Push Buttons within a Push Button Group. To allow the cursor to access each Push Button, the "Tab Within" option will be set for the Push Button Group.

Set the Tab within property to “Yes.” See Chapter B1 for details on setting properties.

Select the Test pull down option from the View menu bar option. The mainmenu panel will be displayed so that you may test it for appearance and function. Press the (ESC) key to return to the Format Window

Save the mainmenu panel by selecting the Save pull down option from the File menu bar option.

CHAPTER B10 - Working with Icon Fields

Scroll the Panel Down to Work at the Bottom of the Panel

Because the Icon fields will be established in the top right area of the panel, you should scroll the panel image up and to the right to view the top right corner of the panel. To scroll up, click the mouse on the upward pointing arrow on the vertical scroll bar. To scroll right, click the mouse on the right pointing arrow on the horizontal scroll bar. The panel will shift to reveal the top right portion. If you are not using a mouse to design your panels, please refer to the instructions, in Chapter B2, How to Use the COBOL sp2 Panel Editor for information on scrolling through the panel with the keyboard.

What is an Icon?

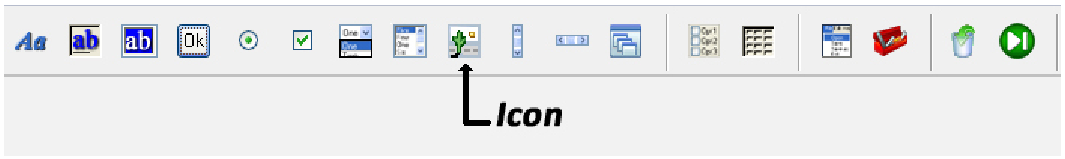
Icons are useful for providing the operator of your application with a visually oriented method of invoking a process in your application. Because icons are bit mapped images, the graphic image in your icons are only limited by your imagination.

Icons may also be useful to dynamically display pictures on the panel. This can be useful in an inventory application to display images of the items in inventory or in a personnel application to display pictures of employees.

Icons used in COBOL sp2 panels are standard 16 color or 256 color ".BMP" files. COBOL sp2 also supports highly compressed “.JPG” images as well. For “.JPG” image support, please contact your local distribution agent or contact Flexus directly. The “.JPG” image support is a low cost add-on product to COBOL sp2.

“.BMP” and “.JPG” bit mapped images are not the same type of icons which are generally used to invoke programs in your GUI environment. Bit mapped images are used for icons within a COBOL sp2 application so that the operator of your application will not be able to relocate the icon within the panel image.

Add an Icon Field to the Panel



To create an icon field, click the left mouse button on the "Icon" icon or select the Field/Icon Menu Bar Option.

If you are using a mouse, move the pointer near the top and center of the Format Window and click once. The first icon to add will simply be a picture that will be added to the Window in order to improve the appearance of the panel.

Select the Value property in the Properties Box. A small pop-up box containing three entry fields will be displayed. The second field in the pop-up box has a caption of “Icon file.” This is the property in which you will type the name of the image file which you wish to use.

If you aren’t sure of the name of the bit mapped image which you plan to use for your icon field, click the small downward pointing arrow at the far right side of the “Icon file” entry field in the pop-up box. This will display the Windows File Open dialog box which will allow you to locate and select the file name from a list of files.

You may now type the file name, "PRODUCE.BMP" directly into the File Name property or scroll through the list box on the Open pop-up window until you highlight the file name. If you choose to select the file name with the highlight, press the (ENTER) key when the proper file name has been highlighted.
You will now see that the Value property in the Properties Box contains the file name, "PRODUCE.BMP." If you already know the name of the bitmap file which you want to display on the panel, you may also type it directly into the Value property in the Properties Box.

You should also notice that the Type property in the Properties Box displays “x”00” which is a regular bitmap image. If you want to display 256 color bitmaps from your COBOL sp2 panel and you will be using video devices that support 256 colors or less, then you may wish to establish the icon field as a "bitmap exact" in the Type property in the Properties Box. This has the effect of displaying the exact 256 color palette for the bitmap. Please bear in mind that each 256 color bitmap may have a unique palette, therefore you may get unpredictable results if you attempt to display multiple 256 color bitmap icons on the panel simultaneously on a video device supporting a low number of colors.

The bitmap image that you have added to the panel is larger than the default icon size, so it is necessary to resize the icon to fit the size of the bitmap. This may be accomplished with the mouse or the keyboard using the same method as resizing any other field in the Format Window. The easiest way to accomplish the desired result is to allow COBOL sp2 to automatically resize the bitmap image.

Set the Special format property to "i = size field to image." See Chapter B1 for details on setting properties.

After you have changed the size of the icon field, move the bitmap image to the top right corner of the panel. Moving an icon field is accomplished in the same manner as moving any field type in COBOL sp2. For instructions on moving a field, please refer to Chapter B2, How to Use the COBOL sp2 Panel Editor.

Now add three other icon fields below the produce icon which you just created. The primary icon file names entered into the Icon file property of the Value pop-up in the Properties Box for the other icons on the panel include:

BIGOK1.BMP
BIGCAN1.BMP
BIGHELP1.BMP

Because the Auxiliary File can only load the bitmap icon file name into the Icon file property of the Value pop-up in the Properties Box, you should move the cursor to the Icon details property of the Value pop-up in the Properties Box and type the following secondary file names immediately below each primary icon file name. The secondary icon file names are:

BIGOK2.BMP
BIGCAN2.BMP
BIGHELP2.BMP

The files, "BIG*.BMP" are bitmaps which have been made to resemble Push Buttons. The primary bitmaps (those with a "1") represent the icon in a normal state. The secondary bitmap file names which include a "2" represent the reverse image of the icon. This allows you to include icons in your application which have the appearance of being depressed when the mouse is clicked on them. These icons give a more graphical appearance to the panel and may also be used to invoke a process. The following section describes how to use an icon as a Push Button field.

Establish an Icon as a Graphical Push Button

If you have not added the bitmapped icons, "BIG*.BMP," please follow the instructions above to add the additional icons.

To establish an icon as a graphical Push Button, you should assign a key value to each icon. When the cursor in the Format Window is on the icon, the Help key property in the Properties Box contains the default value "0." This is the field in which you assign the control key which is represented by the icon. The values assigned to the file name and control key value properties are as follows:

Icon File Name Property	Key Value Property
BIGOK1.BMP/BIGOK2.BMP	013
BIGCAN1.BMP/BIGCAN2.BMP	027
BIGHELP1.BMP/BIGHELP2.BMP	315

Alternative Method for Establishing Graphical Push Buttons

Please also see Chapter B33 for another way of establishing graphical buttons.

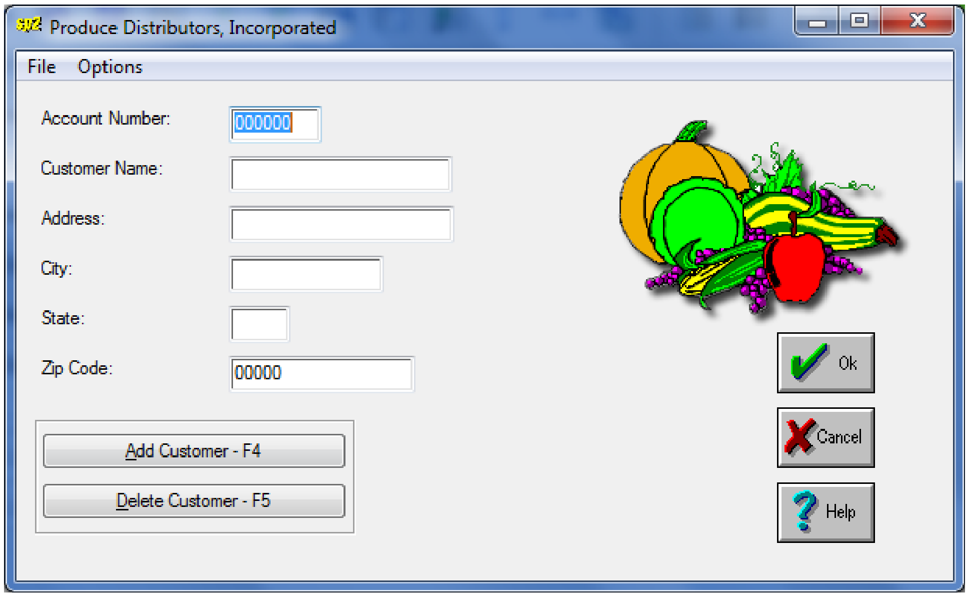
Save the Menu Bar Panel

Before testing the panel, you should save your work by selecting the Save pulldown option from the File menu bar option.

Test the Function of the Main Menu Panel

Select the Test pull down option from the View menu bar option. The Menu Bar will be displayed so that you may test it for appearance and function. Press the (ESC) key to return to the Format Window.

Your MAINMENU panel should look like this:



CHAPTER B11 - Working with Radio Buttons

What is a Radio Button?

A Radio Button is a selection field which contains a symbol and associated text. The symbol indicates whether or not the associated text option has been selected. Radio Buttons may be selected with the mouse or keyboard. Radio Buttons are generally used when you must provide the operator of your application with a limited set of choices from which only a single option may be selected.

The easiest way to process these fields is to make them emulate regular data entry fields. This is done by assigning Field Validation Values for them. For more information on Field Validation Processing, please see Chapter B19, Establishing Field Edit Validation Criteria. Because Radio Buttons are generally included in an auto-select group, only one field value should be assigned to each Radio Button. If you want the Radio Buttons to be included in any other group type than auto-select or single select, two values should be assigned. For example, you may wish to assign an on and an off value.

When the field is selected, the appropriate value will be moved into the program data area just as if the value had actually been entered. For an auto-select or single-select group, only one field will be generated in the data area for the whole group. If a new selection is made, a new value will overwrite the old value because only one selection can be valid at any one time.

Create a New COBOL sp2 Panel

Before creating any Radio Buttons, you should create a new panel for the sample program being developed.

If you exited COBOL sp2, you may begin working immediately with the default Format Window for your new panel. If you are continuing from the previous section, select the New panel pulldown option from the File menu bar option.

Establish Various Panel Properties

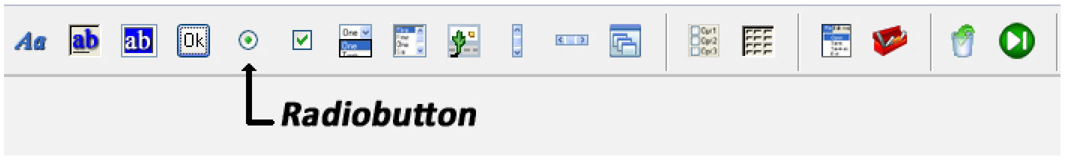
Set the following Properties in the Properties Box for the new panel:

Description:	Order Entry Panel
Title:	Orders

Add the following two static text items and two adjacent System Default Fields with the following Format Properties (PIC Clauses) and Name properties(COBOL Field Names) to the panel:

Static Text Items	**Format Property**	**Name Property**
Account Number:	9(6)	ACCOUNT-NUMBER
Customer Name:	X(25)	CUSTOMER-NAME

Add a Radio Button to the Panel



To create a Radio Button field, click the mouse on the "Radio Button" icon, move the pointer to the Format Window seven or eight cell rows under the Customer Name Field Caption and click once.
If you prefer to use the keyboard to create a Radio Button Field, first position the cursor seven or eight cell rows under the Customer Name Field Caption. Select the Field/Radiobutton Menu Bar Option to create the default Radio Button Field.

Assign a Caption and Mnemonic to the Radio Button

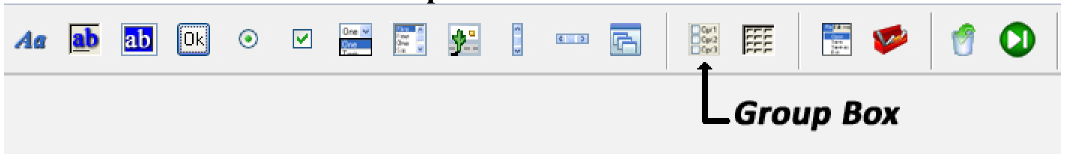
Set the Caption property to "~Check." See Chapter B1 for details on setting properties. The tilde "~" character before the "C" represents the mnemonic character for the Radio Button.

Add Additional Radio Buttons to the Panel

Using the same procedure outlined above, create several more Radio Buttons. The additional Radio Buttons should be given the following properties:

Caption
C~ash
C~redit Card
~Purchase Order

Establish a Radio Button Group



Using the Field Group icon, group the Radio Buttons you have just created. If you do not remember how to create a Field Group, please refer to Chapter B9, Working with Field Groups.
If you are using the keyboard to define the panel, first position the cursor to the left and just above the far left field to be included in the Group. Select the Field/Group Menu Bar Option to create the Field Group. Resize the Field Group as outlined in Chapter B2, Changing the Size of a Field or Group in the Format Window

Automatic Selection Criteria for the Radio Button Group

Radio Buttons should allow only one selection within a Radio Button group.

Set the Select type property to "a = auto-select" See Chapter B1 for details on setting properties.

The automatic selection setting will allow the current Radio Button to be automatically selected when the (ARROW) keys are used to move the cursor from one Radio Button to the next. When the cursor is tabbed out of the Radio Button Group, the selection will be based on the location of the cursor when the (TAB) key was pressed.

Assign a Discrete Value to your Radio Buttons

Discrete Values are part of COBOL sp2's Field Edit Validation Processing which COBOL sp2 allows you to establish for Radio Button and Data Entry Fields. Field Edit Validation Processing allows you to specify a range or table of acceptable values which may be used to allow Radio Buttons to emulate normal Data Entry Fields.

Set the Discrete values property to "C." See Chapter B1 for details on setting properties.

The value of "C" corresponds to your mnemonic for the Radio Button. Although the Discrete Value could actually be assigned any value, using the mnemonic character to represent the Discrete Value for the Radio Button seems to be the most logical approach.

You have now established a Discrete Value for the first Radio Button. This should help to make your programming effort much easier. When control is returned to your program, COBOL sp2 will return the value of "C" instead of a more cryptic code. If the "Check" Radio Button has been selected, the COBOL variable for the Radio Button Group will contain the value of "C." If the Radio Button has not been selected, the variable will contain a blank if there is no default Radio Button or the value of the selected Radio Button.

Repeat this process for the remaining Radio Buttons using the following properties:

Radio Button	**Discrete Value Property**
Cash	A
Credit Card	R
Purchase Order	P

Select the Test pull down option from the View menu bar option. The Order Panel will be displayed so that you may test it for appearance and function. Press the (ESC) key to return to the Format Window

Save the Order Panel by selecting the Save pull down option from the File menu bar option. A small pop-up window will appear allowing you to type the name "ORDER." Press (ENTER) key when you have typed in the name.

CHAPTER B12 - Working with Check Boxes

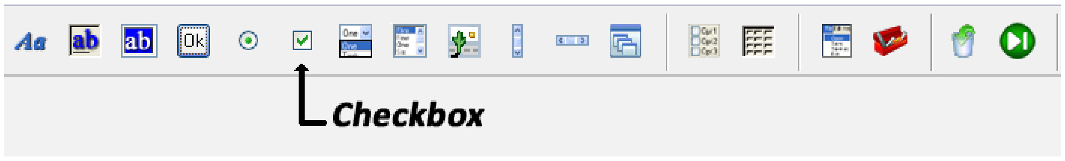
What is a Check Box?

A Check Box is a selection field which contains a symbol and associated text. The symbol indicates whether or not the associated text option has been selected. Check Boxes may be selected with the mouse or keyboard. Check Boxes are generally used when you must provide the operator of your application with a limited set of choices from which any number of options may be selected.

The easiest way to process these fields is to make them emulate regular data entry fields. This is done by assigning Field Validation Values for them. For more information on Field Validation Processing, please see Chapter B19, Establishing Field Edit Validation Criteria. Because Check Boxes are generally included in multiple selection groups, two values should be assigned. For example, you may wish to assign an on and an off value.

When the field is selected, the appropriate value will be moved into the program data area just as if the value had actually been entered.

Add a Check Box to the Panel



To create a Check Box field, click the mouse on the "Check Box" icon, move the pointer at least 3 or 4 cell rows under the Radio Button Group in the Format Window and click once.

To establish a Check Box field using the keyboard, first position the cursor at least 3 to 4 cell rows under the Radio Button Group and select the Field/Checkbox Menu Bar Option.

Assign a Caption and Mnemonic to the Check Box

Set the Caption property to "~Insure." See Chapter B1 for details on setting properties.

The tilde "~" character before the "I" represents the mnemonic character for the Check Box.

Assign Discrete Values to your Check Boxes

Discrete Values are part of COBOL sp2's Field Edit Validation Processing which COBOL sp2 allows you to establish for Check Box and Data Entry Fields. Field Edit Validation Processing allows you to specify a range or table of acceptable values which may be used to allow Check Boxes to emulate normal Data Entry Fields.

Select the Discrete values property to display the drop down list. Type "Y" in the first field in the drop down list and “N” in the second field in the drop down list.

You have now established a set of Discrete Values for the Check Box. This should help to make your programming effort much easier. When control is returned to your program, COBOL sp2 will return the value of "Y" or "N" instead of a more cryptic code. If the "Insure" Check Box has been selected, the COBOL variable for the "Insure" Check Box will contain the value of "Y" (the first value is always returned if the field has been selected). If the Check Box has not been selected, the variable will contain the value of "N."

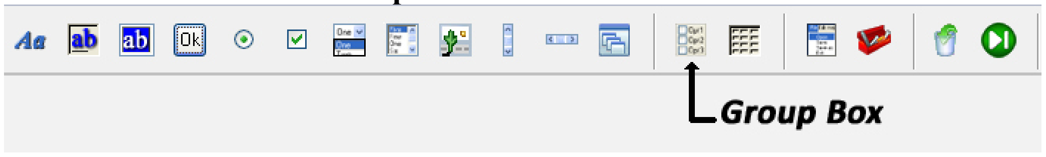
Add Additional Check Boxes to the Panel

Using the same procedure outlined above, create several more Check Boxes. The additional Check Boxes should be given the following properties:

Caption Properties	Discrete Value Properties

~Rush	Y and N
~Fragile Contents	Y and N
~Perishable	Y and N
R~efrigerate	Y and N

Establish a Check Box Group



Using the Field Group icon, group the Check Boxes you have just created. If you do not remember how to create a Field Group, please refer to Chapter B9, Working with Field Groups.

Establish Multiple Selection Criteria for the Check Box Group

Check Boxes should allow multiple choice selections within a Check Box group. In order to establish multiple selection criteria, first make sure that the cursor is on the group box which you just established. If the cursor is on the group box, the Object type property in the Properties Box should read "Group."

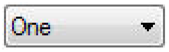
Set the Select type property to "m = multi-select." See Chapter B1 for details on setting properties.

Select the Test pull down option from the View menu bar option. The Order Panel will be displayed so that you may test it for appearance and function. Press the (ESC) key to return to the Format Window

Save the Order Panel by selecting the Save pull down option from the File menu bar option.

CHAPTER B13 - Working with Combination Boxes

What is a Combination Box?



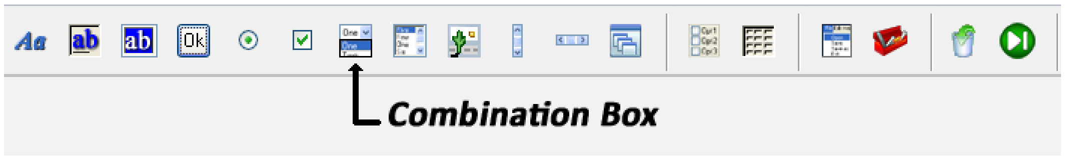
A Combination Box, also referred to as a "Drop Down List" or a "Choice List" is a field which allows the operator of your program to display a list a of valid options under the field. If the operator wishes to change the default option displayed, the mouse pointer may be used in conjunction with the downward pointing arrow to display a list under the field.

If all of the contents of a Combination Box can be displayed in the Drop Down List simultaneously, the Scroll Bar in the drop down list will automatically disappear. When there are more options in the Drop Down List than can be displayed at one time, the Scroll Bar will be present.

Combination boxes may be protected fields which only allow for selection of a valid option or they may be enterable by the operator.

In the following example, the Combination Box will be selection based. The Combination Box will be used as a method for selecting the initials of the sales representative taking the order.

Add a Combination Box to the Panel



Click the left mouse button with the mouse pointer on the "Combination box" icon. When the pointer changes to an upward pointing arrow, move the mouse pointer to the right side of the Account Number Entry Field in the Format Window and click once more. If you prefer to use the keyboard, first position the cursor in the desired location in the Format Window and select the Field/Combobox Menu Bar Option to create the Combination Box.

Set the first field in the drop down list of the Value property to “ARH.” Then set the subsequent fields in the drop down list of the Value property to the following three character strings, placing one per line in the drop down list:

ARH
CEG
DMH
HAB
RTW
TRB

The fact that "ARH" is repeated twice is not a mistake. Because the first default option appears as the default option as well as the first option in the drop down list, it must always be repeated.

Set the Format property to “X (3).” Please note that if you initialize the Combination Box with longer text values, you must increase the size of the COBOL Field Format.

If you test the panel now, you will see that the combination box is also the correct size to display all of the options. If longer text values were added, you may wish to also increase the horizontal size of the Combination Box in the Format Window.

Add a Caption to the Combination Box

To help make the panel easier to use for a novice operator, add the Static Text value, "Sales Rep:," directly above the Combination Box. Please refer to Chapter B6, Add a Line of Static Text to the Panel, for more detailed information on how to create Static Text on the panel.

Select the Test pull down option from the View menu bar option. The Order Panel will be displayed so that you may test it for appearance and function. Press the (ESC) key to return to the Format Window.

Save the Order Panel by selecting the Save pull down option from the File menu bar option.

CHAPTER B14 - Working with Custom Fields

What is a Custom Field?

An Entry Field is simply a field on the panel into which the operator of your application can type information.

The original implementation of COBOL sp2 for Microsoft Windows provided for two unique entry field types, Custom Data Entry Fields and System Default Data Entry Fields. The Custom Field was implemented in the preliminary release in order to provide a field type which behaved more like a traditional input field.

Some of the behavioral characteristics which were originally allowed in a Custom Field but not a System Default Field included:

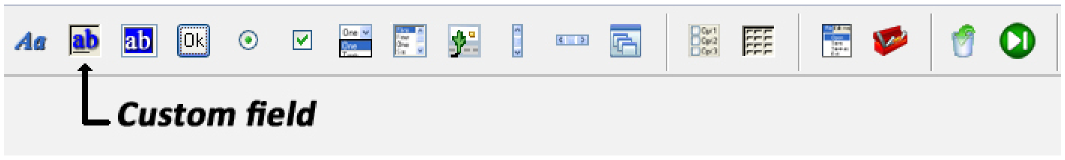
Unlike a System Default Entry Field, the Custom Field will not allow more data to be typed into the field than allowed by the COBOL field format. This can help avoid confusion for the operator of your application in a typical COBOL application. Enhancements to the COBOL sp2 System Default Field behavior during the past few years have enabled System Default Fields to limit the amount of data typed into the field based upon the COBOL Field Format. To learn more about this capability, please review the SP2EDT environment variable setting which may be found in Appendix B - Configuring COBOL sp2.

The Custom Field also allows the operator to toggle insert and overtype modes with the (INS) key. System Default Fields are usually in insert mode which is the generally the default setting in a GUI environment. Enhancements to the COBOL sp2 System Default Field behavior during the past few years has enabled System Default Fields to toggle insert and overtype modes with the (INS) key. To learn more about this capability, please review the SP2EDT environment variable setting which may be found in Appendix B - Configuring COBOL sp2.

In addition, Custom Fields may display a variety of cursor types, including the default underline cursor, no cursor display, a block cursor or a vertical I-beam style cursor. The cursor type for a Custom Field may be set using the Cursor show property in the Properties Box.

Custom Fields are also very useful for creating Repeat Groups. The next few chapters will accomplish this type of complex panel field.

Add Several Custom Fields to the Panel



Using the "Custom Data Entry Field" icon or the Field/Custom field Menu bar Option, add a Custom Field about 6 or 7 cell columns to the right of the Radio Button Group. Increase the size of the Custom Field by 10 cell columns. An easier method of increasing the field size is to change the Width property in the Properties Box from the default value of 50 to the larger size of 150.

Create a second Custom Data Entry Field to the right of the first, but do not increase the size of the field.

Locate a third and final Custom Data Entry Field to the right of the second. Increase the length of the field by 5 cell columns. An easier method of increasing the field size is to change the Width property in the Properties Box from the default value of 50 to the larger size of 100.

Create Static Text Headings for the Repeat Group

The Data Entry Fields would not be complete without Static Text Headings over the three repeated fields. Locate the following three Static Text Captions above the fields in the Group Box. Leave at least one blank cell row between the custom fields and the static text. This will provide additional room for the repeat group box which will be added in the next chapter.

Custom Field 1	Custom Field 2	Custom Field 3
Produce	Quantity	Price

CHAPTER B15 - Working with Repeat Groups

What is a Repeat Group?

A Repeat Group is a group of one or more fields with multiple vertical occurrences on the panel. Fields in a Repeat Group may be enterable, selectable (protected), display-only fields or any combination of field types.

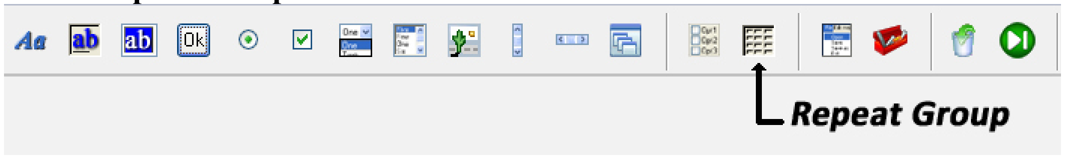
Repeat Groups can hold more fields than are visible on the panel. When a Repeat Group contains more fields than are visible, you should generally add a scroll bar to the Repeat Group. The Scroll Bar is used as a visual indicator to the operator that more fields are available than are displayed. In addition, the Repeat Group can be controlled with the mouse through the Scroll Bar. For more information on Scroll Bars, please refer to Chapter 17, Working with Scroll Bars.

A repeat group can scroll horizontally as well as vertically so long as it only has a single column. Set the Cursor movement property to "Horizontal scroll". The amount of horizontal scrolling is limited by the value contained in the Format property of the input field (PICTURE clause).

You may wish to consider using a List Box instead of a repeat group for some items with multiple occurrences. A List Box should generally be used for a single column list of entries that can be loaded all at once and do not have to be changed by the user. See Chapter B18 for details on how to define a List Box.

All field types except list boxes are allowed in repeat groups. The list of values associated with a repeated combo box must be the same for each occurrence but the selected value may be different. The filename linked to an icon field must be the same for each occurrence.

Add a Repeat Group to the Panel



If you are not using a mouse to design your panels, please refer to the instructions, in Chapter B2, How to Use the COBOL sp2 Panel Editor for information on establishing a Repeat Group with the keyboard.

Select the "Repeat Field" icon to repeat the Custom Fields on the Panel. Position the mouse pointer in the top left inside corner of the leftmost Custom Field which you just created in the last chapter. Click the left button once.

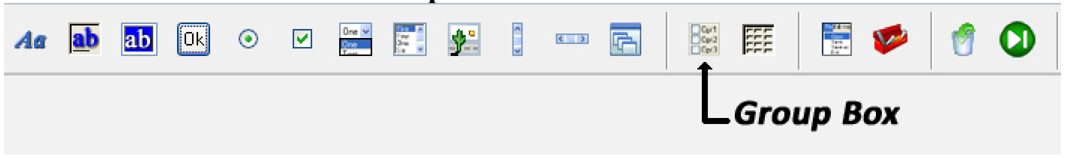
Change the size of the Repeat Field Box so that it extends approximately 4 to 5 cell columns to the right and 15 to 18 cell rows below the Custom Fields.

Specify the Total Number of Field Occurrences

With the cursor in the Format Window on the Repeat Field Box, move the mouse pointer to the Vertical occurs property in the Properties Box. This is where you will establish the number of times the Field Group will be repeated. If the Object type property in the Properties Box does not contain the text “Repeat,” the cursor is not on the Repeat Field Box.

Change the text from "1" to "25" and press the (ENTER) key. You should see the Custom Fields repeat down the panel within the Field Repeat Box.

Establish a Custom Field Group



Using the Field Group icon or the Group/Group Menu Bar Option, group the Custom Data Entry Fields you have just created. If you do not remember how to create a Field Group, please refer to Chapter B9, Working with Field Groups.

Define a Continuous Highlight Across Several Fields

Set the Select type property to “t = color-contiguous.”

Establishing Tabbing for Custom Data Entry Field Groups

Set the Tab within property to “Yes.”

If you test the panel now, you will see that the fields look a bit strange. That is because you are not finished defining the Repeat Group

The Repeat Group is nearly complete. The only things left to do are to remove the borders from the Custom Data Entry Fields and the Field Group as well as define a Scroll Bar for the Repeat Group.

CHAPTER B16 - Changing the Field Border

What is the Field Border?

A Field Border is the thin line which surrounds various panel fields in COBOL sp2. For example, both a Custom Data Entry Field and a System Default Data Entry Field contain a line which surrounds the data that is entered into these fields.

COBOL sp2 allows you to keep the default border, replace the default border with a different type of border or remove the border from the field.

Panel items which allow modification of the border include:

- Custom Data Entry Fields
- System Default Data Entry Fields
- List Boxes
- Field Groups
- Repeat Groups

Eliminate the Border for a Field

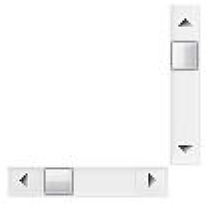
The Field Borders in the Repeat Group should be eliminated. For this type of field, the border is not necessary.

Set the Border type property to "No = no border." The border should disappear.

Repeat this process for the remaining fields in the Repeat Field Box. You should also remove the border for the Field Group using the same procedure as outlined above.

CHAPTER B17 - Working with Scroll Bars

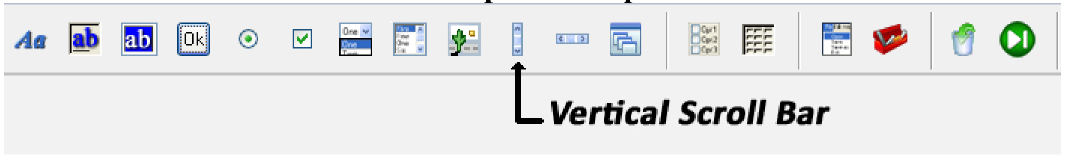
What is a Scroll Bar?



A Scroll Bar is a field which serves two purposes. The Scroll Bar is used for repeated data entry and selection fields which contain more occurrences than are visible on the panel.

The first purpose of the Scroll Bar is to act as a visual indicator to the operator that there are more fields available, than are visible. The second purpose of the Scroll Bar is to allow the operator to control the scrolling area with the mouse. Scroll Bars are available for areas which scroll horizontally or vertically.

Add a Vertical Scroll Bar to the Repeat Group



Select the "Vertical Scroll Bar" icon, move the mouse pointer inside the right border of the Repeat Group Box and click once to establish the Scroll Bar. No special properties need be set for the new Scroll Bar.

You can see that the default Scroll Bar size is too small for the Repeat Group. To automatically adjust the Scroll Bar to fit within the Repeat Group Box, double click on the Repeat Group Box border. The border will become a reverse video border, indicating that it has been selected.

Click once more in the same location. The Scroll Bar should automatically adjust to the height of the Repeat Field Box. If the right edge of the Repeat Field Box extends beyond the right edge of the Scroll Bar, simply select the Repeat Field Box and drag the right edge of the reverse video border until it meets with the right edge of the Scroll Bar. Release the left mouse button and it should adjust properly.

Create Two Push Buttons on the Order Panel

Create two Push Buttons in the lower left corner of the Order panel, assigning a Value and Help key to them. In order to easily establish the ASCII decimal key value represented by the Push Button field in the Help key property, you should:
Select the Push Button field in the Format Window for which you want to assign the Help key value.

Set the Help key property to the specific keyboard key which you want to assign to that Push Button.

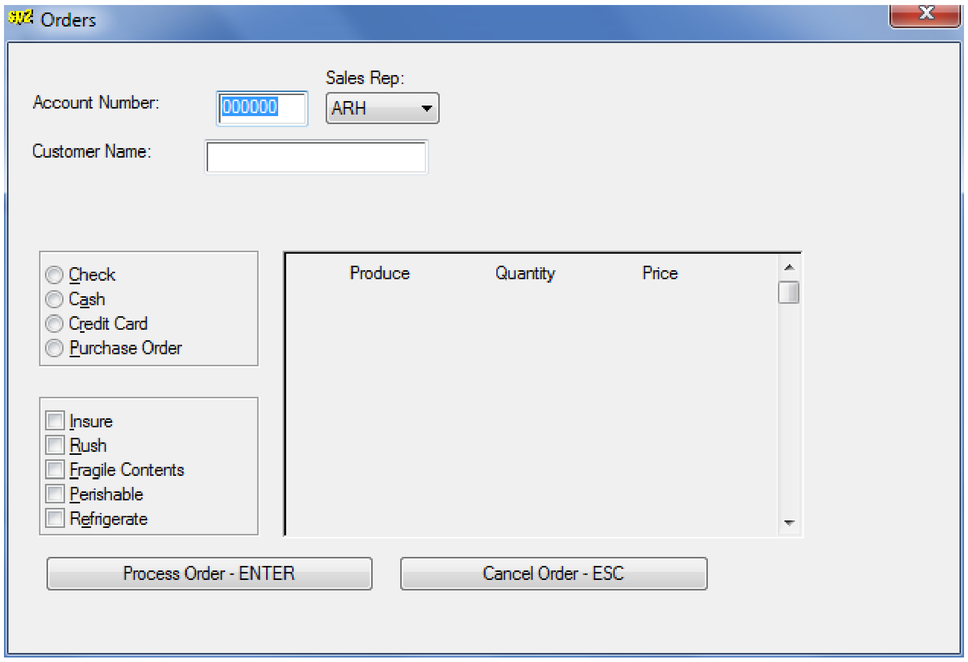
IMPORTANT – Before pressing the (ENTER) key to establish the desired key value, you MUST use the (UP ARROW) or (DOWN ARROW) key to move the cursor out of the Help key property in the Properties Box if you prefer to establish properties using the keyboard instead of the mouse. Once you have done this, you may then press the (ENTER) key and then the (ESC) key to return to the Format Window. If you do not move the cursor out of the Help key property, the key assigned to the Push Button will be the keyboard key pressed. An easier way to set the Help key value is to press the desired keyboard key and then use the mouse to reposition the cursor back into the Format Window.

The Push Buttons should have the following properties:

Value Property	**Help key Property**
~Process Order – ENTER	"013" (ENTER) key
~Cancel Order – ESC	"027" (ESC) key

Select the Test pull down option from the View menu bar option. The Order Panel will be displayed so that you may test it for appearance and function. Press the (ESC) key to return to the Format Window.

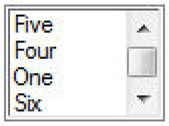
The Order Panel should look like this in Test mode:



Save the Order Panel by selecting the Save pull down option from the File menu bar option.

CHAPTER B18 - Working with List Boxes

What is a List Box?



A List Box is similar to a Repeat Group with more fields than are visible in the window. The primary differences are that List Boxes only scroll vertically and are non-enterable. List Boxes also generally contain a single field which is repeated vertically instead of multiple fields which repeat vertically.

If all of the contents of a List Box can be displayed on the panel simultaneously, the Scroll Bar will automatically disappear. When there are more options in the List Box than can be displayed at one time, the Scroll Bar will be present.

List Boxes also provide a convenient method for the operator of the application to search for the desired option within the List Box. List Boxes automatically sort the data in numeric and alphabetic order. In addition, when the first character (or a number) is typed through the keyboard, the List Box will automatically jump to the first occurrence of the character which was typed.

List Boxes are also used as selection lists, not enterable lists.

Create a New COBOL sp2 Panel

Before creating a List Box, you should create a new panel for the sample program being developed.

If you exited COBOL sp2, you may begin working immediately with the default Format Window for your new panel. If you are continuing from the previous section, select the New panel pulldown option from the File menu bar option.

Establish Various Panel Properties

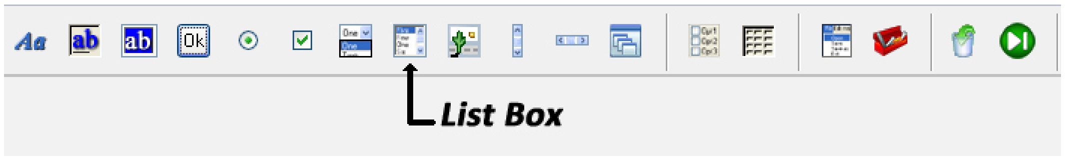
Modify the Description property and the Title property in the Properties Box to set the following properties for the panel:

Description property:	Shipping Method Panel
Title property:	Shipping Method

Add the following two fields to the panel:

Static Text	**Name property**	**Format property**
Account Number:	ACCOUNT-NUMBER	9(6)
Customer Name:	CUSTOMER-NAME	X(25)

Add a List Box to the Panel



If you are not using a mouse to design your panels, please refer to the instructions, in Chapter B2, How to Use the COBOL sp2 Panel Editor for information on adding fields with the keyboard.

Click the left mouse button with the mouse pointer on the "List Box" icon. When the pointer changes to an upward pointing arrow, move the mouse pointer below the "Customer Name" Static Text in the Format Window and click once more.

Set the Format property “X(020).”

Select the Value Property. Press and hold the (DEL) key until all the default options are deleted. Type the following lines of option text into the property, pressing the (ENTER) key after typing in each option. By pressing the (ENTER) key, you will be moving the cursor to the next line in the List Box.

Company Truck
Common Carrier
UPS Ground
UPS Blue
UPS Red
Federal Express
DHL
US Air Parcel Post
US Express Mail

The List Box is too small to see all of the text for some of the options. Increase the horizontal size of the List Box to display the entire text string for each option in the List Box. You can increase the horizontal size of the List Box by increasing the value contained in the Width property in the Properties Box, or you can stretch the field using the mouse.

Create a Static Text Heading for the List Box

The List Box would not be complete without a Static Text Heading over the options. Place the following Static Text Caption above the List Box:

Shipping Method

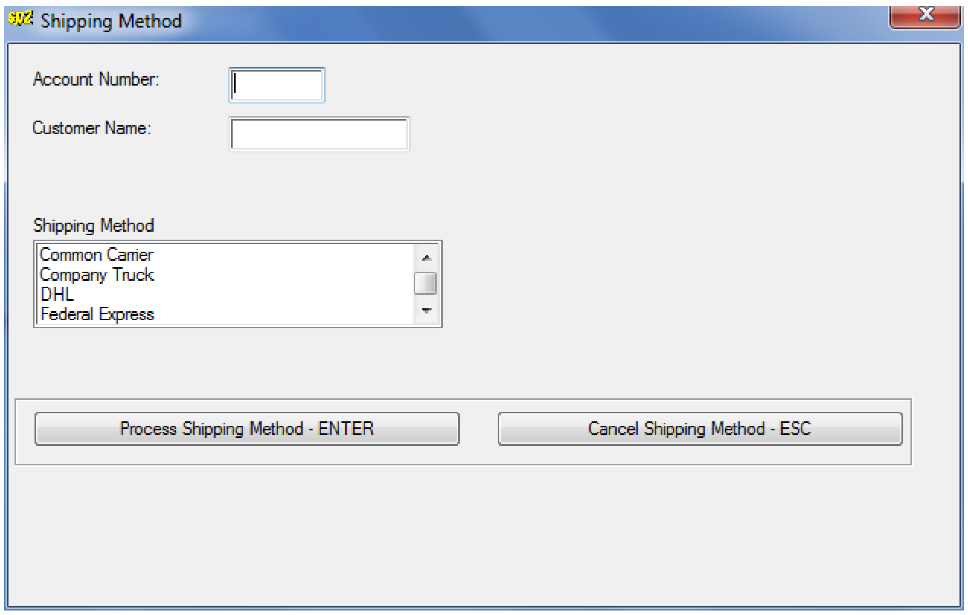
Create Two Push Buttons on the Shipping Method Panel

Before saving the panel definition, create two Push Buttons in the lower left corner of the Shipping Method panel. The Push Buttons should have the following properties:

Value Property	**Help key Property**
~Process Shipping Method – ENTER	(ENTER) key
~Cancel Shipping Method – ESC	(ESC) key

PLEASE NOTE: When you assign the Help key, simply press the actual keyboard key and then press the (UP ARROW) or (DOWN ARROW) key to exit the Help key property in the Properties Box. If you press the (ESC) key before exiting the Help key property, the value will be automatically changed to “027.”
If you test the List Box now, you will see that the list of options has been automatically sorted for you. If you return to the Format Window and eliminate all but five of the options in the List Box Value Property, the Scroll Bar will automatically be removed in Test mode.

The Shipping Method Panel should look like this:



Save the Panel Definition

Save the shipping method entry panel by selecting the Save pull down option from the File menu bar option. A small pop-up window will appear allowing you to type the name "SHIPPING." Press the (ENTER) key when you have typed in the name.

CHAPTER B19 - Establishing Field Edit Validation Criteria

What is Field Edit Validation Processing?

Field Edit Validation Processing is a set of simple rules which COBOL sp2 allows you to establish for data entry fields. The rules specify a range or table of acceptable values which may be entered into a Data Entry Field.

The advantage of using Field Edit Validation in your COBOL sp2 application is that it allows you to avoid writing simple validation rules in your COBOL source application. The data that is returned from the COBOL sp2 panel to your application will be pre-validated.

Open the Main Menu Panel

Select the Open panel pulldown option from the File menu bar option. Double click the mouse on the MAINMENU panel. You can also open an exiting panel by positioning the mouse pointer on the panel name (MAINMENU) in the Treeview Control. Click and hold the left mouse button down while dragging the pointer into the Format Window. This has exactly the same effect as selecting the Open panel pulldown option from the File menu bar option, but is much faster.

Establish a Range of Valid Values

With the cursor located on the Zip Code field you created on the Main Menu panel, select the Range property in the Properties Box.

Type "06000" in the From property. This represents the lower limit for the valid range of values for the first field.

Type "19999" in the To property. This represents the upper limit for the valid range of values for the first field. Press the (ENTER) key, then click the mouse in the Format Window or press the (ESC) key.

Establish a Table of Discrete Values

Place the cursor on the State field in the Main Menu panel and select the Discrete values property in the Properties Box.

Type "CT" in the first property. This represents the first value for the valid table of discrete values for the State field.

Using the (DOWN ARROW) key to move the cursor to the next property in the Discrete values property in the Properties Box. Type the following values, pressing the (DOWN ARROW) key after completing the text for each option:

MA
NY
NJ
PA

Press the (ENTER) key then the (ESC) key to return to the Format Window.

If you test the panel now, you will notice that the State field will not allow you to proceed if you enter an invalid 2 digit state code. The state code entered in this field must be one of those previously established and must be in upper case characters.

In addition, the Zip Code field will only allow entry of a numeric value in the range of "06000" to "19999."

Save the panel definition by selecting the Save Pulldown Menu option from the File Menu Bar option.

CHAPTER B20 - Changing the Default Font Size and Type

What is a Font?

A Font is a specific style of typeface used on the panel. Fonts can have a number of attributes including:

- The typeface name, representing the style of the font, such as Times New Roman or Arial. This text is displayed using the Times New Roman font. This text is displayed using the Arial font.

- The size of the font, expressed in terms of point size or width and height. This font size is a 10 point size. This font is a 12 point size.

- The font pitch which can be a fixed pitch (or monospace) or a proportional pitch (variable) pitch. All characters in a fixed pitch font use the same amount of horizontal space on the panel. Characters in a variable pitch font vary in width. For example, in a variable pitch font, an "i" requires much less horizontal spacing than an "m."
  Variable pitch font: mmiimmiimm
  Fixed pitch font: `mmiimmiimm`

- The weight of the font refers to the amount of emphasis which is placed on the typeface. Fonts may be normal or bold. Bold fonts are generally darker and thicker. This is a regular weight font. **This is a bold weight font.**

- Other font attributes can include: italics, strikethrough and underline.
  This font is in italics.
  ~~This font is in strikethrough.~~
  This font is in underline.

When developing or running your panels in a text mode environment, there is only one default font available. COBOL sp2 will automatically display GUI panels using the default font for these environments.

Open the Order Panel

Open the Order Panel to add 3 lines of Static Text which will be displayed with a special font.

Add Static Text to the Panel

In the upper right corner of the panel, add the following 3 lines of Static Text:

Produce
Distributors
Incorporated

Add a New Font Size and Type

Select the Font id property in the Properties Box.

The Fonts drop down list will be displayed. The two default fonts are both the system default fonts. Because the text will be changed to a new typeface, position the pointer on the Fonts drop down list and click the right mouse button. A small pop up menu will be displayed allowing you to establish a new font or preview an existing font. Select the New menu option to establish a new font.
The "Font" pop-up window will be displayed to allow you to specify a new font. Add the following to the Font pop-up window:

Font:	Times New Roman
Font style:	Select "Bold Italic"
Size:	18

Press the (ENTER) key or click the "Ok" Push Button to return to the Font selection pop-up window.

Change the Default Font

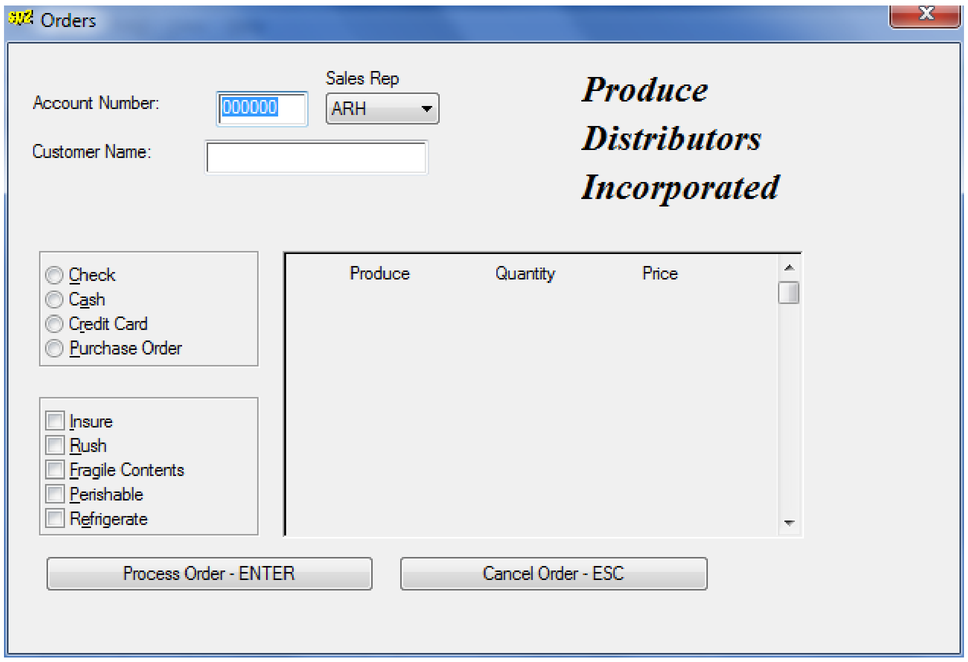
You will now see that the new font has been added to the drop down font list of available fonts for your panel. Highlight the "Times New Roman" font and press the (ENTER) key. Press the (ESC) key to return to the Format Window.

The Static Text is now larger than the original text. This means that you must increase the size of the Static Text. First, make sure that the other Static Text, "Distributors" and "Incorporated" are not too close to the word "Produce." The Static Text "Incorporated" should be moved down two cell rows. The Static Text "Distributors" should then be moved down one cell row.

Increase the size of the Static Text in the same manner as you increase the size of any field. Repeat the procedure for the remaining two Static Text values.

To understand fonts more thoroughly, please refer to section B34 and read the section entitled Stock Fonts.

The Order Panel should look like this in Test mode:



Save the panel definition by selecting the Save Pulldown Menu option from the File Menu Bar option.

CHAPTER B21 - Changing Default Colors

Default Color Schemes

You may wish to keep the default color scheme for the panel and panel fields. The reason for this suggestion is that in most GUI environments, the operator has the ability to customize the colors in their system user interface. This allows the operator to choose the colors that they prefer.

Another reason for keeping the default color scheme is if you are using a 3-dimensional look and feel (see chapter B4). Changing colors may degrade the 3-D effects in your panel.

When you use the COBOL sp2 default color scheme, COBOL sp2 will display the panels and panel fields based upon the defaults established in the system. This means that if the operator changes the system default colors, the panel and panel field colors in your COBOL sp2 based application will also change to the new color scheme.

If you design your panels with a custom color scheme, COBOL sp2 will maintain those custom colors even when the operator changes the system default colors in the GUI environment.

Having said all of the above, it is important that you know how to change colors and this chapter explains how to do this.

Establish Special Color Schemes for your Panels

Making sure that the cursor is located in a general panel area, select the Color property in the Properties Box. The Color property drop down list will be displayed. The Color property drop down list allows you to choose a color combination or create a new color combination for the panel.

To establish a new color scheme for the panel, you must first update the color set. Click the right mouse button on the Color property drop down list. A small pop up menu will be displayed allowing you to establish a new color or preview an existing color.

Type "Black on Cyan" in the Name property on the Color Definition pop-up window. Next, press the (TAB) key to position the cursor in the Foreground Color Combination Box. Press the (DOWN ARROW) key or click the mouse on the downward pointing arrow at the end of the Combination Box to choose the Foreground color. Scroll down the drop down list to display the color, "Black." Select the foreground color of Black by highlighting the text and pressing the (TAB) key or clicking the mouse once on the text.

If you pressed the (TAB) key in the foreground color combination box, the cursor is now in the Background color combination box. Press the (DOWN ARROW) key or click on the downward pointing arrow at the end of the Background Color Combination Box to choose the Background color. Scroll down the drop down list to display the color, "Cyan". Select the background color of cyan by clicking on the text once with the mouse. Click on the "Ok" Push Button or press the (ENTER) key to add the new color combination to the List Box on the Colors pop-up window.

Select the Black foreground on Cyan background color combination for the panel. Press the (ENTER) key to add the new color combination to the panel and then press the (ESC) key to return to the Format Window. You will see that the panel color has been changed to black foreground characters on a cyan background.

CHAPTER B22 - Changing the Border Type

Group Borders

COBOL sp2 allows you to replace the default group border with a variety of border styles, including:

- no border
- default border
- 3-dimensional border
- thin 3d border - This border is only one pixel wide which is useful if space is limited.
- system 3d group border (like system entry field)
- single line border
- thick line border
- raised border
- raised and thick
- lowered border
- lowered and thick
- engraved border
- engraved and thick
- rimmed border
- rimmed and thick

Change the Border for a Group

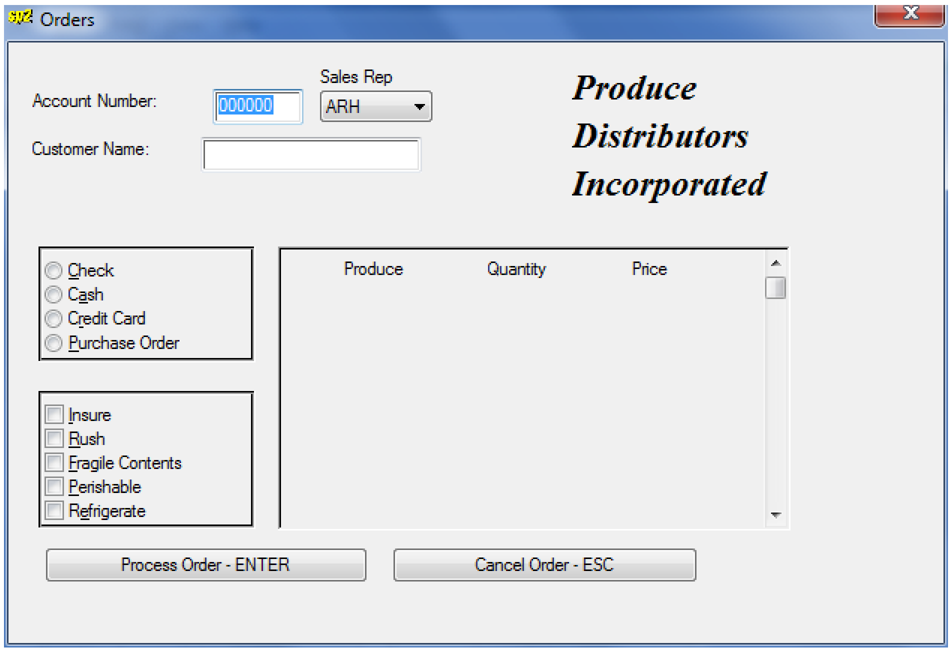
The borders in the Radio Button and Checkbox groups will be changed to an “engraved and thick” border type. Position the mouse pointer on the Radio Button group click once to move the cursor to the group.

Set the Border type property in the Properties Box to “D = engraved and thick.”

If you prefer to use the keyboard, first make sure that the cursor is located directly on the group for which you want the special border. Press the (ESC) key to activate the Properties Box. Position the cursor in Border type property. Use the (DOWN ARROW) key to scroll down through the border type options until the option, “D = engraved and thick” is highlighted. Press the (ENTER) key to select the desired option and then press the (ESC) key to return to the Format Window.

Repeat this process for the Checkbox group on the panel.

The Order Panel should look like this in Test mode:



Save the panel definition by selecting the Save Pulldown Menu option from the File Menu Bar option.

CHAPTER B23 - Working with Input Types

Create a New COBOL sp2 Panel

Before working with Input Types, you should create a new panel for the sample program being developed.

If you exited COBOL sp2, you may begin working immediately with the default Format Window for your new panel. If you are continuing from the previous section, select the New panel pulldown option from the File menu bar option.

Establish Various Panel Properties:

Set the Description and Title properties as follows:

Description Property:	Current Account Balance Panel
Title Property:	Current Account Balance

Add the following two fields to the panel:

Static Text	Name property	Format property
Account Number:	ACCOUNT-NUMBER	9(6)
Customer Name:	CUSTOMER-NAME	X(25)

What is an Input Field Type?

An Input Field Type represents the type of allowable data which may be entered into a Data Entry Field. For both Custom Data Entry Fields as well as System Default Fields, the Input Field Type may be Alphanumeric (any data), Numeric or Date.

A Field Type of Alphanumeric (any data) will allow entry of any keyboard character as the name implies. A Numeric Field Type will only allow entry of numeric data and a Date Field Type will allow entry of a valid date.

Once the Input Field Type is set for the Type property, it is still necessary to establish an appropriate COBOL Field Format in the Format property.

Establish a Special Password Field

Create a line of Static Text and place it to the right of the Account Number Field. Set the Text property to "Password:" Add a Custom Field to the right of the Static Text you just created.

Set the Format property to “X(4).”

The Password field will be enhanced for security purposes later. At program execution, it will be dynamically changed to a Display Only field so that the password must only be entered on the initial screen display.

Establish a Numeric Field

Create a line of Static Text and place it directly below the Customer Name Static Text. Set the Caption property to "Current Account Balance." Add a Custom Field to the right of the Static Text you just created. Increase the size of the Custom Field by 16 cell columns (Set the Width property to 210).
Set the Format property to “$(7).99CR.”
Set the Type property to "Numeric."

Establish a Date Field

Create a line of Static Text and place it directly below the Current Account Balance Static Text. Set the Text to "Current Due Date:". Add a Custom Field to the right of the Static Text you just created. Increase the size of the Custom Field by 7 or 8 cell columns (Set the Width property to 120).

Set the Type property to "Date."

Set the Format property to “MM/DD/YYYY.”

CHAPTER B24 - Working with Protection Types

What is a Protection Type?

A Protection Type determines whether or not the field is enterable, accessible or display-only. For both Custom Data Entry Fields as well as System Default Fields, the Protection Type may be Unprotected, Protected, Display-Only, Secure or Hidden.

- An Unprotected Field will allow entry of data into the field based on the Input Type and COBOL Field Format.
- A Protected Field will allow the cursor to access the field, but not allow data to be entered into the field. These are useful for creating your own selection fields in a Repeat Group.
- A Display Only Field will not allow the cursor to access the field, nor will it allow data to be entered into the field. The Display Only Type is useful for displaying messages on the panel.
- A Greyed out Field is a field that cannot be accessed or written to. Any text that may appear in the field will be shaded out. This is useful for showing text that cannot be changed.
- A Secure Field will allow data to be entered into the field, but the data will not be displayed on the panel. This Type is useful for password fields.
- A Hidden Field will be invisible. The field may be made visible at runtime.

Once the Protection Type is selected, it is still necessary to establish an appropriate COBOL Field Format in the Format property in the Properties Box.

Establish a Protected Field

Move the cursor to the Account Number Field in the Format Window.

Set the Protection property to “p = protected option.”

Establish a Display Only Field

Move the cursor to the Customer Name field in the Format Window.

Set the Protection property to “y = display only.”

Establish a Greyed Out Field

If you would prefer to make the Customer Name field Greyed Out rather than Display Only, repeat the above steps but set the Protection property to "Greyed out."

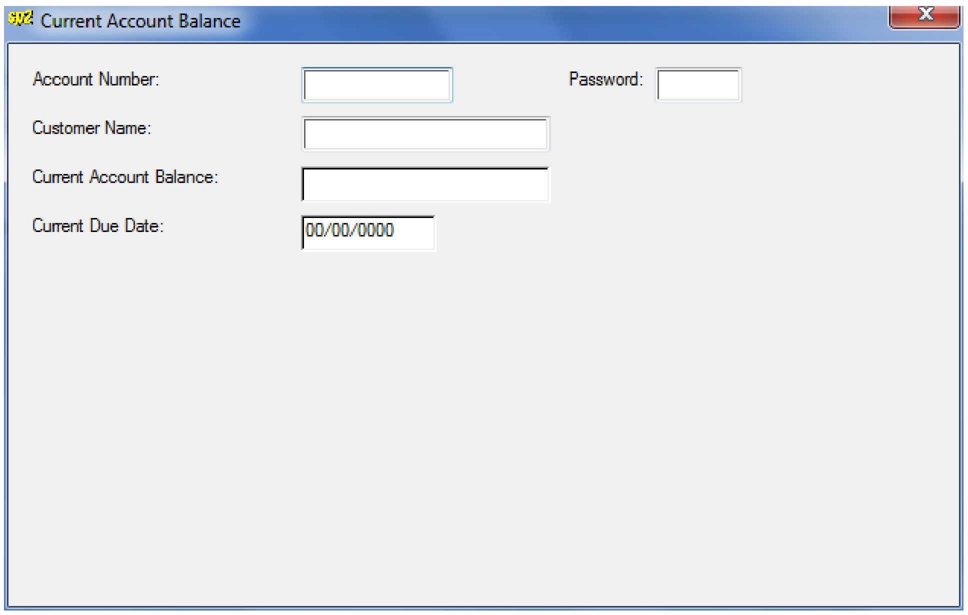
Establish a Secure Field

Move the cursor to the Password field in the Format Window.

Set the Protection property to “s = secure.”

When the Protection type of Secure is established and the operator types data into the field, the data is automatically replaced with an asterisk, (*) character.

The Current Account Balance Panel should look like this in Test mode:



Save the Panel Definition

Save the current account balance panel by selecting the Save pull down option from the File menu bar option. A small pop-up window will appear allowing you to type the name "CURRACCT." Press (ENTER) key when you have typed in the name.

CHAPTER B25 - Returning Control to Your COBOL Program from COBOL sp2

A Discussion of COBOL Program Control

COBOL sp2 is an external program which is "CALLed" from your COBOL application through the use of an ANSI standard COBOL "CALL USING..." statement. Because control is transferred from the application to the COBOL sp2 Runtime System, it is important to establish when to transfer control back from COBOL sp2 to your application.

COBOL sp2 provides a number of ways in which control may be returned to your application. The most common technique is through the use of a set of Control Keys which you establish in the COBOL sp2 Panel Editor. Control keys will always return control directly to the COBOL program along with the decimal value of the key when a particular control key is pressed. The default Control Keys for a new panel are the ENTER and ESC keys. For more information on Control Keys, please refer to Chapter B26, Establishing Control Keys for the Panel.

Another technique for returning control back to the COBOL program is through the use of Push Buttons or Icons which are associated with specific Control Keys. More information on Push Buttons may be found in Chapter B8, Working with Push Button Fields. More information on Icons may be found in Chapter B10, Working with Icon Fields.

If it is necessary to return control from COBOL sp2 to your program on a field-by-field basis or return control only for specific panel fields, you should establish Control Fields. Control Fields are normal data entry or selection fields which will automatically return control from COBOL sp2 back to your COBOL application when the cursor exits the field.

There are five Control Field options including, No = no return to program, a = return on exit, m = return if modified, s = return if selected and o = return on exit or select. Each is discussed as follows:

- No = no return to program will not return control when the cursor exits the field.
- a = return on exit will always return control when the cursor exits the field.
- m = return if modified will only return control when the cursor exits the field if the contents of the field have been modified.
- s = return if selected will return control for selection fields when the selection field is clicked with the mouse or the (SPACEBAR) key is pressed.
- o = return on exit or select will return control when the cursor exits the field or when the field is selected with the mouse or spacebar.

Return to the COBOL Program Control Automatically

The Password Field which was established in a previous section is to provide a method of securing the sensitive Customer Account Balance information. Only those employees with a valid password will be allowed access to the information in this panel. Before data can be displayed on this panel, COBOL sp2 must return control to the COBOL program so that the password can be checked for validity.

Set the Program control property to “m = return if modified."

The return if modified setting will cause COBOL sp2 to automatically return control back to the COBOL program if the contents of this field have been modified. If you want to return control every time this field is accessed, you should set the Program control property to “a = return on exit.”

Save the panel definition by selecting the Save Pulldown Menu option from the File Menu Bar option.

CHAPTER B26 - Establishing Control Keys for the Panel

Control keys are keys which return control to your COBOL application when pressed. Because Push Buttons are generally used to invoke a process in your application, you can assign a Control Key value to a Push Button.

When you assign a Control Key to a Push Button in COBOL sp2, the key value is automatically added to the list of Control keys for the panel on which you are working.

If you want other keys which are not represented by Push Buttons to return control to your COBOL application, you must edit the list of Control keys.

Set the Control keys property to the three digit decimal key value for the key which you want to add as a Control Key. You can also establish the decimal key value by simply pressing the key which you want to establish as a control key. For a list of Decimal Key Values and the keys which the represent, please refer to Appendix A, Decimal Key Values.

The default Decimal Key Values are 013 (ENTER) key and 027 (ESC) key.

It is also possible to delete a Control Key by highlighting the Decimal Key Value and pressing the (DEL) key. You can also highlight the key value and right mouse click on the drop down list to display a menu of editing options for the key value. When you have established the desired list of Control Keys, press the (ENTER) key to save your work and return to the Format Window.

CHAPTER B27 - Requiring Data Entry for Data Entry Fields

What is a Required Data Entry Field?

Required Data Entry Fields allow you to force the operator of your application to enter data into a Data Entry Field. COBOL sp2 will prevent the operator of the application from tabbing out of the required field, or moving the cursor to a different field without entering valid data.

If the operator is unable to enter a valid data item into the field, an Edit Override Key can be pressed to allow the operator to escape from the panel without entering any data. Any Control Key may be assigned as an Edit Override Key.

Establish Data Entry Requirement for a Field

The Password field should be required for the operator of the program. This will prevent the operator from leaving the Password field until a password is typed into the field. Set the Required property to “y = yes.”

This will cause COBOL sp2 to prevent the cursor from leaving the field until something has been entered into the field.

Save the panel definition by selecting the Save Pulldown Menu option from the File Menu Bar option.

CHAPTER B28 - Automatic Cursor Tabbing

What is Automatic Cursor Tabbing?

Automatic Cursor Tabbing is useful for Data Entry Fields which require a specific number of characters to be entered. When Automatic Cursor Tabbing is switched on for a field, the cursor will automatically move to the next field when the last character is entered into the field.

Automatic Cursor Tabbing can be set for all panel fields or only for specific fields.

Open the Orders Panel

Open the ORDER panel to establish automatic Tabbing.

Establish Automatic Tabbing for Specific Fields

Select the Produce field and set the Usage option to “y = skip to next field.” This will cause the cursor to jump to the "Quantity" field when a 10 character value is typed into the field.

Repeat for the Quantity field.

Select the Price field and set the Usage option property to “n = do not skip to next field.” This will cause the cursor to remain in the field until the (TAB) key is pressed.

Establish Automatic Tabbing for All Data Entry Fields

If you want to have automatic tabbing for all panel fields, you would not change the Usage option from the default value of “Default = panel setting.” Instead you should set the Cursor skip property to “yes.”

Save the panel definition by selecting the Save Pulldown Menu option from the File Menu Bar option.

CHAPTER B29 - Special Field Formats for Data Entry Fields

What is a Special Field Format?

Frequently, COBOL applications require that certain fields allow data to be entered in an unconventional format. These fields must include special imbedded characters which are ignored by the cursor. Some examples of this include telephone number fields, serial number fields or identification numbers.

COBOL sp2 provides the ability to allow you to create a special field format which can accommodate these special imbedded characters. Special Field Formats also allow data types to be mixed. This would allow you to create a field which allows several alpha-only characters followed by several numeric-only fields.

Establish a Special Field Format

The Account Number field will be given a special COBOL field Format. The field was previously assigned a Picture Clause of 9(6). The Field Format will be changed to allow for special formatting. First, position the cursor in the Account Number field.

Set the Special format property to “y = special format.”

Set the Format property to "99-9999."

Save the panel definition by selecting the Save Pulldown Menu option from the File Menu Bar option.

CHAPTER B30 - Assigning a Special Field Tabbing Sequence

What is Field Tabbing Sequence?

COBOL sp2 allows you to assign a specific sequence for the field tabbing behavior on a panel. This allows you to customize the field-to-field tabbing behavior so that the cursor does not follow the traditional "left-to right, top-to-bottom" cursor tabbing sequence.

This is particularly useful if your panels incorporate fields which are repeated in columns.

Open the Order Panel

Open the Order Panel with the Open panel Pulldown Option from the File Menu Bar Option or if the panel file is already open, simply drag and drop the Order panel from the Treeview Control into the Format Windows with your mouse to establish a special field tabbing sequence.

Assign a Special Field Tabbing Sequence

Using the Mouse

Position the cursor on the Sales Rep field in the Format Window and view the Tab number property. The Tab number property should display a “2” which indicates that this field is the second field in the tab number sequence. Do not change this setting.

Move the pointer to the first field in the Radio Button Group (Check) and click once. Set the Tab number property to 3. So far, you have set the tab sequence for the Sales Rep field and the Radio Button Group.

You can set the remaining tab sequence numbers by assigning the tab sequence number in the Tab number property.

Using the keyboard to assign the Tab number property in the Properties Box may be a bit easier, because it allows you to assign the tab sequence number for each field without switching back and forth from the keyboard to the mouse and vice versa.

PLEASE NOTE: Any fields not selected during the tabbing sequence assignment process will process in the default "left-to-right/top-to-bottom" sequence after the cursor moves to those fields which have been assigned a special sequence.

Save the Order Panel

Save the panel definition by selecting the Save Pulldown Menu option from the File Menu Bar option.

CHAPTER B31 - Establishing Multi-Line Data Entry Fields

What is a Multi-Line Data Entry Field?

Multi-Line Data Entry Fields are single fields which allow the operator to enter multiple lines of data. The number of enterable lines of data depends on the vertical height of the field and the size of the font used in the field.

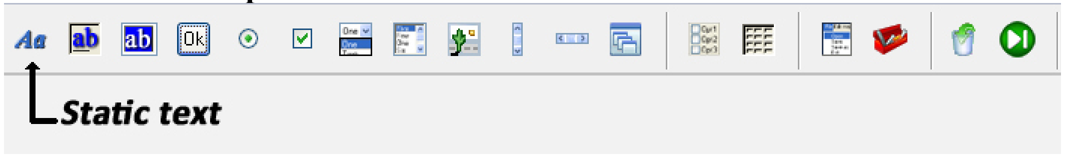
Multi-Line Data Entry Fields must be created using System Default Fields. Unlike the default characteristic of System Default Data Entry Fields, Multi-Line Fields do not allow the operator to enter an unlimited amount of data. Multi-Line Data Entry Fields limit the amount of enterable data to the size of the field on the panel.

Multi-Line Data Entry Fields also provide automatic word wrap. Unfinished words typed at the end of a line will automatically move to the beginning of the next line as long as the operator is not at the bottom line of the field.

Open the Current Account Balance Panel

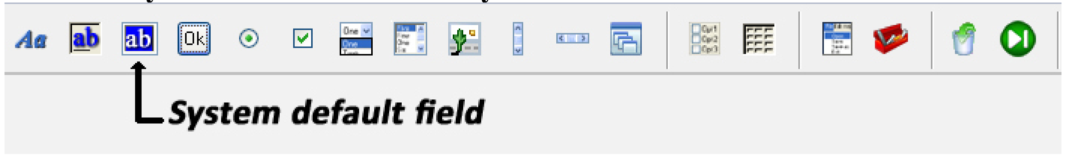
Open the CURRACCT panel to establish a Multi-Line Data Entry Field.

Create a Field Caption with Static Text



Click the mouse pointer on the "Static Text" Icon and move the pointer just below the "Current Due Date" Static Text and click once to establish the static text. Set the Text property to "Special Notes."

Create a System Default Data Entry Field



After adding the field caption with the static text, add an entry field to correspond to the text. Click the mouse pointer on the "System Default Entry Field" icon.

Establish a Multi-Line Data Entry Field

Set the Usage option property to "m = Multi-line."

Change the Size of the System Default Entry Field

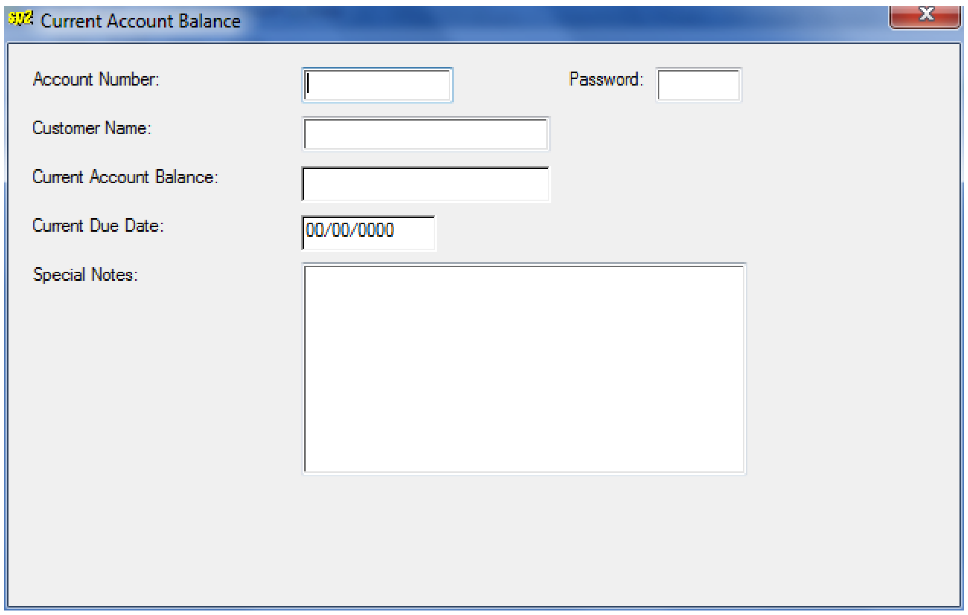
To create a true Multi-line Entry Field, the System Default Entry Field must be increased in size. Use the same procedure as you have for the other fields which were previously increased in size. You can increase the size of the field both horizontally and vertically. Make sure that you have increased the vertical size of the field so that you will have at least two or more lines of available text to enter.

HINT: Increase the Height property to 50 and increase the Width to 350 on the Multi-line Data Entry Field.

For more information on cell rows, please refer to Chapter B2, How to Use the COBOL sp2 Panel Editor.

For more information on changing the size of a panel field, please refer to Chapter B2, How to Use the COBOL sp2 Panel Editor.

The Current Account Panel should now look like this:



Save the panel definition by selecting the Save Pulldown Menu option from the File Menu Bar option.

CHAPTER B32 - Help for your Panels

Types of Help

Two types of help information can be set up for your panels. The first type of help is "standard" Windows help which allows you to set up help text in a special format and have it managed by the Help software included with Windows. The advantage of this is that the help information will be presented in a form that is familiar to Windows users and they will find it easy to search and navigate through the text. The disadvantage is that a good deal of extra work is involved in formatting the text so that it can be used by the Windows help system. The best thing to do is to obtain software that will help you to do this work. Contact your software supplier or search the web for "Help authoring software."

The second type of help is COBOL sp2's own internal help system. This system uses regular text files that are displayed in windows created automatically when the user requests help. The advantage of this is that it is very easy to create these text files – any text editor can be used. The disadvantage is that the help information is presented in a far less glamorous way than "standard help" – there is no index or search facility, for example.

Whichever type of help you decide to use, a particular section of help information will normally be linked to a particular panel so that the appropriate information is presented to the user when help is requested. It is also possible to link help information to a particular field so that the help is even more context-sensitive.

Setting Up Standard Help

As explained above, the first thing that must be done is to set up the help information itself in a Windows format using software designed for that purpose. Help information may be configured in Winhelp format (.hlp files) or HTML format (.chm files) though earlier versions of Windows only support the former and later versions only support the latter. To allow a particular section of the help information to be linked to a panel or field, you must set up a Keyword for that section. Your help authoring software will enable you to do this.

Once your help information has been configured, you can set up your sp2 panels to invoke the Windows help system to display this information. The first thing to do is to specify the name of the help file that is to be used. The easiest way to do this is to set the name of the help file so that it corresponds to the name of your panel file (or the first panel file used in your system). For example, if your panel file is called "myfile.pan," your help file should be called "myfile.hlp." If this is not convenient, you can set the panel Help keyword property to reference the name of your help file. The name of the file should be preceded and followed by a "/" (forward slash) character.

To link your help information to a particular panel, set the panel Help key property to the key that your user is to use to invoke the help system (see appendix A for a list of keys and their codes). When you run your system and the user presses this key, the help information will be displayed. (Note that you cannot test this feature from the editor unless you specify the help file using the Help keyword property.) If you would prefer your user to be able to invoke the help system by clicking on a push button or icon rather than pressing a key, set the pushbutton or icon Help key to be the same as the panel Help key. You can also configure a menu to access help – a menu id of 998 will display help in the same way as the Help key and an id of 999 will display the Table of Contents for the help information.

Help information is linked to a particular field in the same way. Set the field Help key property to the key to be used to display the help information and optionally define a pushbutton or icon with this same Help key. If a field is contained in a dialogbox, you can also use the special "question mark" icon in the dialogbox titlebar to invoke the help system. To enable this feature, set the panel Options-3 property to activate "window help." Your user will now be able to click on the icon and then click on the field to display the help information.

For either a panel or field, the section of the help information to be displayed is derived from the Help keyword property. This should be set to the keyword defined within the help information and associated with the appropriate section, as explained above. Alternatively, it can be set to the value "HELP_CONTENTS" to cause the table of contents to be displayed. (If you are using the Help keyword property to identify the help file to be used, place the keyword itself after the file name.) If you do not set the Help keyword property, sp2 will attempt to use the panel Description property or the field Name property as a keyword.

Setting Up Internal Help

Internal help is accessed in mostly the same way as "standard" help using the panel and field Help key properties, but the Help keyword property is not used. Instead, the Help panel property is used. If this property is set to the name of a panel, that panel will be displayed. If no panel with that name exists, sp2 will search for a file with that name and an extension of "SP2." This file can contain as many lines as you like but the lines should be no longer than 80 characters.

CHAPTER B33 - The Icon Toolbar and Bitmap Buttons

What is an Icon Toolbar?

A toolbar is a collection of fields (normally bitmap buttons) which is displayed in a separate area immediately above the main panel area. These buttons improve the overall look of the panel and offer the operator more ease of use when using the mouse. Fields and groups can be placed in the toolbar area just as in the main panel area.

Setting up an Icon Toolbar

To setup an Icon toolbar, select the specific toolbar definition desired from the Toolbar property drop down list in the Properties Box. If there are no existing toolbars available in the Toolbar property drop down list, select the New toolbar option. This will place a line across the window that acts as a divider which separates it into two different areas. To activate the toolbar area, either click in the area. When you click in the toolbar area, you will notice that the Object type property in the Properties Box changes from “Panel” to “Toolbar panel.”

The height of the toolbar can be adjusted by dragging the bottom border (also referred to as the frame) of the toolbar.

Establish fields in the toolbar just as for the main panel area. Bitmap Push Buttons tend to be the most common fields in toolbars. Use the instructions below to create bitmap buttons.

Options for the Icon Toolbar

Click the mouse pointer in the Toolbar panel area to edit the toolbar.

To change the font of the icon toolbar select the Font id property in the Properties Box. Then select the font type needed from the drop down list, or right mouse click on the Font id drop down list and select the New pop up menu option to create a new font. For more information about changing fonts, refer to chapter B20.

Set the Color property to the color desired by clicking on one of the choices on the drop down list. To create a new color, right mouse click on the drop down list and select the New option on the pop up menu. For more information about changing colors, please refer to chapter B21.

Deleting a Toolbar

If the Toolbar is no longer needed in the panel, set the Toolbar property to “No toolbar.” This will remove the toolbar from the panel.

Bitmap Buttons

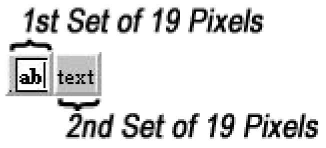
Icon fields can be made to automatically behave like a button (pushbutton, radiobutton or checkbox) without the need to include the button border in the bitmap and have a second bitmap representing the depressed button (see chapter B10). The button image can also be extracted from a single bitmap containing multiple button images. To create a bitmap button, do the following:

Create an icon field and size it if necessary (the default size is 24 X 22 pixels).

Set the Type property to “x02 = bitmap-pushbutton”, “x03 = bitmap-radiobutton” or “x04 = bitmap-checkbox.”

In order to set the bitmap image to be used, select the Value property and click on the small downward pointing arrow at the right side of the Icon file property. This will display the File Open dialog box in which you will be able to select the desired icon file.

It was mentioned above that the button image may be extracted from a single bitmap containing multiple button images. This is accomplished by specifying the specific set of pixels contained in a graphic image in the Icon details of the Value property in the Properties Box. In the example below, the bitmap image consists of 2 sets of images, each image is 19 pixels in width. For example, to select the second image in a strip of 19 pixel images, enter ",images.bmp, 2/19". If the number/size is not entered, the default is 1/16.



Optionally set the border type to be “r = rounded border” or “s = square border” using the Border type property.

Optionally set the icon field to be “w = windowed” using the Usage option property. The main advantage of a windowed icon is that it allows for the display of "hints" or "tooltips" (see below).

Windowed Icons

An icon field can be contained in its own window. There are two advantages to this: first, a windowed icon that is depressed will return to its normal state if the mouse is dragged off the icon; second, a windowed icon may have an icon hint or "tooltip" assigned to it. To set up an icon hint, enter support text using the Message text property and this text will be displayed in a small window when the mouse pointer is moved over the icon and left there for a small period of time. Icon hints are only allowed for display-only windowed icons.

Do not use windowed icons for background bitmaps.

Dynamic Borders for Bitmap Buttons

If Border is set to "d" for a windowed bitmap button, then a border will only be drawn for the button when the mouse is over it. In addition, the color of all pixels matching the top left pixel of the image will be reset to match the Field or Panel color. These two features allow you to design contemporary toolbars like those found in later versions of Windows.

Multiple Image Bitmap Buttons

If the field Miscellaneous property X"20" bit is set and the Item length property is set, then the Value property should contain three image specifications (each Item length long): the default image; the image to be displayed when the mouse is over the button; and the image to be displayed when the button is grayed out. In the editor, Item length is automatically set based on the Format by dividing the Format length by 4. For example, if Format is X(100), each image file name can be up to 25 characters long. The second and third images should be entered into Item details separated by a "|" character, for example:

icons.bmp 2/20 | icons.bmp 3/20

Field Border must be set to "d" and Usage to "w" to get the "mouseover" effect.

Setting this option causes the program Fields area item for the icon field to be formatted in a special way. The item is divided into 4 slots: the default image; the image to be displayed when the mouse is over the button; the image to be displayed when the button is grayed out; the hint text to be displayed when the mouse is over the button. If you do not need to modify the images associated with a bitmap button from your program, set the Program data property to X"02" and set Program length to zero - this will suppress the generation of the program Fields area item for the icon. See below for more details on this.

Program Access to a Toolbar

Normally, the editor will generate items in your program to allow icons to be grayed out, etc. This is not desirable if you plan to assign a single toolbar to multiple panels because the generated program items will only be accessed correctly for the panel for which the toolbar was originally designed. To avoid this problem, set
Program data to X"06" and set both Program length and Program number to zero for all the fields within the toolbar.

CHAPTER B34 - Cell Size and Scaling

Cell Size

In chapter B2, it was explained how fields always snapped to a cell row indicated by the grid in the editor format window. This makes it easy to line up fields in a consistent manner. However, it may sometimes be desirable because of space or alignment considerations to position fields off the grid. This is achieved by adjusting the cell size.

To adjust the cell size, first make sure that the cursor (focus) is located on a general panel area and not on a specific field. Next, note the values in the Cell width and Cell Height properties. You should see that the cell size is set to a width of 8 pixels and a height of 16 pixels (the default). You can reduce these values to as low as a width of 1 and a height of 1 - this would allow you to position fields anywhere within the panel. The disadvantage of small cell sizes, however, is that it becomes more difficult to align fields satisfactorily. It is best to pick the largest cell size that still gives you the desired precision for field positioning.

If you do want to use a cell width and a cell height of 1 in order to be able to position fields with great precision, then positioning should be accomplished with the Row and Column properties. These properties allow you to establish the location of the field numerically instead of through a “dragging and dropping” procedure.

It is also possible to select a group of fields and use the Edit/Align Menu Bar option to horizontally or vertically align the entire group.

Cell Size and Stock Fonts

Stock fonts are fonts that the system picks to fit the screen size being used i.e. the same stock font may appear differently on two different screens. If you choose to use stock fonts, it is a good idea to only use stock fonts because other fonts will not vary depending on the screen size. The following stock fonts are available:

-3 =	large
-4 =	small and bold
-5 =	small
-7 =	fixed pitch

If you use stock fonts, you should also reset the cell size to be based on the stock font rather than a fixed number of pixels. If you don't do this the size of the font will change from screen to screen, but the panel size (which is based on cell size) will not. Use the Cell units property in the Properties Box to set the unit for cell size to be font size rather than pixels. If this option is greyed out it means that you have not selected a stock font as your default panel font using the Font id property.

Scaling with Stock Fonts

Once cell size is based on font size, panels will automatically be scaled in relation to the stock font. If the stock font being used is larger on a particular machine, then your panels will also be that much larger. This means that your application's user interface is likely to be very consistent with the interface of other applications, at least from the perspective of fonts and window size.

Scaling without Stock Fonts

One disadvantage of using stock fonts is that it is difficult to use other fonts in conjunction with stock fonts. This is because these other fonts are based on a fixed pixel size and so will appear the same on all machines. The solution to this problem is to forgo use of stock fonts and use regular fonts in conjunction with scaling based on screen size.

To scale based on screen size set up two configuration variables (see Appendix B for details of setting up configuration variables) representing original screen width and screen height. These values represent the size of the screen of the screen on which your panels were designed. For example, if you did your design work on a super VGA screen which is set to a horizontal resolution of 800 pixels and a vertical resolution of 600 pixels (800 x 600), enter the following:

```
SP2OWD=800
SP2OHT=600
```

Using the above values, a panel that takes up the entire area of an 800 x 600 resolution super VGA screen will be scaled up or down to take up the entire area of any other screen (640x480, 1024x768, etc.)

CHAPTER B35 - More on Icons

Alternative Bitmap Formats

You can use other graphic file formats than .BMP by purchasing an add-on product to the COBOL sp2 Development System. Please contact Flexus for additional information.

If a regular bitmap file uses 32 bits per pixel the final 8 bits will be treated as a measure of the transparency of a pixel - this is known as the Alpha Channel.

Audio and Video Icons

Just as icon fields can be associated with bitmap images, they can also be linked to audio and video files so that you can play audio and video clips.

Establish an icon field as normal.

Set the Type property to "a = audio-play" or "v = video-play."

Select the Value property and click on the small downward pointing arrow on the right side of the Icon file property to display the File Open dialog box. This will allow you to select the desired WAV or AVI file. When you test the panel, the audio or video clip will be automatically played.

OCX Controls

Icon fields can also be associated with OCX (ActiveX) controls if you have the ActiveX Add-on package.

Establish an icon field as normal.

Set the Type property to x"05" = activex control.

Select the Value property and click on the small downward pointing arrow on the right side of the Icon file property to display the ActiveX Controls dialog box. This will allow you to select the Activex (OCX) file that you wish to embed in the panel definition. When you test the panel, the activex controls will display in the panel definition.

Once the Activex control is loaded into the Panel Editor, the ActiveX Control Properties are automatically added to the Properties Box so that you can set the specific default attributes (properties) of the Active X Control which you have selected.

Some OCX controls are very complex and may need significant programming effort to get them to work properly.

For any panel that contains OCX controls, set the panel Options-3 X"20" switch – clip children.

Long Icon File Names and Controlling Icons from a Program

Long file names are supported for icon fields. In the editor, set the Format property to x(50), for example, and this will cause the Maximum length, Program length and Item length properties to be set to 50. This will allow for up to 50 bytes for the icon file name.

Because the Program length property has been set, an entry will be generated in the program Fields area for the icon and you can use this entry to easily change the image that is displayed or set the image for individual occurrences in a repeat group.

This facility is currently only supported for icon fields with Type property set to x"00" (regular bitmap), x"01" (bitmap exact) and x"02" (bitmap pushbutton). If you need to set a bitmap pushbutton from your program, include the number and width of the image (for example "1/20") after the file name, leaving at least one space.

CHAPTER B36 - Container Panels

Introduction to Container Panels

Container panels allow the display a group of panels to be controlled by the runtime without need for program intervention. You can define how the panels within the group interact within the panel editor. "Contained" panels are called "subpanels" and they are linked to the container panel using "subwindow" fields. Container panels are particularly useful for handling tab panels.

When code is generated for a container panel, the converse-data area will contain definitions for the fields in the container panel and all its subpanels. When any panel is modified in a panel file that holds container panels and that panel file is closed, the editor will issue a prompt to regenerate any main container panels. This is to ensure that the offsets of fields within the main program Fields area are set properly. This also means that a panel should not be used as a subpanel of more than one container and should not be used independently if it is used as a subpanel.

A container panel is identified as such by setting the X"80" switch of the panel Options-4 property and should contain subwindow fields as necessary. For each subwindow field, a subpanel should be defined separately. The position of the window containing the subpanel is controlled by the Row and Column properties of the subpanel, not the subwindow field.

Subwindow fields

A subwindow field is a field with the Control type property set to "w". It's position within a container panel is normally not important as it is not visible at runtime, only in the panel editor.

It's Value property defines the panel that will be displayed in the new subwindow. There is always just one subpanel displayed in a subwindow.

It's Usage property defines when the subwindow will be opened: low-value or "t" means that the window will be opened as a child window as soon as the container window is opened; "p" means that a popup window (dialog box) will be opened on demand. Popup subwindows are closed automatically by the Enter and Escape keys, or the Close button.

The Uaage property can also be set to "n" which means that a subpanel is to created without creating a subwindow for it - this causes the container window to become a multi-panel window. To aid in editing this configuration, there is a new option on the editor View pulldown which allows other subpanels to be viewed while a subpanel is being edited.

Its Help key property defines the key code that will be used to open the subwindow if it is a popup window. This key code can relate to a Control key, a push button or icon key, or a menu accelerator key.

Its Protection property controls whether a subpanel will be given focus when tabbing out of a previous field in the container panel. If Protection is set to low-value, the subpanel will receive focus. This is the only situation where the location of the subwindow field is important.

Finer control over subpanels

Often there is no need for program interaction while focus is within a container panel and it's subpanels but if appropriate it is possible to return control to your program when a particular subpanel is activated or deactivated. The following options are available by setting the Options-5 property for the subpanel:

X'01' - return when subwindow is opened.
X'02' - return when subwindow is given focus.
X'04' - return when subwindow loses focus.
X'08' - return when subwindow is closed.

See Chapter C29 for more details on programming for containers.

Examples of container panel usage

It is appropriate to define any subsystem as a container panel and one or more subpanels. Subpanels can themselves be container panels so that you can define popup windows that are displayed on top of other popup windows, for example. See the newexamp sample panel file for an example of container panel usage.

As mentioned above, container panels are well suited for handing tabs - see the newtabs.pan sample panel file for an example of this. It is also possible to create custom tabs styles using text buttons - see the newtabs2 sample panel file for an example of this.

Container panels are also useful for handling frames and splitters. See the frames.pan sample panel file for an example of this.

Finally, container panels can be used to define repeat groups with multiple columns that scroll horizontally (grids) - see the horzscrl.pan sample panel file for an example of this.

CHAPTER B37 - Grids

Introduction to Grids

Grids allow the display of data in a table containing cells. A spread sheet is an example of a grid. There are 3 ways to create grids in sp2: manually define a repeat group to represent the grid; use an ActiveX control to handle the grid; use a specially marked grid panel. Repeat groups are covered in Chapter B15 and offer basic grid functionality. ActiveX controls are covered in Chapter B35 and offer comprehensive grid support depending on the control used but require the add-on ActiveX package and often have a lengthy learning curve. Grid panels offer somewhat of happy medium between the two: fairly simple implementation like repeat groups and some extended functionality like ActiveX controls - they are discussed here.
.

Grid Panels

To define a grid panel in the editor do the following:

1. Set Cell size to 1 by 1.
2. Set Grid panel to Yes.
3. Define first column header field, usually a push button or icon text button.
4. Define first column detail field, usually a system entry field.
5. Click anywhere in the panel to format the grid (you can do this at any time).
6. Repeat steps 3 through 5 to define additional columns.
7. Adjust the visible width/height of the grid by setting the panel Width/Height properties or using the View/Size menu option.
8. Adjust the total number of vertical occurrences in the grid by setting the Vertical occurs property for the generated repeat group.
9. The total width of the grid is automatically set to the total width of the columns plus 50 cells.
10. If the grid is to scroll, the panel Window border type property should be set to "main" for a self-contained grid or "none" for a grid to be included in a container panel. The border for a contained grid should be defined in the container panel using a no-member group.

When you test the panel, you will see that scroll bars have been added as necessary and splitter field have been inserted between column headings to allow resizing of columns.

Further options may be set in panel User data as follows:

dw -	divider width (default = 2)
dh -	divider height (default = 2)
hd -	headings y/n (default = y)
sz -	column resize allowed y/n (default = y)

For example, "dw=1,dh=1".

PART C

Programming with COBOL sp2

CHAPTER C1 - Introduction to Programming

This part of the User Guide explains how to program with COBOL sp2. Chapter C1 introduces various programming concepts. Chapters C2 and C3 discuss code that can be generated by the panel editor. Subsequent chapters discuss techniques for dealing with various situations. These techniques are illustrated by modifying the generated code.

Version 5 of COBOL sp2 introduces the concept of container panels which in many cases can cut down the amount of COBOL code needed to handle sp2 panels. See Chapter C29 for details on this.

All CALLs to the COBOL sp2 runtime are made in the following format:

```
CALL "SP2" USING function parameter.
```

The function specifies the type of operation you wish the runtime to perform. For example:

`OPEN-WINDOW` or **`DISPLAY-WINDOW`**

The parameter holds information which the runtime references during the operation. For example, the size and position of the window.

In some cases the parameter information may not be necessary, therefore a dummy parameter is passed. For example, **`DISPLAY-WINDOW`** always displays the currently active window.

In order to access the panel file in which your panels are stored, the following will be used:

`OPEN-FILE` to open a panel file so that your panel definitions can be accessed.

`CLOSE-FILE` to close the panel file.

If you wish to avoid having to distribute panel files, the following will be used instead:

`SET-RECORD` to establish a panel record in memory based on a generated panel definition in working-storage. Most of the subsequent chapters in Part C assume that panel files will be distributed and SET-RECORD will not be used. For more details on how to use it see Chapter C28.

In order to process a panel, the following will normally be used:

`CONVERSE-PANEL` to display a panel and accept input from it. This function will automatically open up an appropriate window to fit the panel.

`CLOSE-WINDOW` to close the window used to process the panel.

In order to shut down sp2 processing, the following will be used:

`END-SESSION` to terminate the COBOL sp2 session so that any memory can be freed.

COBOL sp2 allows you to generate a program containing all the above function calls to display and process a particular panel. Additionally a copy file is generated containing the parameter for the **`CONVERSE-PANEL`** function call. The next two chapters discuss the generated code.

To generate code from a panel, use the File/Generate menu option. Code is generated using a template file called "SP2.CBX". The template file is initially set up to produce the two files mentioned above. After you have generated one or two test programs, you may find the procedural portion of the generated code no longer to be necessary. This is because you will start to code your own call statements and to incorporate these call statements into more complex programs. The generated copy file will continue to be necessary, however, because this defines your program's access to panel fields.

To suppress generation of the test program, you should load the template file into a text editor and delete all lines up to the $000 statement for the CPY file.

Do not leave any blank lines in front of the $000 statement. If you wish the copy file to be written to a specific directory, you may modify the $000 statement by inserting a directory name between the $000 and the $001.

If you leave the test program generation intact, the generator will ask your permission before overwriting any existing program with the same name in case you have modified the generated code. This is a result of coding $029 instead of $000 for the file name.

CHAPTER C2 - The Generated Program Data Division

This chapter describes the items found in the Working-Storage section of the generated program.

Function Codes and Parameters

All Function Code and Parameter Definitions are contained in a COPY file:

```
COPY "SP2.CPY".
```

The following Function Codes are used in the generated code:

```
    05  SP2-OPEN-FILE           PIC S9(4) COMP-5 VALUE +1.
    05  SP2-CLOSE-WINDOW        PIC S9(4) COMP-5 VALUE +12.
    05  SP2-END-SESSION         PIC S9(4) COMP-5 VALUE +16.
    05  SP2-CONVERSE-PANEL      PIC S9(4) COMP-5 VALUE +19.
    05  SP2-CLOSE-FILE          PIC S9(4) COMP-5 VALUE +23.
```

Each function code is actually a binary numeric item but the function name should always be used to reference a function in your code.

Parameter for OPEN-FILE Function

```
*****************************
* parameter for OPEN-FILE   *
*****************************
 01  SP2-FILE-DEF.
    05  SP2-FI-RET-CODE         PIC S9(4) COMP-5.
    05  SP2-FI-LENS.
        10  SP2-FI-LEN-LEN      PIC S9(4) COMP-5 VALUE +10.
        10  SP2-FI-NUM-LEN      PIC S9(4) COMP-5 VALUE +0.
        10  SP2-FI-CHAR-LEN     PIC S9(4) COMP-5 VALUE +2.
        10  SP2-FI-VAR-LEN      PIC S9(4) COMP-5 VALUE +80.
        10  SP2-FI-NAME-LEN     PIC S9(4) COMP-5 VALUE +80.
    05  SP2-FI-DATA.
******** SP2-FI-CHAR-DATA *******
        10  SP2-FI-MODE         PIC X.
        10  FILLER              PIC X.
******** SP2-FI-VAR-DATA ********
        10  SP2-FI-NAME         PIC X(80).
```

This is the parameter used by the OPEN-FILE function. The only data item that needs to be set is the name of the file which is set in the procedural code.

Null Parameter

```
********************************
* parameter for DISPLAY-WINDOW *
* parameter for CLEAR-WINDOW   *
* parameter for CLOSE-WINDOW   *
* parameter for END-SESSION    *
* parameter for CLOSE-FILE     *
********************************
 01  SP2-NULL-PARM.
    05  SP2-NP-RET-CODE         PIC S9(4) COMP-5.
```

The null parameter is used for functions that do not need to be passed any data. In this program the CLOSE-FILE function uses it.

All the items described so far are included in the file, SP2.CPY. If you would prefer to use this copy file, please delete these items.

Parameter for CONVERSE-PANEL Function

```
 COPY "MAINMENU.CPY".
 *******************************
 * parameter for CONVERSE-PANEL *
 * parameter for GET-INPUT      *
 *******************************
  01  MAINMENU-CONVERSE-DATA.
      05  MAINMENU-RET-CODE
                                PIC S9(4) COMP-5.
      05  MAINMENU-LENS.
          10  MAINMENU-LEN-LEN
                                PIC S9(4) COMP-5 VALUE +20.
          10  MAINMENU-IP-NUM-LEN
                                PIC S9(4) COMP-5 VALUE +40.
          10  MAINMENU-IP-CHAR-LEN
                                PIC S9(4) COMP-5 VALUE +106.
          10  MAINMENU-OP-NUM-LEN
                                PIC S9(4) COMP-5 VALUE +6.
          10  MAINMENU-OP-CHAR-LEN
                                PIC S9(4) COMP-5 VALUE +2.
          10  MAINMENU-FIELD-LEN
                                PIC S9(4) COMP-5 VALUE +78.
          10  MAINMENU-COLR-LEN
                                PIC S9(4) COMP-5 VALUE +0012.
          10  MAINMENU-TYPE-LEN
                                PIC S9(4) COMP-5 VALUE +0012.
          10  MAINMENU-ACTION-LEN
                                PIC S9(4) COMP-5 VALUE +0.
          10  MAINMENU-SELECT-LEN
                                PIC S9(4) COMP-5 VALUE +0.
      05  MAINMENU-DATA.

******** MAINMENU-IP-NUM-DATA ********
          10  MAINMENU-KEY
                                PIC S9(4) COMP-5.
          10  MAINMENU-NEXT-FLD-ID
                                PIC S9(4) COMP-5.
          10  MAINMENU-NEXT-FLD-NUM
                                PIC S9(4) COMP-5.
          10  MAINMENU-NEXT-TAB-NUM
                                PIC S9(4) COMP-5.
          10  MAINMENU-NEXT-OCCURS
                                PIC S9(4) COMP-5.
          10  MAINMENU-LAST-FLD-ID
                                PIC S9(4) COMP-5.
          10  MAINMENU-LAST-FLD-NUM
                                PIC S9(4) COMP-5.
          10  MAINMENU-LAST-TAB-NUM
                                PIC S9(4) COMP-5.
          10  MAINMENU-LAST-OCCURS
                                PIC S9(4) COMP-5.
          10  MAINMENU-MENU-ID
                                PIC S9(4) COMP-5.
          10  MAINMENU-ROW-COL-SW
                                PIC S9(4) COMP-5.
          10  MAINMENU-CURSOR-ROW
```

```
                                PIC S9(4) COMP-5.
          10  MAINMENU-CURSOR-COL
                                PIC S9(4) COMP-5.
          10  MAINMENU-LAST-ROW
                                PIC S9(4) COMP-5.
          10  MAINMENU-LAST-COL
                                PIC S9(4) COMP-5.
          10  MAINMENU-DISP-SW
                                PIC S9(4) COMP-5.
          10  MAINMENU-NEXT-VERT
                                PIC S9(4) COMP-5.
          10  MAINMENU-LAST-VERT
                                PIC S9(4) COMP-5.
          10  MAINMENU-NEXT-HOR
                                PIC S9(4) COMP-5.
          10  MAINMENU-LAST-HOR
                                PIC S9(4) COMP-5.
******** MAINMENU-IP-NUM-DATA ********
          10  MAINMENU-NEXT-PANEL
                                PIC X(8).
          10  MAINMENU-NEXT-FIELD
                                PIC X(30).
          10  MAINMENU-LAST-FIELD
                                PIC X(30).
          10  MAINMENU-MENU-OPTION
                                PIC X(30).
          10  MAINMENU-SWITCH-SW
                                PIC X.
          10  MAINMENU-SIZE-SW
                                PIC X.
          10  MAINMENU-MOUSE-SW
                                PIC X.
          10  MAINMENU-CAPTURE-SW
                                PIC X.
          10  MAINMENU-WAIT-SW
                                PIC X.
          10  MAINMENU-CURS-SW
                                PIC X.
          10  MAINMENU-CHG-SW
                                PIC X.
          10  MAINMENU-TIMEOUT
                                PIC X.
******** MAINMENU-OP-NUM-DATA ********
          10  MAINMENU-PAN-POS-SW
                                PIC S9(4) COMP-5.
          10  MAINMENU-PAN-ROW
                                PIC S9(4) COMP-5.
          10  MAINMENU-PAN-COL
                                PIC S9(4) COMP-5.
******** MAINMENU-OP-CHAR-DATA ********
          10  MAINMENU-NEW-WINDOW
                                PIC X.
          10  MAINMENU-CUR-COLR-SW
                                PIC X.
******** MAINMENU-OP-VAR-DATA ********
      05  MAINMENU-FIELDS.
          10  MAINMENU-ACCOUNT-NUMBER
                                PIC 9(006).
          10  MAINMENU-CUSTOMER-NAME
                                PIC X(025).
          10  MAINMENU-ADDRESS
                                    PIC X(025).
          10  MAINMENU-CITY
                                    PIC X(015).
          10  MAINMENU-STATE
                                    PIC X(002).
          10  MAINMENU-ZIP-CODE
                                    PIC 9(005).
      05  MAINMENU-COLRS.
          10  MAINMENU-ACCOUNT-NUMBER-C
                                    PIC X.
          10  MAINMENU-FIELD-ID-009-C
                                    PIC X.
          10  MAINMENU-CUSTOMER-NAME-C
                                    PIC X.
          10  MAINMENU-ADDRESS-C
                                    PIC X.
          10  MAINMENU-CITY-C
                                    PIC X.
          10  MAINMENU-STATE-C
                                    PIC X.
          10  MAINMENU-FIELD-ID-010-C
                                    PIC X.
          10  MAINMENU-ZIP-CODE-C
                                    PIC X.
          10  MAINMENU-FIELD-ID-011-C
                                    PIC X.
          10  MAINMENU-FIELD-ID-007-C
                                    PIC X.
          10  MAINMENU-FIELD-ID-008-C
                                    PIC X.

          10  MAINMENU-FIELD-ID-012-C
                                    PIC X.
      05  MAINMENU-TYPES.
          10  MAINMENU-ACCOUNT-NUMBER-T
                                    PIC X.
          10  MAINMENU-FIELD-ID-009-T
                                    PIC X.
          10  MAINMENU-CUSTOMER-NAME-T
                                    PIC X.
          10  MAINMENU-ADDRESS-T
                                    PIC X.
          10  MAINMENU-CITY-T
                                    PIC X.
          10  MAINMENU-STATE-T
                                    PIC X.
          10  MAINMENU-FIELD-ID-010-T
                                    PIC X.
          10  MAINMENU-ZIP-CODE-T
                                    PIC X.
          10  MAINMENU-FIELD-ID-011-T
                                    PIC X.
          10  MAINMENU-FIELD-ID-007-T
                                    PIC X.
          10  MAINMENU-FIELD-ID-008-T
                                    PIC X.
          10  MAINMENU-FIELD-ID-012-T
                                    PIC X.
***********************************************
```

```
* field ids - use for cursor positioning, etc. *
************************************************
  01  MAINMENU-IDS.
      05  MAINMENU-ACCOUNT-NUMBER-I
                                PIC S9(4) COMP-5 VALUE +001.
      05  MAINMENU-FIELD-ID-009-I
                                PIC S9(4) COMP-5 VALUE +009.
      05  MAINMENU-CUSTOMER-NAME-I
                                PIC S9(4) COMP-5 VALUE +002.
      05  MAINMENU-ADDRESS-I
                                PIC S9(4) COMP-5 VALUE +003.
      05  MAINMENU-CITY-I
                                PIC S9(4) COMP-5 VALUE +004.
      05  MAINMENU-STATE-I
                                PIC S9(4) COMP-5 VALUE +005.
      05  MAINMENU-FIELD-ID-010-I
                                PIC S9(4) COMP-5 VALUE +010.
      05  MAINMENU-ZIP-CODE-I
                                PIC S9(4) COMP-5 VALUE +006.
      05  MAINMENU-FIELD-ID-011-I
                                PIC S9(4) COMP-5 VALUE +011.
      05  MAINMENU-FIELD-ID-007-I
                                PIC S9(4) COMP-5 VALUE +007.
      05  MAINMENU-FIELD-ID-008-I
                                PIC S9(4) COMP-5 VALUE +008.
      05  MAINMENU-FIELD-ID-012-I
                                PIC S9(4) COMP-5 VALUE +012.
```

This parameter is actually contained in a separate copy file. The panel name precedes all the data items so that multiple panels can be processed in the same program. In this program, two of the data items are referenced: NEXT-PANEL is set to the name of the panel to be displayed and NEW-WINDOW is set so that a new window will be automatically created to match the size of the panel.

You should notice that some items are included in PANEL003-COLRS and PANEL003-TYPES but not included in PANEL003-FIELDS . This is because some of the fields in the panel do not need to be referenced for data purposes but may need to have their type changed. For example, a Push Button may need to be disabled during program execution.

If you wish to suppress generation of the panel name prefix, edit the SP2.CBX file to remove all "$001-" strings from the file, except on those lines containing an 01-level.

CHAPTER C3 - The Generated Program Procedure Division

This chapter describes the logic found in the Procedure Division of the generated program. The logic of this program is quite simple but illustrates the code that MUST be included in a program to process a panel.

Open the Panel File

```
PERFORM PROC-OPEN-FILE THRU PROC-OPEN-FILE-EXIT.
```

The first thing to be done is to open a panel file. Until this is done no panel may be accessed. Opening the panel file only needs to be done once at the start of your application. It does not need to be done in every program.

Initialize CONVERSE-DATA

```
MOVE LOW-VALUES TO MAINMENU-DATA.
MOVE "MAINMENU" TO MAINMENU-NEXT-PANEL.
MOVE "y" TO MAINMENU-NEW-WINDOW.
MOVE LOW-VALUES TO MAINMENU-FIELDS.
MOVE LOW-VALUES TO MAINMENU-COLRS.
MOVE LOW-VALUES TO MAINMENU-TYPES.
```

It is very important to initialize parameters properly. Length items at the start of a parameter definition must not be altered but the rest of the parameter should at minimum be set to LOW-VALUES. Do not try setting the whole parameter to LOW-VALUES or the lengths will be erased.

In this example, the panel name is set as well as the switch to cause a new window to be opened. This switch should only be set when a new window is to be created. If a panel is to be displayed again, the switch should be set back to LOW-VALUE.

Process the Panel

```
PERFORM PROC-CON-MAINMENU THRU PROC-CON-MAINMENU-EXIT.
```

The CONVERSE-PANEL function is used to display the panel and accept user input. Control will be returned to the program when the user hits a control key or clicks on a Push Button. No processing of user input is performed.

Close the Window

```
PERFORM PROC-CLOSE-WINDOW THRU PROC-CLOSE-WINDOW-EXIT.
```

Although a window was automatically created by the CONVERSE-PANEL function the window will not be closed automatically.

Close the Panel File

```
PERFORM PROC-CLOSE-FILE THRU PROC-CLOSE-FILE-EXIT.
```

Closing the panel file only needs to be done once. The panel file should be closed immediately before terminating your application.

End the Current Session

```
PERFORM PROC-END-SESSION THRU PROC-END-SESSION-EXIT.
STOP RUN.
```

This function call is very important. If it is not done, system resources will not be released. This could result in eventually running out of memory which may cause other applications to fail. End the current session immediately before terminating your application.

Open File Paragraph

```
     PROC-OPEN-FILE.
    *****************
    * OPEN SP2 FILE *
    *****************
         MOVE LOW-VALUES TO SP2-FI-DATA.
         MOVE "user.pan" TO SP2-FI-NAME.
         CALL "SP2" USING SP2-OPEN-FILE SP2-FILE-DEF.
     PROC-OPEN-FILE-EXIT.
         EXIT.
```

As with all parameters, the data items in the parameter should all be initialized. Once the data items are initialized, only the name of the panel file needs to be set.

Converse Paragraph

```
  PROC-CON-MAINMENU.
 *****************
 * CONVERSE PANEL *
 *****************
      CALL "SP2" USING SP2-CONVERSE-PANEL MAINMENU-CONVERSE-DATA.
      MOVE LOW-VALUE TO MAINMENU-NEW-WINDOW.
  PROC-CON-MAINMENU-EXIT.
      EXIT.
```

Initialization for the CONVERSE-PANEL function was done in the mainline logic because a paragraph like this would normally be used more than once. The NEW-WINDOW switch is reset here to ensure that the window is created only once.

Miscellaneous Function Paragraphs

```
  PROC-CLOSE-WINDOW.
 ************************
 * CLOSE CURRENT WINDOW *
 ************************
      CALL "SP2" USING SP2-CLOSE-WINDOW SP2-NULL-PARM.
  PROC-CLOSE-WINDOW-EXIT.
      EXIT.

  PROC-CLOSE-FILE.
 **********************
 * CLOSE CURRENT FILE *
 **********************
      CALL "SP2" USING SP2-CLOSE-FILE SP2-NULL-PARM.
  PROC-CLOSE-FILE-EXIT.
      EXIT.

  PROC-END-SESSION.
 *******************
 * END SP2 SESSION *
 *******************
      CALL "SP2" USING SP2-END-SESSION SP2-NULL-PARM.
  PROC-END-SESSION-EXIT.
      EXIT.
```

These functions all use NULL-PARM so no initialization is necessary.

CHAPTER C4 - Responding to a Control Key

Control can be returned to your program for a variety of reasons. One of the most common reasons is that the user presses a control key or clicks on a Push Button which sends back a control key. The control key used will be placed in MAINMENU-KEY in MAINMENU-CONVERSE-DATA. Default control keys for a new panel are (ENTER) and (ESC).

The MAINMENU panel has the F4 and F5 keys as control keys as a result of the Add and Delete Customer Push Buttons. Control will be returned if F4 or F5 is pressed or if one of the Push Buttons is selected. As far as your program is concerned, the user will have pressed one of the keys. There is no need to worry about the Push Buttons.

In this example, we will exit the program if the (ESC) key is pressed but continue processing if any other key is pressed:

```
PERFORM PROC-CON-MAINMENU THRU PROC-CON-MAINMENU-EXIT
    UNTIL MAINMENU-KEY = SP2-KEY-ESCAPE.
```

The definition of SP2-KEY-ESCAPE is included in SP2.CPY.

```
05  SP2-KEY-ESCAPE            PIC S9(4) COMP-5 VALUE 27.
```

This code will reprocess the panel until the user presses the (ESC) key. The panel will be processed in the same window that was created on the initial call because the paragraph PROC-CON-MAINMENU contains the statement:

```
MOVE LOW-VALUE TO MAINMENU-NEW-WINDOW.
```

which suppresses the creation of a new window. It is very important that this switch is set properly otherwise you may create unwanted windows without realizing it.

CHAPTER C5 - Responding to a Menu Selection

If there is a menu for the current window and the user makes a selection from the menu, control will be returned to your program.

A special value is placed in MAINMENU-KEY in MAINMENU-CONVERSE-DATA to indicate that a menu selection has been made and the actual menu option is placed in MAINMENU-MENU-ID. The menu option is identified by the menu id entered in the panel editor.

The following code will cause the program to terminate if the exit option is selected:

```
PERFORM PROC-CON-MAINMENU THRU PROC-CON-MAINMENU-EXIT
    UNTIL MAINMENU-KEY = SP2-KEY-ESCAPE
       OR (MAINMENU-KEY = SP2-KEY-MENU
       AND MAINMENU-MENU-ID = MAINMENU-OPT-EXIT).
```

The special menu key value is defined in the SP2.CPY file:

```
05  SP2-KEY-MENU            PIC S9(4) COMP-5 VALUE -6.
```

The id for the menu exit option needs to be setup:

```
01  MAINMENU-MENU-OPTIONS.
    05  MAINMENU-OPT-EXIT       PIC S9(4) COMP-5 VALUE 23.
```

CHAPTER C6 - Processing User Data

Data to be displayed in the panel and entered by the user into the panel is held in the panel field data area. If this area is initially set to low-values the initial field values defined in the panel editor will be displayed. If spaces are moved to the area, all fields will be displayed as blank regardless of the initial field values set up. Thus the following generated statement is particularly important:

```
      MOVE LOW-VALUES TO MAINMENU-FIELDS.
```

Do not assume that your compiler will automatically set the area to low-values.

When control is returned to your program, the field area will contain either the original initial values or any new data entered by the user. At this point you may need to read a file to validate data entered or retrieve additional data based on a key entered and then display the panel again.

Modify the program to perform a new paragraph to display the panel and manipulate the field data:

```
   PERFORM CONVERSE-MAINMENU THRU CONVERSE-MAINMENU-EXIT
       UNTIL MAINMENU-KEY = SP2-KEY-ESCAPE
          OR (MAINMENU-KEY = SP2-KEY-MENU
          AND MAINMENU-MENU-ID = MAINMENU-OPT-EXIT).
```

Within the CONVERSE-MAINMENU paragraph we will read a record based on the ACCOUNT-NUMBER entered and then move the data from the record back into the field area:

```
 CONVERSE-MAINMENU.
*********************************************
* DISPLAY MAINMENU AND PROCESS DATA ENTERED *
*********************************************
     PERFORM PROC-CON-MAINMENU THRU PROC-CON-MAINMENU-EXIT.
     IF MAINMENU-KEY = SP2-KEY-ESCAPE
     OR (MAINMENU-KEY = SP2-KEY-MENU
     AND MAINMENU-MENU-ID = MAINMENU-OPT-EXIT)
         GO TO CONVERSE-MAINMENU-EXIT.
     MOVE MAINMENU-ACCOUNT-NUMBER TO RECORD-KEY.
     PERFORM READ-RECORD THRU READ-RECORD-EXIT.
     MOVE RECORD-CUSTOMER-NAME TO MAINMENU-CUSTOMER-NAME.
     MOVE RECORD-ADDRESS TO MAINMENU-ADDRESS.
     MOVE RECORD-CITY TO MAINMENU-CITY.
     MOVE RECORD-STATE TO MAINMENU-STATE.
     MOVE RECORD-ZIP-CODE TO MAINMENU-ZIP-CODE.
 CONVERSE-MAINMENU-EXIT.
     EXIT.
```

The READ-RECORD is not shown here because all file processing is dependent on your own installation and file access methods used.

Because this paragraph will be executed every time the user presses the (ENTER) key, you may wish to put in a conditional to prevent unnecessary file access:

```
   IF MAINMENU-ACCOUNT-NUMBER NOT = RECORD-KEY
       MOVE MAINMENU-ACCOUNT-NUMBER TO RECORD-KEY
       PERFORM READ-RECORD THRU READ-RECORD-EXIT
       MOVE RECORD-CUSTOMER-NAME TO MAINMENU-CUSTOMER-NAME
       MOVE RECORD-ADDRESS TO MAINMENU-ADDRESS
       MOVE RECORD-CITY TO MAINMENU-CITY
       MOVE RECORD-STATE TO MAINMENU-STATE
       MOVE RECORD-ZIP-CODE TO MAINMENU-ZIP-CODE.
```

CHAPTER C7 - Controlling Cursor Position

The easiest way to set and detect cursor position is to use the field ids which are listed in the generated code. For instance, to set the initial cursor position in the MAINMENU panel, use MAINMENU-NEXT-FLD-ID:

```
   MOVE MAINMENU-ZIP-CODE-I TO MAINMENU-NEXT-FLD-ID.
```

This overrides the default cursor position which is always the first field in the tabbing sequence.

To detect cursor position, always use MAINMENU-LAST-FLD-ID rather than MAINMENU-NEXT-FLD-ID, which may point to the field where the cursor will be placed next.

It is also possible to set and detect the offset of the cursor within a field by using ROW-COL-SW set to 2 and CURSOR-COL set to the character offset. This feature can only be used with custom fields and all the fields on the MAINMENU panel are system fields.

Other data items in the CONVERSE-DATA parameter that relate to cursor position are as follows (see chapter E2):

NEXT-FLD-NUM (only used if NEXT-FLD-ID is 0)
NEXT-TAB-NUM (only used if NEXT-FLD-ID and NEXT-FLD-NUM are 0)
NEXT-FIELD (only used if NEXT-FLD-ID, NEXT-FLD-NUM and NEXT-TAB-NUM are 0)
LAST-FLD-NUM
LAST-TAB-NUM
LAST-FIELD

CHAPTER C8 - Opening a Second Window

When an option is selected from a menu, it is often necessary to display a pop-up window or dialog box. This chapter discusses the code needed to do this.

Within the CONVERSE-MAINMENU paragraph, add a conditional to test if the Orders option has been selected on the menu:

```
IF (MAINMENU-KEY = SP2-KEY-MENU
AND MAINMENU-MENU-ID = MAINMENU-OPT-ORDERS)
    PERFORM CONVERSE-ORDER THRU CONVERSE-ORDER-EXIT.
```

This code should be inserted after the code to read a new record in the event that the user changes ACCOUNT-NUMBER before selecting the Orders option.

The new menu option must be defined in working-storage:

```
05  MAINMENU-OPT-ORDERS     PIC S9(4) COMP-5 VALUE 41.
```

The CONVERSE-ORDER paragraph will simply display the ORDER panel in a new window, wait for user input, then close the window:

```
 CONVERSE-ORDER.
***************************************
* DISPLAY AND PROCESS THE ORDER PANEL *
***************************************
     MOVE LOW-VALUES TO ORDER-DATA.
     MOVE "ORDER" TO ORDER-NEXT-PANEL.
     MOVE "y" TO ORDER-NEW-WINDOW.
     MOVE LOW-VALUES TO ORDER-FIELDS.
     MOVE LOW-VALUES TO ORDER-COLRS.
     MOVE LOW-VALUES TO ORDER-TYPES.
     PERFORM PROC-CON-ORDER THRU PROC-CON-ORDER-EXIT.
     PERFORM PROC-CLOSE-WINDOW THRU PROC-CLOSE-WINDOW-EXIT.
 CONVERSE-ORDER-EXIT.
     EXIT.
```

The code for this paragraph can actually be extracted from the test program generated by the panel editor. You will also need the generated copy file for the ORDER panel. Insert a copy statement in working storage after the copy statement for the MAINMENU panel:

```
 COPY "ORDER.CPY".
```

You should notice that the same CLOSE-WINDOW paragraph can be used to close the ORDER window as the MAINMENU window. This is because the CLOSE-WINDOW paragraph simply closes the current window regardless of the panel that window contains.

CHAPTER C9 - Responding to the user exiting a field

So far we have discussed how control is returned to your program as a result of a control key being pressed, a Push Button being selected or a menu option being selected. Sometimes it is useful to have control returned without any conscious user intervention. For example, when the user tabs out of a field, it may be necessary to immediately validate the data entered in the field. To do this you need to define the field as a "control field" by setting the Program control property for the field.

When the Program control property is set to "return on exit", control will normally be returned to your program as soon as the user tabs out of the field. You can also specify that control be returned only if the user modifies the field.

In this example, we will make the ACCOUNT-NUMBER field in the MAINMENU panel a control field so that address data may be updated as soon as the user types in a new number. If you wish to test the code, you should invoke the panel editor to set the Program control property for the ACCOUNT-NUMBER field to "return on exit."

No program changes are necessary because the CONVERSE-MAINMENU paragraph automatically reads the file when control is returned:

```
 CONVERSE-MAINMENU.
*********************************************
* DISPLAY MAINMENU AND PROCESS DATA ENTERED *
*********************************************
     PERFORM PROC-CON-MAINMENU THRU PROC-CON-MAINMENU-EXIT.
     IF MAINMENU-KEY = SP2-KEY-ESCAPE
     OR (MAINMENU-KEY = SP2-KEY-MENU
     AND MAINMENU-MENU-ID = MAINMENU-OPT-EXIT)
         GO TO CONVERSE-MAINMENU-EXIT.
     IF MAINMENU-ACCOUNT-NUMBER NOT = RECORD-KEY
         MOVE MAINMENU-ACCOUNT-NUMBER TO RECORD-KEY
         PERFORM READ-RECORD THRU READ-RECORD-EXIT
         MOVE RECORD-CUSTOMER-NAME TO MAINMENU-CUSTOMER-NAME
         MOVE RECORD-ADDRESS TO MAINMENU-ADDRESS
         MOVE RECORD-CITY TO MAINMENU-CITY
         MOVE RECORD-STATE TO MAINMENU-STATE
         MOVE RECORD-ZIP-CODE TO MAINMENU-ZIP-CODE.
 CONVERSE-MAINMENU-EXIT.
     EXIT.
```

In fact your program is specifically notified that a control field has been encountered by a special value in MAINMENU-KEY in CONVERSE-DATA. The actual key used to exit the field is placed in MAINMENU-MENU-ID.
The special value is defined in the SP2.CPY file:

```
     05  SP2-KEY-CTRL-FIELD      PIC S9(4) COMP-5 VALUE -4.
```

In this case there is only one control field in the panel so it does not matter which field caused control to be returned. This may not always be the case, however, so you may have to check the field. This is done using MAINMENU-LAST-FLD-ID in CONVERSE-DATA and the generated list of field ids. For example:

```
     IF MAINMENU-KEY = SP2-KEY-CTRL-FIELD
     AND MAINMENU-LAST-FLD-ID = MAINMENU-ACCOUNT-NUMBER-I
         PERFORM .........
```

In some cases a control field will cause control to be returned and no changes need be made to the panel display. In this case, this could happen if the user left the ACCOUNT-NUMBER field without modifying it (you could avoid this by making the field a modify-only control field). To speed up processing in these situations, you may wish to consider the use of the GET-INPUT function instead of CONVERSE-PANEL. You can use the same parameter for GET-INPUT so it is usually fairly easy to modify the logic to incorporate the new function. For example:

```
 CONVERSE-MAINMENU.
*********************************************
* DISPLAY MAINMENU AND PROCESS DATA ENTERED *
```

```
*********************************************
     IF FIELDS-NOT-MODIFIED = "Y"
         PERFORM PROC-GET-MAINMENU THRU PROC-GET-MAINMENU-EXIT.
     ELSE
         PERFORM PROC-CON-MAINMENU THRU PROC-CON-MAINMENU-EXIT.
     MOVE LOW-VALUE TO FIELDS-NOT-MODIFIED.
     IF MAINMENU-KEY = SP2-KEY-ESCAPE
     OR (MAINMENU-KEY = SP2-KEY-MENU
     AND MAINMENU-MENU-ID = MAINMENU-OPT-EXIT)
         GO TO CONVERSE-MAINMENU-EXIT.
     IF MAINMENU-ACCOUNT-NUMBER NOT = RECORD-KEY
         MOVE MAINMENU-ACCOUNT-NUMBER TO RECORD-KEY
         PERFORM READ-RECORD THRU READ-RECORD-EXIT
         MOVE RECORD-CUSTOMER-NAME TO MAINMENU-CUSTOMER-NAME
         MOVE RECORD-ADDRESS TO MAINMENU-ADDRESS
         MOVE RECORD-CITY TO MAINMENU-CITY
         MOVE RECORD-STATE TO MAINMENU-STATE
         MOVE RECORD-ZIP-CODE TO MAINMENU-ZIP-CODE
     ELSE
         MOVE "Y" TO FIELDS-NOT-MODIFIED.
 CONVERSE-MAINMENU-EXIT.
     EXIT.
```

The GET-INPUT paragraph is as follows:

```
  PROC-GET-MAINMENU.
************************
* GET INPUT FROM PANEL *
************************
     CALL "SP2" USING SP2-GET-INPUT MAINMENU-CONVERSE-DATA.
 PROC-GET-MAINMENU-EXIT.
     EXIT.
```

CHAPTER C10 - Prohibiting User Input

It is sometimes necessary to dynamically protect fields from user input. This is sometimes known as "greying out" fields. There are data items set up in CONVERSE-DATA to allow you to accomplish this very easily.

Suppose, for example, you wish to prohibit entry into the ADDRESS field of the MAINMENU panel. This is done using the following code:

```
    MOVE SP2-DISPLAY-ONLY TO MAINMENU-ADDRESS-T.
```

Insert this code prior to the CONVERSE-PANEL function call, but after initializing MAINMENU-TYPES. SP2-DISPLAY-ONLY is defined in the SP2.CPY file:

```
 01  SP2-FIELD-TYPES.
     05  SP2-DISPLAY-ONLY        PIC X VALUE "o".
```

If you wish to deactivate a Push Button or Icon, the code is slightly different because mouse input is accepted by Push Buttons and Icons even if they are designated as display-only. Use SP2-GREYED-OUT instead of SP2-DISPLAY-ONLY and no input will be accepted:

```
     05  SP2-GREYED-OUT          PIC X VALUE "g".
```

If you look at the generated MAINMENU-TYPES you will see that there does not seem to be a slot for any of the Push Buttons. This is because the Push Button fields were not given a name when they were defined in the panel editor. Remember to supply a name for Push Buttons if you need to grey them out. This will avoid code like this:

```
    MOVE SP2-GREYED-OUT TO MAINMENU-FIELD-ID-008-T.
```

This references the name supplied by the generator when no real name is
established.

It may also be necessary to disallow menu options. The technique for doing this is slightly different and a little more complicated.

CONVERSE-DATA does not reference the menu for a panel other than holding the menu option that was selected by the user. To modify a menu, a separate function call must be made. This function is SET-MENU-OPTION which uses the MENU-OPTION parameter with a key of MO-ID (the option id entered in the panel editor). This function must be made prior to the first CONVERSE-PANEL call. This raises a new difficulty.

Before the first CONVERSE-PANEL call there is no window open and therefore no menu to modify. The solution is to use the OPEN-WINDOW function to open the window beforehand. Insert the following code immediately after the OPEN-FILE call:

```
MOVE LOW-VALUES TO SP2-WD-DATA.
   MOVE "MAINMENU" TO SP2-WD-PANEL-NAME.
   PERFORM PROC-OPEN-WINDOW THRU PROC-OPEN-WINDOW-EXIT.
```

The WINDOW-DEF parameter is in the SP2.CPY file. See chapter E3 for a detailed description of it.
Because we are supplying a panel name, all the data items for the window will be derived from the panel definition. The actual call paragraph is:

```
 PROC-OPEN-WINDOW.
*********************
* OPEN A NEW WINDOW *
*********************
     CALL "SP2" USING SP2-OPEN-WINDOW SP2-WINDOW-DEF.
 PROC-OPEN-WINDOW-EXIT.
     EXIT.
```

Because we are opening the window manually, we do not require the CONVERSE-PANEL function to open a window. Remove the following statement in the CONVERSE-DATA initialization code:

```
   MOVE "y" TO MAINMENU-NEW-WINDOW.
```

Once the window has been opened, the menu will have been established and any of the menu options may be modified. The SET-MENU-OPTION function should be used in conjunction with the GET-MENU-OPTION so that unintended modifications to a menu option are not made. Insert the following code after the OPEN-WINDOW call:

```
   MOVE LOW-VALUES TO SP2-MO-DATA.
   MOVE MAINMENU-OPT-ORDERS TO SP2-MO-ID.
   PERFORM PROC-GET-MENU-OPTION THRU PROC-GET-MENU-OPTION-EXIT.
   MOVE SP2-GREYED-OUT TO SP2-MO-STATE.
   PERFORM PROC-SET-MENU-OPTION THRU PROC-SET-MENU-OPTION-EXIT.
```

The definition for MENU-OPTION is in the SP2.CPY file. See chapter E12 for a detailed description of it.

CHAPTER C11 - Modifying Field Colors

Dynamic modification of field colors is probably less important in contemporary user interfaces than it has been in the past. Nevertheless, facilities are provided to do this.

Suppose you wish to modify the color of the ACCOUNT-NUMBER field in the MAINMENU panel so that it is displayed with the default highlight color. Insert the following code before the first CONVERSE-PANEL call but after the initialization of MAINMENU-COLRS:

```
MOVE SP2-COLR-HIGHLIGHT TO MAINMENU-ACCOUNT-NUMBER-C.
```

The following default colors are defined in the SP2.CPY file:

```
01  SP2-COLRS.
    05  SP2-COLR-TEXT           PIC X VALUE X"01".
    05  SP2-COLR-HIGHLIGHT      PIC X VALUE X"02".
    05  SP2-COLR-MENU           PIC X VALUE X"03".
```

These are the colors which are always available. Other colors have to be set up either in your program (SET-COLOR-DEF function) or in the panel editor (Color definition dialog box) and have an id greater than X"0A".

To modify the color of a system field at runtime, set the field More Options X'04' switch.

CHAPTER C12 - Processing Repeat Groups

Your program's view of a repeat group is that of a table containing all the occurrences of that group, regardless of how many occurrences are actually displayed to the user. This makes processing a repeat group very easy - your program does not have to worry about the complexities of scrolling the data. Simply load up the table in the generated field area prior to displaying your panel and then extract data from the table upon return to your program.

The CONVERSE-DATA area contains most of the data items needed to control a repeat group. These are as follows:

xxxxxxxx-NEXT-FLD-ID is the id of the field where the cursor is to be placed. In the case of a repeat group, you should always use the id of the base field.

xxxxxxxx-NEXT-OCCURS is the occurrence of the field where the cursor is to be placed. If this occurrence is not currently visible, the repeat group will automatically be scrolled so that it is visible.

xxxxxxxx-LAST-FLD-ID is the id of the field where the cursor was positioned prior to control being returned. In the case of a repeat group, this is always the id of the base field.

xxxxxxxx-LAST-OCCURS is the occurrence of the field where the cursor is to be placed.

xxxxxxxx-DISP-SW should be set to 1 if you wish to alter the portion of the repeat group being displayed.

xxxxxxxx-NEXT-VERT is the displacement in terms of rows of the first row to be displayed. This is only used if xxxxxxxx-DISP-SW is 1. This will be overridden if it is necessary to scroll the repeat group to display the row where the cursor is positioned.

xxxxxxxx-LAST-VERT is the displacement in terms of rows of the first row displayed prior to control being returned.

For example, the following code detects which occurrence the cursor is on when control is returned from the ORDER panel:

```
 CONVERSE-ORDER.
****************************************
* DISPLAY AND PROCESS THE ORDER PANEL *
****************************************
     MOVE LOW-VALUES TO ORDER-DATA.
     MOVE "ORDER" TO ORDER-NEXT-PANEL.
     MOVE "y" TO ORDER-NEW-WINDOW.
     MOVE LOW-VALUES TO ORDER-FIELDS.
     MOVE LOW-VALUES TO ORDER-COLRS.
     MOVE LOW-VALUES TO ORDER-TYPES.
     PERFORM PROC-CON-ORDER THRU PROC-CON-ORDER-EXIT.
     IF ORDER-LAST-FLD-ID = ORDER-PRODUCE-I OR
                            ORDER-QUANTITY-I OR
                            ORDER-PRICE-I
         MOVE ORDER-LAST-OCCURS TO WS-SUB-1
         PERFORM PROCESS-ITEM THRU PROCESS-ITEM-EXIT.
     PERFORM PROC-CLOSE-WINDOW THRU PROC-CLOSE-WINDOW-EXIT.
 CONVERSE-ORDER-EXIT.
     EXIT.
```

Here ORDER-LAST-OCCURS is used as a subscript into the produce table. Your program does not need to be concerned with which portion of the table is actually displayed.

One of the data items that is not included in CONVERSE-DATA is the number of occurrences within the repeat group. You may need to reset this item based on the number of records read from a file, for instance. To do this, use the following code sequence:

```
     MOVE LOW-VALUES TO SP2-RX-DATA.
     MOVE 1 TO SP2-RX-ID.
     MOVE 1 TO SP2-RX-TOTAL-SW.
     MOVE WS-RECORD-COUNT TO SP2-RX-TOTAL-OCCS.
     CALL "SP2" USING SP2-SET-REPEAT-EXT SP2-REPEAT-EXT.
```

Remember that if you want to do this prior to the first display of the panel, you must use the OPEN-WINDOW function rather than just a CONVERSE-PANEL function call with the NEW-WINDOW switch set. See chapter C10 for more on this.

In this example, RX-ID is set to 1. This will normally be correct unless you have more than one repeat group in your panel. In this case, ids are allocated in the order in which the repeat groups are defined.

The SET-REPEAT-EXT function can also be used to reset the visible portion of the repeat group:

```
MOVE LOW-VALUES TO SP2-RX-DATA.
MOVE 1 TO SP2-RX-ID.
MOVE 1 TO SP2-RX-DISP-SW.
COMPUTE SP2-RX-NEW-DISP = WS-RECORD-TO-BE-DISPLAYED - 1.
CALL "SP2" USING SP2-SET-REPEAT-EXT SP2-REPEAT-EXT.
```

You must use this method rather than merely setting xxxxxxxx-DISP-SW and xxxxxxxx-NEXT-VERT if the cursor is not currently within the repeat group.

CHAPTER C13 - Controlling Radio Buttons and Check Boxes

As long as they are defined properly, these types of fields can be treated as though they are standard data entry fields. This is because the runtime will return an appropriate value to the panel field area rather than indicate, for example, that the user has clicked on one of these fields.

For example, the Radio Buttons in the ORDER panel send the results of all user input back to the data item ORDER-CHECK in ORDER-FIELDS. This is actually the name of the first Radio Button in the group. The value is one of those set up for each of the buttons. If you had a field representing payment method in your file, you could simply move the value to the record field as long as the values have been defined properly:

```
MOVE ORDER-CHECK TO RECORD-PAYMENT-METHOD.
```

Check Boxes are generated so that they each have their own slot in the panel field area. Two values are setup. The first value is put in the appropriate slot if the Check Box is turned on. The second value is put in the appropriate slot if it is turned off. For example:

```
IF ORDER-INSURE = "Y"
    PERFORM INSURE-ORDER THRU INSURE-ORDER-EXIT.
```

CHAPTER C14 - Controlling List Boxes and Combination Boxes

In program terms, these two types of fields work identically. Both field types return to the program field area the selected value. The only difference between the two is that, in the case of the combination box, the value may have been typed in rather than just selected. This is irrelevant to your program so you can treat both types of fields as if they are normal single data entry fields. There are two complications with respect to these types of field. The first relates to changing the list of items displayed which is normally set up using the Properties Box in the editor. The second relates to multi-select Combination Boxes.

The list of items is assigned to the Value property of the field when it is defined either in your program or in the panel editor. To change the list you have to reset this property using the SET-FIELD-DEF function or the SET-PROPERTY function. This is the first time that these functions have been mentioned. They are functions that allow you to dynamically define panels and set object properties at runtime. Here it is necessary to reset the Value property of a Combination Box field.

Remember that if you want to reset the Combination Box prior to the first display of the panel, you must use the OPEN-WINDOW function rather than just a CONVERSE-PANEL function call with the NEW-WINDOW switch set. See chapter C10 for more on this.

The code needed to update the Value property of a field using the SET-FIELD-DEF function has been packaged up in a copy file called "SETVAL.CPY". The following could be used to reload a listbox with new data:

```
COPY "SETVALWS.CPY".

    MOVE WS-NEW-ITEMS TO SETVAL-DATA.
    COMPUTE SETVAL-LEN = WS-NEW-COUNT * WS-ITEM-LEN.
    MOVE LISTBOX-I TO SETVAL-ID.
    PERFORM SETVAL.

COPY "SETVAL.CPY".
```

SETVAL.CPY contains a single paragraph, so remember to precede the copy statement by a section name if you use sections in your program.

If you define a List Box to be "multi-select" (field Usage property is "m" or "n"), you need to write special code to retrieve the selections made because there is room for only one selection in the program Fields area. Use the following code:

```
    MOVE 2000 TO SP2-FD-VAR-LEN SP2-FD-INITIAL-LEN.
    MOVE LOW-VALUES TO SP2-FD-DATA.
    MOVE field-id TO SP2-FD-ID.
    CALL "SP2" USING SP2-GET-FIELD-DATA SP2-FIELD-DEF.
    MOVE SP2-FD-VAR-DATA (1 : 10) TO WS-SELECTION-1.
    MOVE SP2-FD-VAR-DATA (11 : 10) TO WS-SELECTION-2.
```

It may also be necessary to programmatically select items in a multi-select List Box. Use the following code to do this:

```
    MOVE LOW-VALUES TO SP2-FD-VAR-LENS
    COMPUTE SP2-FD-INITIAL-LEN = SP2-FD-ITEM-LEN * NUMBER-OF-SELECTIONS
    MOVE LOW-VALUES TO SP2-FD-DATA
    MOVE FIELD-ID TO SP2-FD-ID
    MOVE 1 TO WS-SUB
    MOVE SELECTION-1 TO SP2-FD-VAR-DATA (WS-SUB : SP2-FD-ITEM-LEN)
    ADD SP2-FD-ITEM-LEN TO WS-SUB
    MOVE SELECTION-2 TO SP2-FD-VAR-DATA (WS-SUB : SP2-FD-ITEM-LEN)
    CALL "SP2" USING SP2-DISPLAY-FIELD SP2-FIELD-DEF
```

To select all, set selection-1 to low-values and selection-2 to high-values.

To clear all selections, set selection-1 to high-values and selection-2 to low-values.

CHAPTER C15 - Controlling Scroll Bars

Scroll Bars included in repeat groups are essentially invisible to your program. Scroll Bars simply control the scrolling of data within the groups. The only thing you might need to do at runtime is to grey out a Scroll Bar. This is unfortunately a little more complicated than graying out a regular field because there is no entry in the
Program Types area for a Scroll Bar. Instead, use the following special code:

```
MOVE LOW-VALUES TO SP2-FD-DATA
MOVE SCROLL-BAR-I TO SP2-FD-ID
CALL "SP2" USING SP2-GET-FIELD-DEF SP2-FIELD-DEF
MOVE SP2-FD-REPEAT-ID TO WS-SAVE-ID
MOVE 0 TO SP2-FD-REPEAT-ID
MOVE "g" TO SP2-FD-OUTPUT
CALL "SP2" USING SP2-SET-FIELD-DEF SP2-FIELD-DEF
```

To restore the Scroll Bar, use the following:

```
MOVE LOW-VALUES TO SP2-FD-DATA
MOVE SCROLL-BAR-I TO SP2-FD-ID
CALL "SP2" USING SP2-GET-FIELD-DEF SP2-FIELD-DEF
MOVE WS-SAVE-ID TO SP2-FD-REPEAT-ID
MOVE "y" TO SP2-FD-OUTPUT
CALL "SP2" USING SP2-DELETE-FIELD SP2-FIELD-DEF
CALL "SP2" USING SP2-SET-FIELD-DEF SP2-FIELD-DEF
```

Scroll Bars can also be used as stand alone fields, to allow user input based on a sliding scale.

This sort of input is translated for your program into a resultant value, and therefore can be handled by your program as if the value had been entered in a regular data entry field. The value ranges from 0 to 100 based on the position of the Scroll Bar slider and the value is reformatted based on however you set the field Format property.

Likewise, placing a value into the field area for a Scroll Bar will cause the Scroll Bar slider to be positioned appropriately.

For example, if actual values for a particular data item range from 0 through 255, the following code could be used to derive this value from Scroll Bar input:

```
COMPUTE RECORD-VALUE = (SCROLLBAR-INPUT * 255) / 100.
```

CHAPTER C16 - Working with Windows

Chapter C8 discussed how to open a second window. Opening a second or pop-up window is particularly easy with COBOL sp2 because all windows are essentially pop-up windows. Once a window has been closed, the underlying window is automatically restored and your program can simply loop round to the original CONVERSE-PANEL call.

Chapter C9 introduced the concept of opening a window prior to the first CONVERSE-PANEL call using the OPEN-WINDOW function. This was done so that a menu option could be modified. The most important thing to recognize here is that a window must be open before you can really do anything apart from open and close files. For example, if you wanted to set the window title on the first display of a window, the first thing you must do is open the window:

```
    MOVE LOW-VALUES TO SP2-WD-DATA.
    MOVE "TESTPAN" TO SP2-WD-PANEL-NAME.
    CALL "SP2" USING SP2-OPEN-WINDOW SP2-WINDOW-DEF.
```

This code sequence opens a window based on the TESTPAN panel definition so that you do not have to worry about setting any of the window data items apart from the panel name. Now you can reset the title:

```
    MOVE "TESTPAN" TO SP2-WD-NAME.
    CALL "SP2" USING SP2-GET-WINDOW-DEF SP2-WINDOW-DEF.
    MOVE "New title" TO SP2-WD-TITLE.
    CALL "SP2" USING SP2-SET-WINDOW-DEF SP2-WINDOW-DEF.
```

And now you can resume normal processing:

```
    MOVE LOW-VALUES TO TESTPAN-DATA.
    MOVE "TESTPAN" TO TESTPAN-NEXT-PANEL.
    MOVE LOW-VALUES TO TESTPAN-FIELDS.
    MOVE LOW-VALUES TO TESTPAN-COLRS.
    MOVE LOW-VALUES TO TESTPAN-TYPES.
    CALL "SP2" USING SP2-CONVERSE-PANEL TESTPAN-CONVERSE-DATA.
```

Notice here that the NEW-WINDOW switch is NOT set. This is because the window has already been opened using the OPEN-WINDOW function.

What if we extend the example in chapter C8 and open three windows one after another either by using CONVERSE-PANEL with the NEW-WINDOW switch or by using OPEN-WINDOW and then CONVERSE-PANEL? After doing this, what happens if the third window is closed? There are two windows still open - which one becomes the active window? The answer is that the second window opened now becomes the active window. This is because as windows are opened there is a natural stacking order developed. As windows are closed, the previous window on the stack becomes active.

Normally, windows are usually processed using this natural stacking order in mind. Users normally will expect that windows should be opened and closed and reactivated in this natural order. However, this does not have to be so. For example, after closing the third window, it is quite possible to reactivate the first window rather than the second window. This is done using the ACTIVATE-WINDOW function, as follows:

```
* display PANEL001
    MOVE LOW-VALUES TO PANEL001-DATA.
    MOVE "PANEL001" TO PANEL001-NEXT-PANEL.
    MOVE "y" TO PANEL001-NEW-WINDOW.
    MOVE LOW-VALUES TO PANEL001-FIELDS.
    MOVE LOW-VALUES TO PANEL001-COLRS.
    MOVE LOW-VALUES TO PANEL001-TYPES.
    CALL "SP2" USING SP2-CONVERSE-PANEL PANEL001-CONVERSE-DATA.
* display PANEL002
    MOVE LOW-VALUES TO PANEL002-DATA.
    MOVE "PANEL002" TO PANEL002-NEXT-PANEL.
    MOVE "y" TO PANEL002-NEW-WINDOW.
    MOVE LOW-VALUES TO PANEL002-FIELDS.
    MOVE LOW-VALUES TO PANEL002-COLRS.
```

```
           MOVE LOW-VALUES TO PANEL002-TYPES.
           CALL "SP2" USING SP2-CONVERSE-PANEL PANEL002-CONVERSE-DATA.
      * display PANEL003
           MOVE LOW-VALUES TO PANEL003-DATA.
           MOVE "PANEL003" TO PANEL003-NEXT-PANEL.
           MOVE "y" TO PANEL003-NEW-WINDOW.
           MOVE LOW-VALUES TO PANEL003-FIELDS.
           MOVE LOW-VALUES TO PANEL003-COLRS.
           MOVE LOW-VALUES TO PANEL003-TYPES.
           CALL "SP2" USING SP2-CONVERSE-PANEL PANEL003-CONVERSE-DATA.
      * close PANEL003
           CALL "SP2" USING SP2-CLOSE-WINDOW SP2-NULL-PARM.
      * activate PANEL001 instead of processing panel002
           MOVE "PANEL001" TO SP2-ND-NAME.
           CALL "SP2" USING SP2-ACTIVATE-WINDOW SP2-NAME-DEF.
      * process PANEL001
           CALL "SP2" USING SP2-CONVERSE-PANEL PANEL001-CONVERSE-DATA.
      * close PANEL001
           CALL "SP2" USING SP2-CLOSE-WINDOW SP2-NULL-PARM.
      * process PANEL002
           CALL "SP2" USING SP2-CONVERSE-PANEL PANEL002-CONVERSE-DATA.
      * close PANEL002
           CALL "SP2" USING SP2-CLOSE-WINDOW SP2-NULL-PARM.
```

You should notice that the name of the window used in the ACTIVATE-WINDOW call is the same as that of the panel. This is normally the case unless you specifically assign a different window name in an OPEN-WINDOW call. After PANEL001 has been closed there is only one window left open so there is no need for another ACTIVATE-WINDOW call. If there was more than one window left open, it would be advisable to use an ACTIVATE-WINDOW because a stacking order should not be assumed once ACTIVATE-WINDOW has been used.

It should now be clear how your program can cause control to switch to any open window. What if you want this switching to be under the control of the user? This introduces the topic of modal versus modeless windows.

A modal window might be described as a normal window - one that has to be closed before the user can switch to a previous window. A modeless window is one that can be left open even though the user switches to a previous window. All windows discussed so far have been modal ones - modeless ones are a little more complicated.

You can effectively make a window modeless just by not closing it when control is returned to your program, and then you can use the ACTIVATE-WINDOW call to activate another window that has already been opened. The normal way, however, for the user to switch between windows is to use the mouse to merely click on the window that he or she wishes to make active. This is a problem because, by default, COBOL sp2 does not allow other windows to be activated with the mouse. This is so because your program may have a problem if it is processing one window and the user suddenly switches to another one.

To allow your user to switch to another window using the mouse, you must set the SWITCH-SW to "y" in a panel's converse-data area prior to making a CONVERSE-PANEL call. Once you do this, it is up to your program to be able to transfer control in its logic to the code that processes the new window. -3 will be returned in the KEY field of the current panel's converse data area and the name of the window that has been made active will be returned in the NEXT-PANEL field.

To be able to process these user switches without using GOTO's all over your program, you should code something like the following:

```
           MOVE "PANEL001" TO SP2-ND-NAME.
           CALL "SP2" USING SP2-ACTIVATE-WINDOW SP2-NAME-DEF.
           MOVE "PANEL001" TO WS-NEXT-PANEL.
           PERFORM CONVERSE-ALL UNTIL .....

       CONVERSE-ALL.
           IF WS-NEXT-PANEL = "PANEL001"
               PERFORM CONVERSE-PANEL001
           ELSE
```

```
         IF WS-NEXT-PANEL = "PANEL002"
             PERFORM CONVERSE-PANEL002
         ELSE
         IF WS-NEXT-PANEL = "PANEL003"
             PERFORM CONVERSE-PANEL003
     CONVERSE-PANEL001.
         MOVE "PANEL001" TO PANEL001-NEXT-PANEL.
         CALL "SP2" USING SP2-CONVERSE-PANEL PANEL001-CONVERSE-DATA.
         MOVE PANEL001-NEXT-PANEL TO WS-NEXT-PANEL.

     CONVERSE-PANEL002.
         MOVE "PANEL002" TO PANEL002-NEXT-PANEL.
         CALL "SP2" USING SP2-CONVERSE-PANEL PANEL002-CONVERSE-DATA.
         MOVE PANEL002-NEXT-PANEL TO WS-NEXT-PANEL.

     CONVERSE-PANEL003.
         MOVE "PANEL003" TO PANEL003-NEXT-PANEL.
         CALL "SP2" USING SP2-CONVERSE-PANEL PANEL003-CONVERSE-DATA.
         MOVE PANEL003-NEXT-PANEL TO WS-NEXT-PANEL.
```

The trick is to use a controlling paragraph (in this case CONVERSE-ALL) to switch between the different windows. Notice that no ACTIVATE-WINDOW CALLs are needed (once the user is in control) because windows are automatically activated by the user's mouse click. You would only need an ACTIVATE-WINDOW call if your program needed to override the user's choice of window, but if you are going to give the user this switching capability, you should really honor the user's choice.

CHAPTER C17 - Getting and setting properties

In the previous chapter, a code example was given for resetting the window title:

```
MOVE LOW-VALUES TO SP2-WD-DATA.
MOVE "TESTPAN" TO SP2-WD-NAME.
CALL "SP2" USING SP2-GET-WINDOW-DEF SP2-WINDOW-DEF.
MOVE "New title" TO SP2-WD-TITLE.
CALL "SP2" USING SP2-SET-WINDOW-DEF SP2-WINDOW-DEF.
```

This was in fact an example of resetting an object property at runtime. In many cases, it is possible to code your user interface without ever accessing object properties directly. This is because there are many data items defined in the generated panel CONVERSE-DATA area which allow you to control a panel's behavior by accessing transient properties associated with the panel being conversed. Examples of these properties are the converse panel Next field id property (TESTPAN-NEXT-FLD-ID) and the Control key pressed property (TESTPAN-KEY) which are used to position the cursor and check the key pressed by the user. Converse panel properties are described in detail in chapter E2.

Sometimes, however, as with the example of resetting the window title above, it is necessary to set an object property directly because there is no converse panel property available. In this case, the object was the "TESTPAN" window (derived from the "TESTPAN" panel) and the property was the Title property. It is possible, however, to reset almost all of the properties of all the SP2 objects (panels, fields, etc.) These properties are described in detail in part E.

This chapter describes the function calls needed to modify properties. The first part describes the traditional method which involves retrieving the definition of an object, resetting a particular property or properties, then updating the object definition. The second part describes the newer method which involves updating a particular property directly. This newer method should be used in new code when only a single property needs to be retrieved or modified.

It is always important to remember that, in most cases, an object cannot be accessed until a window has been opened to contain the object. The following is the easiest way to open a window:

```
MOVE LOW-VALUES TO SP2-WD-DATA.
MOVE "TESTPAN" TO SP2-WD-PANEL-NAME.
CALL "SP2" USING SP2-OPEN-WINDOW SP2-WINDOW-DEF.
```

This code opens a window based on the "TESTPAN" panel and reads the panel definition into memory along with all the panel-related objects. You now have access to the window based on the "TESTPAN" panel, the "TESTPAN" panel itself, and all the objects within the panel – menu, fields, static fields, groups, repeat groups.

Accessing Properties Using Object Definitions

As with resetting the window Title property, the technique used for resetting a property is to call a function to get the object and its properties, reset the data item related to the appropriate property, then call a function to reset the object. Here is a list of the objects and the functions used to access them:

Window -	GET/SET-WINDOW-DEF
Panel -	GET/SET-PANEL-DEF
Field -	GET/SET-FIELD-DEF
Static field -	GET/SET-STATIC-DEF
Group -	GET/SET-GROUP-DEF
Repeat group -	GET/SET-REPEAT-DEF
Menu -	GET/SET-MENU-DEF
Menu option -	GET/SET-MENU-OPTION
Font -	GET/SET-FONT-DEF
Color -	GET/SET-COLOR-DEF

To reset the text in a static field, for example, you need to reset the Text property of the static field:

```
MOVE LOW-VALUES TO SP2-SD-DATA.
MOVE 5 TO SP2-SD-ROW.
```

```
    MOVE 10 TO SP2-SD-COL.
    CALL "SP2" USING SP2-GET-STATIC-DEF SP2-STATIC-DEF.
    MOVE "New text" TO SP2-SD-TEXT.
    CALL "SP2" USING SP2-SET-STATIC-DEF SP2-STATIC-DEF.
```

Notice that you must set the data item associated with the key of the object that is being accessed, prior to the GET call. Here is a list of the keys associated with each object:

Window -	WD-NAME or WD-WINDOW-ID
Panel -	PD-NAME
Field -	FD-ID or FD-NAME
Static field -	SD-ROW/COL or SD-ID
Group -	GD-ID
Repeat group -	RD-ID
Menu -	MD-NAME
Menu option -	MDO-ID
Font -	FO-ID
Color -	CO-ID

Where alternate keys are listed, the alternate key may only be used if the primary key is set to low-values or zero. In the case of static fields, the alternate key (SD-ID) may only be used if the Id property for the static field is set when the static is created (in the editor it defaults to zero).

Accessing field properties is a special case because the format of the variable length field properties varies so much. The following code should be used:

```
    MOVE 2000 TO SP2-FD-VAR-LEN.
    MOVE LOW-VALUES TO SP2-FD-VAR-LENS.
    MOVE LOW-VALUES TO SP2-FD-DATA.
    MOVE ACCOUNT-NUMBER-I TO SP2-FD-ID.
    CALL "SP2" USING SP2-GET-FIELD-DEF SP2-FIELD-DEF.
    MOVE SP2-COLR-HIGHLIGHT TO SP2-FD-CUR-COLR.
    CALL "SP2" USING SP2-SET-FIELD-DEF SP2-FIELD-DEF.
```

2000 is the length of FD-VAR-LEN as defined in SP2.CPY. This length may need to be increased if the combined length of the variable field properties exceed this length. This might be the case for list box and combo box fields which store the items to be listed in their Value properties. See chapter C14 for details of this. Setting FD-VAR-LENS to low-values cause the variable-length property values to be returned as one concatenated string in FD-VAR-DATA rather than reformatted and split up as for panel variable-length properties in PD-VAR-DATA. To access each property individually, you need to refer to the variable lengths returned in FD-VAR-LENS after a GET-FIELD-DEF call and unstring FD-VAR-DATA. Code to do this is provided in the following copy files:

FD-FMT -	GETFMT.CPY, RESETFMT.CPY
FD-CAPTION -	GETCAP.CPY, RESETCAP.CPY
FD-INITIAL-VAL -	GETVAL.CPY, RESETVAL.CPY
FD-MSG-TEXT -	GETMSG.CPY, RESETMSG.CPY

To reset the caption of a radio button, for example, you would use the following code:

```
    COPY "SETVALWS.CPY"

    MOVE 2000 TO SP2-FD-VAR-LEN.
    MOVE LOW-VALUES TO SP2-FD-VAR-LENS.
    MOVE LOW-VALUES TO SP2-FD-DATA.
    MOVE RADIO-BUTTON-I TO SP2-FD-ID.
    CALL "SP2" USING SP2-GET-FIELD-DEF SP2-FIELD-DEF.
    MOVE "New caption" TO SETVAL-DATA
    MOVE 11 TO SETVAL-LEN.
    PERFORM RESETCAP.
```

```
CALL "SP2" USING SP2-SET-FIELD-DEF SP2-FIELD-DEF.

COPY "RESETCAP.CPY"
```

The SETVAL routine mentioned in chapter C14 uses the same code as RESETVAL but includes the preceding GET-FIELD-DEF call and the succeeding SET-FIELD-DEF call.

Some properties cannot be changed without deleting and recreating the object. This is the case with the Row and Column properties that control the location of an object. To move a field, for example, you must do the following:

```
MOVE 2000 TO SP2-FD-VAR-LEN.
MOVE LOW-VALUES TO SP2-FD-VAR-LENS.
MOVE LOW-VALUES TO SP2-FD-DATA.
MOVE ACCOUNT-NUMBER-I TO SP2-FD-ID.
CALL "SP2" USING SP2-GET-FIELD-DEF SP2-FIELD-DEF.
ADD 5 TO SP2-FD-ROW
ADD 10 TO SP2-FD-COL
CALL "SP2" USING SP2-DELETE-FIELD SP2-FIELD-DEF.
CALL "SP2" USING SP2-SET-FIELD-DEF SP2-FIELD-DEF.
```

Accessing Properties Directly

The code at the start of this chapter to reset the Window Title could have been written like this:

```
MOVE LOW-VALUES TO SP2-PR-DATA.
MOVE "WVA0000000009-RL" TO SP2-PR-KEY.
MOVE "New title" TO SP2-PR-VALUE.
CALL "SP2" USING SP2-SET-PROPERTY SP2-PROPERTY.
```

This code is a little shorter than the original code mainly because it is no longer necessary to retrieve the property definition before resetting it (although of course there is a GET-PROPERTY function which corresponds to the SET-PROPERTY function). The code is fairly self-explanatory except for the second line. This is the code which tells the SET-PROPERTY function exactly which property is being accessed – the property Key. The rest of this chapter will discuss how to set this property Key. The only other thing you need to worry about is setting the Id of the object being accessed. In the case of a window or panel, an Id is not needed. For a field, use the ids in the generated copy files, for example:

```
MOVE panel-ACCOUNT-NUMBER-I TO SP2-PR-ID.
```

For a group or repeat group, use the id as listed in the editor or retrieved using the field Group Id or Repeat Id properties. For a static, use row and column:

```
 MOVE 10 TO SP2-PR-ROW.
MOVE 15 TO SP2-PR-COLUMN.
```

Once the Id of the object has been set, the property key must be set. The easiest way to find the key of a property is to look it up in Appendix F where all the properties are listed. The key is actually made up of a number of parts, defined in COBOL as follows:

```
10  SP2-PR-KEY.
    15  SP2-PR-OBJECT-TYPE  PIC X.
        88  SP2-PR-WINDOW       VALUE "W".
        88  SP2-PR-PANEL        VALUE "P".
        88  SP2-PR-STATIC       VALUE "S".
        88  SP2-PR-FIELD        VALUE "F".
        88  SP2-PR-GROUP        VALUE "G".
        88  SP2-PR-REPEAT       VALUE "R".
    15  SP2-PR-TYPE         PIC X.
        88  SP2-PR-LEN-T        VALUE "L".
        88  SP2-PR-NUM-T        VALUE "N".
        88  SP2-PR-CHAR-T       VALUE "C".
        88  SP2-PR-VAR-T        VALUE "V".
    15  SP2-PR-VAR-TYPE     PIC X.
        88  SP2-PR-VAR-1        VALUE "A".
        88  SP2-PR-VAR-2        VALUE "B".
        88  SP2-PR-VAR-3        VALUE "C".
        88  SP2-PR-VAR-4        VALUE "D".
        88  SP2-PR-VAR-5        VALUE "E".
        88  SP2-PR-VAR-6        VALUE "F".
        88  SP2-PR-VAR-7        VALUE "G".
        88  SP2-PR-VAR-8        VALUE "H".
        88  SP2-PR-VAR-9        VALUE "I".
        88  SP2-PR-VAR-10       VALUE "J".
    15  SP2-PR-OFFSET       PIC 9(5).
    15  SP2-PR-LEN          PIC 9(5).
    15  SP2-PR-FORMAT       PIC X.
        88  SP2-PR-NUMBER       VALUE "N".
        88  SP2-PR-BINARY       VALUE "B".
        88  SP2-PR-DECIMAL      VALUE "D".
    15  SP2-PR-ACTION       PIC X.
        88  SP2-PR-REDRAW       VALUE "R".
        88  SP2-PR-RECREATE     VALUE "C".
    15  SP2-PR-VAR-ACT      PIC X.
           88  SP2-PR-RESET-LEN VALUE "L".
```

By setting the values of these parts, it is possible to derive the value of the key itself without having to look it up. The parts are defined as follows:

Object type - identifies the object being accessed.
W = window
P = panel
S = static
F = field
G = group
R = repeat

Type - identifies the section within the object definition (see sp2.cpy.)
L = length
N = num-data
C = char-data
V = var-data

Var-type - identifies the sub-section within the variable section of the object definition.
A = first sub-section
B = second sub-section
C = third sub-section
Etc.

Offset - the zero-based offset (in bytes) of the property within the section.

Length - the length (in bytes) of the property within the section.

Format - the format of the property value.
Blank = character value held in pr-value
N = numeric value held in pr-num-value for a numeric property
B = binary value (00000000 thru 11111111) held in pr-bin-value for a character property, anything other than 0 or 1 means leave bit unchanged
D = decimal value (0-255) held in pr-num-value for a character property

Action - the action to be taken after the property has been set.
R = redraw object
C = recreate object

Var-act -	the action to be taken if the property is in the variable portion of the object definition. L = reset length of variable section to the length of the new value

For example, to reset the field More options property so that the field will respond to a right mouse click, use the following:

```
MOVE LOW-VALUES TO SP2-PR-DATA.
MOVE FIELD-ID TO SP2-PR-ID.
MOVE "FC-0004800001B" TO SP2-PR-KEY.
MOVE "------1-" TO PR-BIN-VALUE.
CALL "SP2" USING SP2-SET-PROPERTY SP2-PROPERTY.
```

In this case:

Object type -	F (field)
Type -	C (char-data)
Var-type -	- (N/A)
Offset -	00048
Length -	00001
Format -	B (binary value in pr-bin-value)

For some field types, it's possible to use the set-property function to exceed the 32k limit on field size. This feature allows you to assign a large amount of data to the Value property for list boxes and combo boxes and thus create extra long lists. Set pr-len to zero, set pr-var-len-l to the length of the data and move the data to pr-value-l. For example:

```
MOVE LOW-VALUES TO SP2-PR-DATA
MOVE COMBOBOX-LIST-I TO SP2-PR-ID.
MOVE "FVC0000000000-RL" TO SP2-PR-KEY
MOVE 50000 TO SP2-PR-VAR-LEN-L
MOVE SP2-PROPERTY TO MY-PROPERTY
COMPUTE WS-SUB-1 = 2 + SP2-PR-LEN-LEN + SP2-PR-NUM-LEN
                     + SP2-PR-CHAR-LEN + SP2-PR-VAR-LEN + 1
MOVE MY-DATA TO MY-PROPERTY (WS-SUB-1 :)
CALL "SP2" USING SP2-SET-PROPERTY MY-PROPERTY
```

CHAPTER C18 - Dynamic Panel Creation

As well as resetting an object's properties, as explained in the previous chapter, you can use the SET functions to actually create objects. This means that you can define your user interface without ever using the panel editor. An important application of this feature is the writing of conversion programs. If you can read in an old screen definition into your program somehow, then you should be able to write that definition out as a COBOL sp2 panel without too much trouble.

This chapter will not cover this topic in exhaustive detail for it is clearly a large topic. Instead, some basic concepts will be covered and then you should refer to the sample programs which illustrate the various techniques. Also refer to part E which fully describes all the properties that can be set when defining new objects.

The first thing to do is open a window in which to define a panel. This is done using the OPEN-WINDOW function, but two types of window can be used. The first type is a normal window that will be immediately displayed on the screen. When using this type of window, it is necessary to carefully set a number of data items in the WINDOW-DEF parameter - in this case, window properties cannot be derived from a panel definition (as in chapter C10) because there is no panel definition yet! You need to match the window to the panel that you will be defining, as follows:

```
MOVE LOW-VALUES TO SP2-WD-DATA.
MOVE 40 TO SP2-WD-WIDTH.
MOVE 19 TO SP2-WD-HEIGHT.
MOVE 16 TO SP2-WD-ROW.
MOVE 80 TO SP2-WD-COL.
MOVE -1 TO SP2-WD-TITLE-ROWS.
MOVE -1 TO SP2-WD-MENU-ROWS.
MOVE "PANEL001" TO SP2-WD-NAME.
MOVE "Window one" TO SP2-WD-TITLE.
CALL "SP2" USING SP2-OPEN-WINDOW SP2-WINDOW-DEF.
```

Here, the size and position of the window is set and the window will have a title and Menu Bar.

The second type of window that can be used is what is called a "hidden" window. This type of window is not displayed on the screen but allows panels to be defined and then used later. This is very useful if you are writing a conversion program. The technique can also be used if you want to predefine some panels at the start of your program. The size and position of a hidden window are not really important. The following is an example:

```
MOVE LOW-VALUES TO SP2-WD-DATA.
MOVE "y" TO SP2-WD-HIDE-SW.
CALL "SP2" USING SP2-OPEN-WINDOW SP2-WINDOW-DEF.
```

Once a window has been opened, a panel must be defined. This must be done panel still before any fields can be set defined.

```
MOVE LOW-VALUES TO SP2-PD-DATA.
MOVE 40 TO SP2-PD-WIDTH.
MOVE 19 TO SP2-PD-HEIGHT.
MOVE 0 TO SP2-PD-ROW.
MOVE 0 TO SP2-PD-COL.
MOVE "PANEL001" TO SP2-PD-NAME.
MOVE SP2-KEY-LEFT TO SP2-PD-LEFT.
MOVE SP2-KEY-RIGHT TO SP2-PD-RIGHT.
MOVE SP2-KEY-UP TO SP2-PD-UP.
MOVE SP2-KEY-DOWN TO SP2-PD-DOWN.
MOVE SP2-KEY-TAB TO SP2-PD-TAB.
MOVE SP2-KEY-BACKTAB TO SP2-PD-BACKTAB.
MOVE SP2-KEY-BACKSPAC TO SP2-PD-BACKSPAC.
MOVE SP2-KEY-DELETE TO SP2-PD-DELETE.
MOVE SP2-KEY-INSERT TO SP2-PD-INSERT.
MOVE SP2-KEY-HOME TO SP2-PD-HOME.
MOVE SP2-KEY-END TO SP2-PD-END.
MOVE SP2-KEY-CTRL-HOME TO SP2-PD-HOME-PAN.
MOVE SP2-KEY-CTRL-END TO SP2-PD-END-PAN.
```

```
    MOVE SP2-KEY-ENTER TO SP2-PD-CTRL-KEY (1).
    MOVE SP2-KEY-ESC TO SP2-PD-CTRL-KEY (2).
    MOVE SP2-KEY-F2 TO SP2-PD-CTRL-KEY (3).
    MOVE 500 TO SP2-PD-CTRL-KEY (4).
    CALL "SP2" USING SP2-SET-PANEL-DEF SP2-PANEL-DEF.
```

Be careful when setting the position of the panel. If the panel is going to be used immediately within a visible window, the Row and Column properties should normally be set to 0. In this case Row and Column are used for positioning the panel WITHIN the window. If the panel is being defined within a hidden window and will not be used until later, set Row and Column to the position on the screen of the window that will be derived from the panel.

The remaining properties that are being set here refer to the keys that will be used to move around the panel and to give control back to your program. Your program has to define these keys - there are no defaults set up.

Once a panel has been defined, you can set up fields within the panel. You will probably need to define both static and non-static fields. Static fields are used for general text items and field labels and for drawing lines. The following is an example:

```
    MOVE LOW-VALUES TO SP2-SD-DATA.
    MOVE 1 TO SP2-SD-ROW.
    MOVE 10 TO SP2-SD-COL.
    MOVE 200 TO SP2-SD-WIDTH.
    MOVE 10 TO SP2-SD-HEIGHT.
    MOVE "Some static text" TO SP2-SD-TEXT.
    CALL "SP2" USING SP2-SET-STATIC-DEF SP2-STATIC-DEF.
```

The basic items that have to be set are position, size and the actual text. Notice that Row and Column are in terms of cells, and Width and Height are in terms of 1/10 cells. The size of a cell defaults to 16 pixels high by 8 pixels. For rendering an item in text mode, it is assumed that the hardware character cell size is also 16 pixels by 8 pixels so if you are going to run in text mode it is advisable to stick to this default.

Non-static fields are a little more complex. The following defines a custom data entry field:

```
    MOVE LOW-VALUES TO SP2-FD-DATA.
    MOVE 1 TO SP2-FD-ID.
    MOVE 5 TO SP2-FD-ROW.
    MOVE 5 TO SP2-FD-COL.
    MOVE 100 TO SP2-FD-WIDTH.
    MOVE 0 TO SP2-FD-HEIGHT.
    MOVE 10 TO SP2-FD-MAX-LEN.
    MOVE 10 TO SP2-FD-PROG-LEN.
    MOVE 0 TO SP2-FD-PROG-OFF.
    MOVE 1 TO SP2-FD-PROG-NUM.
    MOVE "l" TO SP2-FD-BOR-TYPE.
    MOVE 0 TO SP2-FD-VAR-LEN.
    MOVE LOW-VALUES TO SP2-FD-VAR-LENS.
    CALL "SP2" USING SP2-SET-FIELD-DEF SP2-FIELD-DEF.
```

Here some extra items have to be set. Notice first that Height has been set to 0. This means that a default height should be used because it is hard to know what height to use for a field with a border.

All non-static fields have to have Id set. Here an id is being supplied by the program - if the FD-ID had been set to zero, an id would have been assigned automatically.

As well as having a physical width (the Width property), entry fields also have a width in terms of the number of characters that they can hold (the Maximum length property or FD-MAX-LEN item). This number of characters need not be related in any way to the visible width of the field because all entry fields scroll if necessary.

When you set up an entry field, you will normally want to be able to capture data entered in your program. This is done by setting two more properties, Program length (FD-PROG-LEN) and Program offset (FD-PROG-OFF). Program length is the length of the data that is to be returned to the program and will normally be the same as Maximum length. Program offset is the offset of the data within your program's Field area. Normally, you will set this offset to 0 for the first field that you define and then increment it as you add more fields that will be passing data back to your program. It is important to make sure that you set this item correctly, otherwise you will not be able to process your user's data properly.

The last property set here is the type of border for the field (Border property). In this case the border will be a regular line border.

It is not necessary here to set any of the variable length field properties. The one that you may normally want to set is Format (FD-FMT) which holds the COBOL format of the field. If this format is not supplied the format defaults to PIC X for a length specified by Maximum length.

The default layout for FIELD-DEF defines the variable length properties area as a single PIC X item for a length of 2000. This definition allows needed flexibility because the makeup of the variable data area will be different depending upon the type of field that is being defined. To set up a value within this area use reference modification. The following sets a COBOL field format specifically:

```
   MOVE "X(10)" TO SP2-FD-VAR-DATA (1 : 5).
   MOVE 5 TO SP2-FD-FORMAT-LEN.
   ADD 5 TO SP2-FD-VAR-LEN.
```

Once you have defined all your fields, your panel is complete and can either be used directly in a CONVERSE-PANEL call or the definition can be written to a file. If you are going to process the panel immediately, you do not have a generated converse-data area as you would have had if you had used the panel editor to define the panel. A generalized converse-data area is defined within the SP2.CPY copy file and this can be used for processing any number of panels. The important thing to notice is that there is no Fields area defined in this converse area, therefore, data entered by the user will not be immediately accessible. There are two solutions to this problem. The first is to set up an independent Fields area for the panel and then use the SET-PANEL-FIELDS function to assign this area:

```
        MOVE LOW-VALUES TO FIELD-AREA-1.
        CALL "SP2" USING SP2-SET-PANEL-FIELDS FIELD-AREA-1.
```

This area must be big enough to hold all the fields which have a non-zero Program length (FD-PROG-LEN). It does not matter if it is bigger. You can use reference modification to move values in and out of this area based on Program offset (FD-PROG-OFF) and Program length (FD-PROG-LEN).
Once the Fields area has been set up, the panel can be processed with a CONVERSE-PANEL call using the generalized converse-data area:

```
        MOVE LOW-VALUES TO SP2-CD-DATA.
        MOVE "PANEL001" TO SP2-CD-NEXT-PANEL.
        CALL "SP2" USING SP2-CONVERSE-PANEL SP2-CONVERSE-DATA.
```

The second way of retrieving data entered by the user is to use the GET-FIELD-DATA function. If you use this method, you do not have to set up a Fields area at all. The following code will retrieve data entered into the ACCOUNT-NUMBER field (assuming that this field has an Id of 1):

```
        MOVE LOW-VALUES TO SP2-FD-DATA.
        MOVE 1 TO SP2-FD-ID.
        MOVE LOW-VALUES TO SP2-FD-VAR-LENS
        MOVE 200 TO SP2-FD-INITIAL-LEN
        CALL "SP2" USING SP2-GET-FIELD-DATA SP2-FIELD-DEF.
```

The data will be returned in FD-VAR-DATA up to a length of 200.

You may have a problem with using the generalized CONVERSE-DATA area if you are using multiple windows. This is because some of the data items within CD-DATA (cursor position, for instance) may have inappropriate values when you return to a previous panel. The solution is to save off CD-DATA to a panel-specific area before you start processing a new panel and then restore CD-DATA when the window containing the panel is reactivated.

If you are not going to use the panel immediately and wish to save the panel definition to a file, use the WRITE-PANEL function:

```
   CALL "SP2" USING SP2-WRITE-PANEL SP2-NULL-PARM.
```

This writes off the panel to the current panel file - make sure you have opened one with OPEN-FILE or CREATE-FILE and that you set FI-MODE to " w". If you need to define another panel in the same hidden window (as might be the case with a conversion program), use the CLEAR-WINDOW function to clear out the hidden window. Even though the window is hidden, remember to close it using CLOSE-WINDOW when you have finished defining panels.

As mentioned above, please refer to the sample programs for more details on dynamic panel definition, including the definition of other types of fields.

CHAPTER C19 - Scrollable Windows

All windows that are resizable will scroll as necessary to display their contents. Such resizing is normally done by the user after a window is displayed. What if you need a window to be set up for scrolling as soon as it is displayed? This might be the case if you have a report to display that is wider than the screen. By default, the panel editor sets the size of the panel and the size of the window in which it is to be displayed to be the same. This size is set using the View/Size menu option as explained in Chapter B5.

If you know at design time how much the window will need to scroll, you can merely set the panel Total width and height properties to the appropriate values using the panel editor.

If you want to control at runtime the amount that a window may scroll, you must reset the Total width and height properties programmatically. The following code resets the total size of the window:

```
MOVE LOW-VALUES TO SP2-WD-DATA.
MOVE "BIGPANEL" TO SP2-WD-PANEL-NAME.
CALL "SP2" USING SP2-OPEN-WINDOW SP2-WINDOW-DEF.
MOVE "BIGPANEL" TO SP2-WD-NAME.
CALL "SP2" USING SP2-GET-WINDOW-DEF SP2-WINDOW-DEF.
MOVE 132 TO SP2-WD-TOT-WIDTH.
MOVE "x" TO SP2-WD-SBAR-SW.
CALL "SP2" USING SP2-SET-WINDOW-DEF SP2-WINDOW-DEF.
```

Here, a window is opened by deriving it from the panel definition, as usual. Next the window definition is retrieved so that it can be modified slightly. The property that needs resetting is the Total width property (WD-TOT-WIDTH) which represents the total width of the window as opposed to the visible width which is held in the Width property (WD-WIDTH). Also, a special switch needs to be set to cause the window dimensions to be recalculated by the system. This switch, the Scrollbar switch (WD-SBAR-SW), will cause Scroll Bars to be drawn showing that the user can now scroll the window to see its contents.

A normal CONVERSE-PANEL call will complete the painting of the window and give control to the user. Remember not to set NEW-WINDOW-SW, because the window has already been opened.

CHAPTER C20 - Fonts and Colors

When a special font or color is assigned to a panel or field, what is actually assigned is a code for that font or color - the font or color definition itself is held separately in a file called FONTS.SP2. This means that a certain font or color only has to be defined once but can be used any number of times throughout your system. If the definition is changed, then all the objects with the corresponding font or color code assigned will automatically be changed.

Although this technique is very convenient in most cases, there are some things of which you should beware:

1. If you are developing on a network, it is best to locate the FONTS.SP2 file in a public directory on the network. If you have a copy of the FONTS.SP2 file in a private directory, you may find that other developers have used the same font- or color-ids as you but for different purposes.

2. When you distribute your finished system, you should distribute your FONTS.SP2 file along with your panel files to your customers. This might cause a problem if a customer happened to already have a system written with COBOL sp2. If you are concerned about this, you can put some initialization code at the start of your system to set up fonts and colors programmatically so that the FONTS.SP2 file does not need to be referenced. Use the following code:

```
MOVE LOW-VALUES TO SP2-FO-DATA.
MOVE 1 TO SP2-FO-ID.
MOVE 6 TO SP2-FO-WIDTH.
MOVE 12 TO SP2-FO-HEIGHT.
MOVE "b" TO SP2-FO-WEIGHT.
MOVE "Helv" TO SP2-FO-NAME.
CALL "SP2" USING SP2-SET-FONT-DEF SP2-FONT-DEF.
....etc.
MOVE LOW-VALUES TO SP2-CO-DATA.
MOVE 11 TO SP2-CO-ID.
MOVE "t" TO SP2-CO-TYPE (1).
MOVE X"00" TO SP2-CO-TEXT (1).
MOVE "t" TO SP2-CO-TYPE (2).
MOVE X"03" TO SP2-CO-TEXT (2).
CALL "SP2" USING SP2-SET-COLOR-DEF SP2-COLOR-DEF.
....etc.
```

You do not need to have a window open in order to do the above.

CHAPTER C21 - Using an Existing Window for a New Panel

Many existing systems use a succession of full-screen panels to process user-input. While this is perhaps not the norm in a contemporary user interface, there is no reason why this technique cannot be replicated using COBOL sp2.

The first panel to be displayed can be processed as normal using either an OPEN-WINDOW function call followed by a CONVERSE-PANEL function call, or a CONVERSE-PANEL function call on its own with the NEW-WINDOW switch set. To display the next panel, the most straightforward thing to do would be to close the window with a CLOSE-WINDOW function call and then repeat the process. This will work quite satisfactorily but you may feel that it is not really appropriate to close the window in these circumstances.

In order to use the same window for the new panel, the first thing to do is make a CLEAR-WINDOW function call to clear out the old panel. If you do not do this, fields in the old panel may "bleed" through when the new panel is displayed.

If you had a menu connected to the old panel, you may also need to make a CLEAR-MENU call to clear out this menu. This is not always the case, however, because you may want to use a common Menu Bar for all your panels. If this is so, omit the CLEAR-MENU call.

Once the window has been cleared, you can make a CONVERSE-PANEL call to display the new panel. This call will be very similar to any other CONVERSE-PANEL call with one exception, the use of the NEW-WINDOW switch. Clearly in this case we do not want to set the switch to "y" because there is already a window open. Nevertheless, the switch does need to be set so as to ensure that the panel is displayed properly.

The problem is the values of the panel Row and Column properties. Normally these properties are used to derive the position of the window on the screen. This is the case both when you use CONVERSE-PANEL with NEW-WINDOW set to "y" and when you use OPEN-WINDOW with a valid panel name. In this case, however, a window is not being derived from a panel because a window is already open. When a panel is displayed within this window, panel Row and Column will be used as the position of the panel within the window. This is probably not what you want - you need the panel to be aligned with the top left hand corner of the window as normal.

The solution is to set the NEW-WINDOW switch to "x". This will cause the Row and Column properties to be automatically reset to zero and the panel will be displayed neatly within the window.

CHAPTER C22 - Child Windows

Child windows are essentially windows within windows. Strictly speaking, in a graphical environment, all the field types except static text items, custom data entry fields and non-windowed icons are child windows (the system window handle is held in the field GUI id and GUI id (2) properties). These child windows need no special treatment as far as you are concerned. However, it is possible to define your own child window which may contain a whole new panel.

Please be aware that child windows should be used only in special situations. Normally pop-up windows or dialog boxes (see Chapter C8) are perfectly satisfactory. Although not child windows, this type of window does have a relationship to the window that was opened before it - the previous window is by default the owner of the new window. This means that the new window will always appear in front of its owner and that the window will not be included in any list of main windows maintained by the operating system.

One example of using child windows is the panel editor itself, which uses a child window to implement the format window within the main window, which also contains the editor icon toolbar, treeview and properties box. This is the case in most environments. However, some systems do not support child windows like this. Please refer to the release notes to check if this applies to your system.

To open a child window, the only thing you need is the id of the parent window. This is retrieved with the GET-WINDOW-DEF call after the parent window has been opened.

You must use the OPEN-WINDOW function rather than CONVERSE-PANEL with the NEW-WINDOW switch to actually open the child window:

```
MOVE LOW-VALUES TO SP2-WD-DATA.
MOVE "PANEL001" TO SP2-WD-PANEL-NAME.
CALL "SP2" USING SP2-OPEN-WINDOW SP2-WINDOW-DEF.
CALL "SP2" USING SP2-DISPLAY-WINDOW SP2-NULL-PARM.
MOVE "PANEL001" TO SP2-WD-NAME.
CALL "SP2" USING SP2-GET-WINDOW-DEF SP2-WINDOW-DEF.
MOVE SP2-WD-WINDOW-ID TO WS-PARENT-ID.
MOVE LOW-VALUES TO SP2-WD-DATA.
MOVE WS-PARENT-ID TO SP2-WD-OWNR-ID.
MOVE "PANEL002" TO SP2-WD-PANEL-NAME.
CALL "SP2" USING SP2-OPEN-WINDOW SP2-WINDOW-DEF.
CALL "SP2" USING SP2-DISPLAY-WINDOW SP2-NULL-PARM.
```

This code opens a parent window derived from PANEL001 and displays it, then opens a child window derived from PANEL002 and displays it. Once the two windows are open, you can use either CONVERSE-PANEL or GET-INPUT to process user input. When using a child window like this, it is normal to allow the user to freely switch between the parent window and the child window. To allow this, use the SWITCH-SW on your function call. This is explained in chapter C16. If you not do this you will need to use the ACTIVATE-WINDOW function to switch between windows programmatically, which is also explained in chapter C16.

CHAPTER C23 - Controlling the Mouse

Normally mouse events are translated by the runtime into a useful function and your program does not need to do anything special in order to handle the mouse. For example, if the mouse is clicked on a pushbutton, the key code for that pushbutton is returned; if the mouse is clicked on an entry field, the cursor is moved to that entry field; if the mouse is double-clicked, the (ENTER) key is returned.

There may be situations, however, where it is helpful to monitor the mouse more closely:

1. Detecting double-clicks - if you do not want double-clicks to be translated into (ENTER) keys, unset this option by setting the panel Miscellaneous property to X'40'. SP2-KEY-DOUBLE-CLICK (-12) will be returned whenever the mouse is double-clicked.

2. Detecting single-clicks - set the converse panel Mouse switch property (XXXXXXXX-MOUSE-SW) to "y" before calling the CONVERSE-PANEL or GET-INPUT function and SP2-KEY-MOUSE (-10) will be returned whenever the mouse is clicked. Detecting mouse clicks like this works only for custom data entry fields unless the converse panel Capture switch property is set (see below).

3. Detecting right mouse button clicks - set the converse panel Mouse switch property (XXXXXXXX-MOUSE-SW) to "r" or "o" before calling the CONVERSE-PANEL or GET-INPUT function ("o" means detect left clicks as well). SP2-KEY-CLICK-RIGHT (-11) will be returned for single clicks and SP2-KEY-DOUBLE-RIGHT (-13) will be returned for double clicks. Unlike left mouse button clicks, a right mouse button click does not effect current cursor position (field focus). However, the following converse panel properties are set:

   Last field id (LAST-FLD-ID) (will be 0 if click is outside a field)
   Last field number (LAST-FLD-NUM) (will be 0 if click is outside a field)
   Last tab number (LAST-TAB-NUM) (will be 0 if click is outside a field)
   Last occurrence (LAST-OCCURS) (will be 0 if field is not in a repeat group)
   Last row (LAST-ROW) (may reflect a position outside a field)
   Last column (LAST-COL) (may reflect a position outside a field)
   Last vertical displacement (LAST-VERT) (will be 0 if field is not in a repeat group)

   If you want to cause the right click to move the cursor to the field that has been clicked on, move LAST-FLD-ID to NEXT-FLD-ID.

4. Capturing the mouse - if you want to have complete control of the mouse, you may capture it by setting the converse panel Capture switch (XXXXXXXX-CAPTURE-SW) to "y" before calling the CONVERSE-PANEL or GET-INPUT function. This will prevent the mouse from being used to activate pushbuttons, etc. and SP2-MOUSE-CLICK (-10) will be returned for all clicks as long as Mouse switch (XXXXXXXX-MOUSE-SW) is set to "y". Mouse capture remains on until the mouse pointer is moved outside the current window. If you want capture to remain on wherever the pointer is moved, set Capture switch (XXXXXXXX-CAPTURE-SW) to "x". This is useful for setting the mouse pointer to a new shape and holding that shape wherever the mouse is moved while you perform some processing:

```
MOVE SP2-MOUSE-WAIT TO SP2-NP-RET-CODE.
CALL "SP2" USING SP2-SET-MOUSE-SHAPE SP2-NULL-PARM.
MOVE "x" TO PANEL-CAPTURE-SW.
MOVE "k" TO PANEL-WAIT-SW.
MOVE SP2-KEY-TIMEOUT TO PANEL-KEY.
PERFORM UNTIL DATA-PROCESSED
           OR PANEL-KEY NOT = SP2-KEY-TIMEOUT
    PERFORM PROCESS-DATA
    CALL "SP2" USING SP2-GET-INPUT PANEL-CONVERSE-DATA
END-PERFORM.
MOVE "n" TO PANEL-CAPTURE-SW.
MOVE "n" TO PANEL-WAIT-SW.
CALL "SP2" USING SP2-GET-INPUT PANEL-CONVERSE-DATA.
MOVE SP2-MOUSE-ARROW TO SP2-NP-RET-CODE.
CALL "SP2" USING SP2-SET-MOUSE-SHAPE SP2-NULL-PARM.
```

Notice that mouse capture must be specifically turned off by setting Capture switch (XXXXXXXX-CAPTURE-SW) to "n".

CHAPTER C24 - Multi-panel Windows

Normally a window will be completely filled with a single panel and this is the easiest way to handle things. In certain cases, however, it may be desirable to have more than one panel in a window. One situation when this might be so is if you need to migrate an older system that uses multiple panels or forms. This should be no problem but there are a few rules and limitations that apply when using multiple panels in a window:

1. Unless you are defining the panel programmatically (using a conversion program, for example), it is usually easiest to set the position of the panel in the window using the converse panel Panel position switch (PAN-POS-SW), Panel row (PAN-ROW) and Panel column (PAN-COL) properties.

2. If a panel is not the active panel in a window, it must contain only the following field types: custom entry, static and icon (non-windowed).

3. A panel may be activated using the DISPLAY-PANEL function which takes the name of the panel in the NAME-DEF parameter.

4. A panel may be closed using the CLEAR-PANEL function which also takes the name of the panel in the NAME-DEF parameter.

A secondary panel in a window is normally used for displaying information that is common to many windows. A secondary panel may only be used as a regular input panel if all panels within the window use only custom fields as a means of input.

CHAPTER C25 - Toolbars

See chapter B33 for details on setting up a toolbar using the editor.

When the main panel is displayed at runtime, the toolbar will be automatically displayed. Input from the toolbar will be reflected in the CONVERSE-DATA area as KEY-TOOLBAR (-15) with MENU-ID holding the code of the button pressed and LAST-FLD-ID holding its id (NEXT-FLD-ID will point to the current field in the main panel). Although most toolbar fields (usually icon fields) will merely return codes in MENU-ID, values from radio buttons, etc. will be returned in the regular Fields area.

Space is automatically set aside by the generator at the end of the Fields area for toolbar fields. This allows you to retrieve values from radio buttons, etc. as mentioned above but it also allows you to reset the Protection property of toolbar fields using the Types area. For example, to gray out a toolbar icon, you can merely set the appropriate Types area item to "g".

Reserving this space is a problem, however, if you want to use the same toolbar for multiple panels (see chapter C21) because the layout of the CONVERSE-DATA area will be entirely different for second and subsequent panels. In this case, it is important to specify that no space be reserved in the Fields, Colors or Types area for toolbar fields. Do this by setting the Program data property to X"06" and the Program length and number properties to zero for each toolbar field. Alternatively, you can set configuration variable SP2TBF=1 to specify that the program areas are to be ignored for toolbars. To gray out a toolbar field in this situation, you must directly reset the Protection property for the toolbar field. Use the following code:

```
CALL "SP2" USING SP2-ACTIVATE-TOOLBAR SP2-NULL-PARM.
MOVE 2000 TO SP2-FD-VAR-LEN.
MOVE LOW-VALUES TO SP2-FD-VAR-LENS.
MOVE LOW-VALUES TO SP2-FD-DATA.
MOVE TOOLBAR-FIELD-I TO SP2-FD-ID.
CALL "SP2" USING SP2-GET-FIELD-DEF SP2-FIELD-DEF.
MOVE "g" TO SP2-FD-OUTPUT.
CALL "SP2" USING SP2-SET-FIELD-DEF SP2-FIELD-DEF.
CALL "SP2" USING SP2-DEACTIVATE-TOOLBAR SP2-NULL-PARM.
```

A toolbar is actually just another panel with its name used as the value of the Toolbar property of the main panel. As above, you can modify this panel just like any other panel but you must first activate the window containing the toolbar. This window is actually the parent window for the main panel window but you can use the ACTIVATE-TOOLBAR function to make this window current and the DEACTIVATE-TOOLBAR function to return to the main window.

Once the toolbar window is current, you can perform any of the normal panel functions to modify the toolbar. The following is a list of some of the functions which might be appropriate:

CLEAR-WINDOW	erase the current toolbar
READ-PANEL	read an existing toolbar definition into the current toolbar window.
SET-PANEL-DEF	define a new toolbar dynamically
SET-FIELD-DEF	define a new toolbar field dynamically

One very important thing to notice is that if a window is to contain a toolbar, it must be initially opened in a "toolbar-ready" state. This means that the window Toolbar rows property must be non-zero when the window is opened. This is normally set automatically if the panel used to open the window has a toolbar connected to it (Toolbar property is not blank. If Toolbar rows is positive, an area of this height (in scaled pixels) will be reserved for the toolbar. If it is -1, no area will be reserved but a toolbar may be added later using the OPEN-TOOLBAR function (see below).

Once a toolbar is in place in a window, you can use CLEAR-PANEL and CONVERSE-PANEL with NEW-WINDOW set to "x" to process additional panels in the same window with this same toolbar, i.e. the toolbar only has to be defined once for the first panel used.

The following additional functions can be used in conjunction with toolbars:

OPEN-TOOLBAR - dynamically creates a toolbar in an existing "toolbar-ready" window. The parameter for this function is WINDOW-DEF with WD-PANEL-NAME set to the name of the toolbar panel or WD-HEIGHT set to the height of the toolbar (in scaled pixels) if the toolbar panel has not yet been created. "Toolbar-ready" means that the existing window must have been opened with the Toolbar rows property (WD-TOOLBAR-ROWS) set to non-zero.

ACTIVATE-TOOLBAR - activates the toolbar so that its definition can be modified using SET-FIELD-DEF, etc.

DEACTIVATE-TOOLBAR - reactivates the main panel after ACTIVATE-PANEL has been used to activate the toolbar.

DISPLAY-TOOLBAR - force refresh of the toolbar. The contents of a toolbar do not usually change as often as those of the main panel so toolbars are not automatically refreshed when you use DISPLAY-WINDOW or CONVERSE-PANEL for the main panel. If you are using CONVERSE-PANEL, you can also force a toolbar refresh by setting DISPLAY-SW to "t".

CLOSE-TOOLBAR - closes toolbar and shrinks overall window. A new toolbar can be opened using OPEN-TOOLBAR.

See the toolbar sample program for an example of coding for a toolbar.

CHAPTER C26 - Extended repeat groups

In chapter C12, it was explained that your program views a repeat group as a table containing all the occurrences. What if you want to use a repeat group to browse through a file containing thousands of records? You probably don't want to load all those records into a working-storage table in one go and how big should you make the table anyway? The answer is to use a regular repeat group with a fixed number of occurrences but process it in a slightly different way.

This technique is called extended repeat group processing and it allows you to pass data to a repeat group on a buffered basis. To use extended repeat processing, you need to make a special function call, SET-REPEAT-EXT, prior to making your CONVERSE-PANEL call. This means that you must also use an OPEN-WINDOW function call to open a window for your panel rather than using the NEW-WINDOW switch on the CONVERSE-PANEL call.

The SET-REPEAT-EXT function call sets up details of the data which you will be displaying in the repeat group. It uses the REPEAT-EXT parameter which is described in detail in chapter E14.

The idea is that data is read from a file in blocks. Once your panel has been displayed and the user starts scrolling down through the data, if the data in the current block is exhausted, control is returned to your program so that a new block can be retrieved. The big advantage of this processing over more traditional page-by-page processing is that the user can still scroll up and down one line at a time or use a Scroll Bar to scroll using the mouse.

Because of a limitation on the length of the program Fields area, previous versions of SP2 required that individual blocks be moved to the Fields area and only a single block passed to SP2. This in turn meant that blocks had to be retrieved randomly from a file in order to support the user's viewing different sections of the file. The recommended method for doing this was building a table of record keys that could be linked to displacements within the repeat group. Sample program REPEAT.CBL is an example of this technique.

Now that there is no real limit on the size of the Fields area, it is easiest to define a repeat group with a high number of occurrences and load blocks sequentially into the Fields area as needed. This means that records can be read sequentially from a file and no record has to be read more once. The following is an example of this:

```
          MOVE LOW-VALUES TO SP2-RX-DATA.
          MOVE 1 TO SP2-RX-ID.
          MOVE 3 TO SP2-RX-BLOCK-SW.
          MOVE 20 TO SP2-RX-BLOCK-OCCS.
          MOVE 1 TO SP2-RX-TOTAL-SW.
          MOVE 999 TO SP2-RX-TOTAL-OCCS.
          PERFORM PROC-MORE-DATA THRU PROC-MORE-DATA-EXIT.
          PERFORM PROC-REPEAT
              UNTIL REPEAT-KEY = SP2-KEY-CLOSE.
          . . .

      PROC-REPEAT.
     *****************************************************
     * PROCESS PANEL INPUT AND GET MORE DATA IF NECESSARY *
     *****************************************************
          CALL "SP2" USING SP2-CONVERSE-PANEL REPEAT-CONVERSE-DATA.
          IF REPEAT-KEY = SP2-KEY-MORE
              PERFORM PROC-MORE-DATA.

      PROC-MORE-DATA.
     **********************************
     * GET MORE DATA FOR REPEAT GROUP *
     **********************************
          PERFORM PROC-GET-BLOCK.
          CALL "SP2" USING SP2-SET-REPEAT-EXT SP2-REPEAT-EXT.

      PROC-GET-BLOCK.
     *********************************
     * GET RECORDS FOR CURRENT BLOCK *
     *********************************
```

```
          COMPUTE SP2-RX-BLOCK-OCCS = REPEAT-NEXT-VERT + 20
          PERFORM PROC-GET-RECORD THRU PROC-GET-RECORD-EXIT
              UNTIL WS-RECORD-NUM = SP2-RX-BLOCK-OCCS.
```

The displacement of the new data required is returned in REPEAT-NEXT-VERT which is in the panel's converse-data area. The suggested technique is to sequentially read the file until you have retrieved the records needed to fill the block which is expanded as necessary based on REPEAT-NEXT-VERT. Please see the sample program, NEWREP.CBL, for a complete example of using this technique.

The method described above can deal with a file with up to 9999 records (or 32000 if your compiler can handle 5 digit numbers in a 2 byte binary item). If you need even more capacity, you need to make the following adjustments:

1. Set SP2-RX-BLOCK-SW to –3 rather than 3.

2. Set SP2-RX-TOTAL-OCCS-L rather than SP2-RX-TOTAL-OCCS.

3. Use SP2-RX-NEW-DISP-L rather than REPEAT-NEXT-VERT to derive the number of the first record that needs to be retrieved.

4. After SP2-KEY-MORE is returned, make a GET-REPEAT-EXT call to retrieve the new value of SP2-RX-NEW-DISP-L.

Please see the sample program, NEWREP2.CBL for an example of using extended repeat processing with these extra capacity (long) data items.

The behavior of extended repeat groups can in many cases be enhanced under Windows by using sp2thred.dll instead of sp2.dll and by setting the repeat group Miscellaneous X"02" switch. In particular, this allows data to be scrolled past the current block using the mouse. What actually happens is that sp2thred.dll allows sp2-key-more to be returned to your program before the mouse button is released. Because of this it is important to make only get-repeat-ext and set-repeat-ext calls in response to sp2-key-more before continuing with get-input or converse-panel - other calls will be ignored. It is also possible that sp2-key-more might be returned before sp2-key-switch when the user switches windows and immediately starts scrolling. In this case, the Next panel property will be set to reflect the new panel as if sp2-key-switch had been returned and the Menu id property will be set to the id of the repeat group being scrolled.

CHAPTER C27 - Tab Controls

Tab controls allow a number of child windows to be grouped together in one parent window and conveniently accessed using tabs appearing at the top of the parent window. In the editor, set Window border type to “t” for the panels that will appear in the tab windows. The tab windows will automatically be resized at runtime to fill the parent window. If the parent window does not have Window border type of “m” (main window), the height of the tab panel controls how much space is left at the bottom of the parent window for push buttons, etc.

Programmatically tab windows are handled in the same way as regular child windows (see chapter C22.) Open the parent window and retrieve its Id:

```
MOVE LOW-VALUES TO SP2-WD-DATA.
MOVE "tabs" TO SP2-WD-PANEL-NAME.
CALL "SP2" USING SP2-OPEN-WINDOW SP2-WINDOW-DEF.
CALL "SP2" USING SP2-DISPLAY-WINDOW SP2-NULL-PARM.
CALL "SP2" USING SP2-GET-WINDOW-DEF SP2-WINDOW-DEF.
MOVE SP2-WD-WINDOW-ID TO WS-OWNER-ID.
```

Open the tab windows:

```
MOVE LOW-VALUES TO SP2-WD-DATA.
MOVE "tab1" TO SP2-WD-PANEL-NAME.
MOVE WS-OWNER-ID TO SP2-WD-OWNR-ID.
CALL "SP2" USING SP2-OPEN-WINDOW SP2-WINDOW-DEF.
CALL "SP2" USING SP2-DISPLAY-WINDOW SP2-NULL-PARM.
MOVE LOW-VALUES TO SP2-WD-DATA.
MOVE "tab2" TO SP2-WD-PANEL-NAME.
MOVE WS-OWNER-ID TO SP2-WD-OWNR-ID.
CALL "SP2" USING SP2-OPEN-WINDOW SP2-WINDOW-DEF.
CALL "SP2" USING SP2-DISPLAY-WINDOW SP2-NULL-PARM.
MOVE "tab2" TO WS-NEXT-PANEL
MOVE "y" TO TAB1-SWITCH-SW TAB2-SWITCH-SW
```

Accept input into the tab windows, switching between them as appropriate:

```
PERFORM UNTIL WS-KEY = SP2-KEY-CLOSE
    IF WS-NEXT-PANEL = "TAB1"
        MOVE "tab1" TO TAB1-NEXT-PANEL
        CALL "SP2" USING SP2-CONVERSE-PANEL
                         TAB1-CONVERSE-DATA
        MOVE TAB1-NEXT-PANEL TO WS-NEXT-PANEL
        MOVE TAB1-KEY TO WS-KEY
    ELSE
        IF WS-NEXT-PANEL = "TAB2"
            MOVE "tab2" TO TAB2-NEXT-PANEL
            CALL "SP2" USING SP2-CONVERSE-PANEL
                             TAB2-CONVERSE-DATA
            MOVE TAB2-NEXT-PANEL TO WS-NEXT-PANEL
            MOVE TAB2-KEY TO WS-KEY
        END-IF
    END-IF
END-PERFORM
```

See also Appendix B for details on the SP2TAB configuration variable which allows you to control the behavior and appearance of the tabs.

CHAPTER C28 - Panel File Records in Working-Storage

Panel file records can be stored in working-storage eliminating the need for panel files at runtime. The main disadvantage of this technique is that your programs will be bigger and performance may be degraded in thin client mode - records will be transmitted each time they are used rather than on a one-time basis.

The working-storage definitions can be generated along with the regular COBOL code using a new version of the code generator - use the template file sp2nofil.cbx rather than sp2.cbx if you want to do this (see Appendix E for details on the code generator).

A new function, sp2-set-record, allows the records to be moved to sp2 memory so that they can be accessed by converse-panel/open-window logic as if they had just been read from a panel file. For example:

```
CALL "SP2" USING SP2-SET-RECORD MAINMENU-PANEL-RECORD.
CALL "SP2" USING SP2-SET-RECORD MAINMENU-MENU-RECORD.
MOVE LOW-VALUES TO SP2-WD-DATA.
MOVE "MAINMENU" TO SP2-WD-PANEL-NAME.
CALL "SP2" USING SP2-OPEN-WINDOW SP2-WINDOW-DEF.
```

Individual set-record calls must be made for each panel, toolbar and menu being used. A set-record call should normally be made immediately prior to a record being used - all records are cleared from memory when a close-window or clear-window call is made unless any container windows are open.

A single set-record call for fonts and colors is sufficient to load all pre-defined fonts and colors into memory. However, it may be better to include this code in a copy file or called routine so that the code only has to be changed once (i.e. regenerated and recompiled) when a font or color is changed.

CHAPTER C29 - Working with Container Panels

As explained in Chapter B36, container panels can cut down the amount of COBOL code needed to interact with sp2 panels. For example, in many cases the rules for triggering the display of a panel can be defined at panel definition time so you do not have to write code for doing this. Additionally, the code generated for container panels can be more extensive so that less code has to be hand-written.

To generate code for a container panel, the regular SP2.CBX template file should be used. You need only generate code for the container panel itself - code for the subpanels will be automatically included.

In the generated working-storage there is a consolidated Fields, Types and Colors area that includes the fields for the container and all its subpanels.. For example:

```
05  newexamp-FIELDS.
  08  newexamp-SUBFIELDS.
    10  newexamp-ACCOUNT-NUMBER
                            PIC 9(006).
    etc.
  08  ORDER-SUBFIELDS.
    10  ORDER-ACCOUNT-NUMBER
                            PIC 9(006).
    etc.
```

The next subpanel to be displayed is specified in NEXT-SUBPANEL and the current subpanel is returned n LAST-SUBPANEL:

```
10  newexamp-NEXT-SUBPANEL
                        PIC X(8).
10  newexamp-LAST-SUBPANEL
                        PIC X(8).
```

Set NEXT-SUBPANEL to low-values to display the container panel itself or an external panel.

In the generated procedure division, the following paragraphs will be generated:

`PROC-SET-FIELDS` -	include code here to initialize items in the Fields, Types and Colors areas.
`PROC-panel-OPENED` -	if panel Options-5 X"01" switch has been set, include code here for when a panel is first displayed.
`PROC-panel-ENTERED` -	if panel Options-5 X"02" switch has been set, include code here for when a panel gains the focus.
`PROC-panel-EXITED` -	if panel Options-5 X"04" switch has been set, include code here for when a panel loses the focus.
`PROC-panel-CLOSED` -	if panel Options-5 X"08" switch has been set, include code here for when a panel is closed. A popup panel is closed automatically when the Enter, Escape or Close keys are pressed - the code of the key used will be returned in panel-MENU-ID.
`PROC-panel-#-RETURNED` -	if Program control has been set for a field in a panel, include code here for when control is returned as a result of exiting this field (Id = #).
`PROC-panel-#-SELECTED` -	if Return if selected has been set for a field (Id = #) in a panel, include code here for when control is returned as a result of selecting this field (Id = #).
`PROC-MENU-#` -	if a menu is present, include code here for when control is returned as a result of selecting this menu option (Id = #).

`PROC-TOOLBAR-#-`	if a tool bar is present, include code here for when control is returned as a result of selecting a tool bar item (Id = #).
`PROC-panel-KEY-#-`	include code here for when control is returned as a result of pressing this Control key (Code = #) or clicking on a push button with this key as Help key.
`PROC-panel-KEY-CLOSE-`	include code here for when control is returned as a result of closing the panel with the window Close icon or the Alt+F4 key.

See the newexamp, newtabs and newtabs2 sample programs for example of container panel code.

PART D

COBOL sp2 Programming Functions

CHAPTER D1 - Introduction to Functions

This part of the Reference Manual explains the programming functions in detail. This chapter explains the various categories of functions that are available and the subsequent chapters define each function in detail.

Function calls are made to perform a specific task. Each function has an appropriate descriptive name and uses a single parameter which holds all data items needed by the function. This helps makes your COBOL code easily understood and therefore easy to maintain.

Functions are classified into the following categories:

- Primary
- Window
- Panel
- Field
- Static field
- Group
- Repeat group
- Font
- Color
- Menu
- Message
- Clipboard
- Toolbar
- File
- Miscellaneous

Primary functions are those which you will almost always use in your program. The rest of the functions are mostly classified based on the objects which they are used to access.

Primary

OPEN-FILE - open a panel file
CONVERSE-PANEL - display and process a panel
CLOSE-WINDOW - close window
CLOSE-FILE - close panel file
END-SESSION - end session prior to termination
SET-RECORD - set record in memory

Window

OPEN-WINDOW - open a window
DISPLAY-WINDOW - display current window
CLEAR-WINDOW - clear window
ACTIVATE-WINDOW - activate a window
ACTIVATE-INTERNAL - activate a window without visibly activating it
GET-WINDOW-DEF - get window properties
SET-WINDOW-DEF - set window properties
DEACTIVATE-WINDOW - deactivate a window

Panel

GET-INPUT - get user input
SET-PANEL-FIELDS - set Fields area for panel
SET-PANEL-COLORS - set Colors area for panel
SET-PANEL-TYPES - set Types area for panel
GET-PANEL-DEF - get panel properties
SET-PANEL-DEF - create panel and/or set its properties
DISPLAY-PANEL - display a panel and make it the current panel

CLEAR-PANEL - clear a panel
READ-PANEL - read a panel from current panel file into memory
WRITE-PANEL - write a panel out from memory to current panel file
READ-NEXT-PANEL - get name of next panel in current panel file and read it into memory
DELETE-RECORD - delete panel record from current panel file
COPY-PANEL - copy the current panel in memory

Field

DISPLAY-FIELD - display field
GET-FIELD-DATA - get user input into a field
GET-FIELD-DEF - get field properties
SET-FIELD-DEF - create field and/or set its properties
GET-NEXT-FIELD-DEF - retrieve next field and get its properties
DELETE-FIELD - delete a field
MARK-FIELD - highlight a field or portion of a field

Static field

GET-STATIC-DEF - get static field properties
SET-STATIC-DEF - create static field and/or set its properties
GET-NEXT-STATIC-DEF - retrieve next static field and get its properties
DELETE-STATIC - delete a static field

Group

GET-GROUP-DEF - get group properties
SET-GROUP-DEF - create a group and/or set its properties
GET-NEXT-GROUP-DEF - retrieve next group and get its properties
DELETE-GROUP - delete a group

Repeat group

SET-REPEAT-EXT - set extended repeat group properties
GET-REPEAT-EXT - get extended repeat group properties
GET-REPEAT-DEF - get repeat group properties
SET-REPEAT-DEF - create a repeat group and/or set its properties
DELETE-REPEAT - delete a repeat group

Font

GET-FONT-DEF - get font properties
SET-FONT-DEF - create a font and/or set its properties
QUERY-FONT - allow user to pick a font
WRITE-FONTS - write fonts out to fonts/colors file
DELETE-FONT - delete a font

Color

GET-COLOR-DEF - get color properties
SET-COLOR-DEF - create a color and/or set its properties
QUERY-COLOR - allow user to pick a color
WRITE-COLORS - write colors out to fonts/colors file
DELETE-COLOR - delete a color

Menu

GET-MENU-OPTION - get menu option properties

SET-MENU-OPTION - set menu option properties
INSERT-MENU-OPTION - insert menu option and set its properties
DELETE-MENU-OPTION - delete menu option
CLEAR-MENU - clear Menu Bar
GET-MENU-DEF - get menu properties
SET-MENU-DEF - create a menu and set its properties
READ-MENU - read a menu from the current panel file into memory
WRITE-MENU - write a menu from memory out to the current panel file

Message

DISPLAY-MESSAGE - display message box
SET-MOUSE-SHAPE - set shape for mouse pointer

Clipboard

GET-CLIPBOARD - get clipboard properties
SET-CLIPBOARD - set clipboard properties

Toolbar

OPEN-TOOLBAR - open a toolbar
ACTIVATE-TOOLBAR - activate the toolbar
DEACTIVATE-TOOLBAR - activate the main panel
DISPLAY-TOOLBAR - refresh the toolbar
CLOSE-TOOLBAR - close the toolbar

File

CREATE-FILE - create a new panel file
QUERY-FILE - allow user to pick a file to be opened
SET-ICON-FILE-NAME - set the name of the application icon to be used

Miscellaneous

SET-TEXT-MODE - control text mode screen
SET-KEYBOARD-BUFFER - insert characters to be used as input
GET-VERSION - query runtime version
CHECK-OLD-VERSION - check runtime support for GET-VERSION
RESERVE-MEMORY - reserve memory for a large panel
SET-CONFIGURATION - set configuration variables
EXECUTE-PROGRAM-2 - execute another program
GET-PROPERTY - get object property
SET-PROPERTY - set object property
SET-NOTIFY-ICON - place icon in task bar

Copy File

All the functions and their parameters are defined in a copy file called SP2.CPY. You may include this file in your program as is, or you may edit it to exclude any definitions that you will not be using. If you have a program that is based on the code generated from the editor, the SP2.CPY will automatically be included.

CHAPTER D2 - Primary Functions

OPEN-FILE

The OPEN-FILE function opens the panel file containing the panels to be used or created. A file may be opened in read-only or update mode. In read-only mode, panel definitions may still be updated temporarily (for example, using the SET-PANEL-DEF function) but the updates may not be written back to the file.

This function is also used to make a file that has already been opened, the "current" file. This is the file that the runtime will search first to find a particular panel.

Parameter: FILE-DEF.

Example:

```
MOVE LOW-VALUES TO SP2-FI-DATA.
MOVE "testfile.pan" TO SP2-FI-NAME.
CALL "SP2" USING SP2-OPEN-FILE SP2-FILE-DEF.
```

This opens a file in read-only mode. The file can be opened in read-only mode by other users but may not be opened for update.

CONVERSE-PANEL

The CONVERSE-PANEL function is the simplest way of processing a panel. This function allows:

- a new window to be opened, if necessary;
- a panel definition to be retrieved from the current panel file;
- data to be passed to the fields within the panel;
- the window to be displayed, and
- user input to be accepted.

This function is actually equivalent to the following secondary function calls:

- OPEN-WINDOW (if appropriate);
- READ-PANEL;
- SET-PANEL-FIELDS;
- DISPLAY-WINDOW, and
- GET-INPUT.

Parameter: CONVERSE-DATA.

Example:

```
MOVE LOW-VALUES TO TESTPAN-DATA.
MOVE "TESTPAN" TO TESTPAN-NEXT-PANEL.
MOVE "y" TO TESTPAN-NEW-WINDOW.
MOVE LOW-VALUES TO TESTPAN-FIELDS.
MOVE LOW-VALUES TO TESTPAN-COLRS.
MOVE LOW-VALUES TO TESTPAN-TYPES.
CALL "SP2" USING SP2-CONVERSE-PANEL TESTPAN-CONVERSE-DATA.
MOVE LOW-VALUE TO TESTPAN-NEW-WINDOW.
```

This example:

- initializes the parameter;
- displays panel TESTPAN in a new window;
- accepts user input into the panel, and
- resets the new-window switch to prevent erroneous windows from being created.

CLOSE-WINDOW

The CLOSE-WINDOW function closes the currently active window. All windows should be closed before terminating your system. If you have windows open but not active prior to termination, use ACTIVATE-WINDOW prior to CLOSE-WINDOW.

Parameter: NULL-PARM.

Example:

```
CALL "SP2" USING SP2-CLOSE-WINDOW SP2-NULL-PARM.
```

This closes the current window.

CLOSE-FILE

The CLOSE-FILE function closes the current file. If you have a file open but not current prior to terminating your system, use OPEN-FILE to make that file the current file.

Parameter: NULL-PARM.

Example:

```
CALL "SP2" USING SP2-CLOSE-FILE SP2-NULL-PARM.
```

This closes the current panel file (the file previously referenced by an OPEN-FILE call).

END-SESSION

The END-SESSION function ends the current COBOL sp2 session and releases system resources used by it. This function call must be made prior to terminating your system; otherwise memory may not be released and you may later run out of memory completely.

Parameter: NULL-PARM.

Example:

```
CALL "SP2" USING SP2-END-SESSION SP2-NULL-PARM.
```

This terminates the current session and releases all resources used by it.

SET-RECORD

The SET-RECORD function is used to move a record from program working-storage to memory. It can be used with panel records, menu records, font records and color records. All these records are generated using the Code Generator (see Appendix E). Use of the SET-RECORD function avoids the need for panel files to be distributed and accessed at run time.

Parameter: SP2-RECORD.

Example:

```
CALL "SP2" USING SP2-SET-RECORD MYPANEL-RECORD.
```

This establishes MYPANEL in memory based on the panel definition produced by the generator and included in working-storage.

CHAPTER D3 - Window Functions

OPEN-WINDOW

The OPEN-WINDOW function opens a new window for processing panels. If WD-WIDTH is set to zero (see WINDOW-DEF) and WD-PAN-NAME is set to a valid panel name, the window will be sized to fit the panel definition (a READ-PANEL call will be done automatically).

Parameter: WINDOW-DEF.

Example:

```
MOVE LOW-VALUES TO SP2-WD-DATA.
MOVE "TESTPAN" TO SP2-WD-PANEL-NAME.
CALL "SP2" USING SP2-OPEN-WINDOW SP2-WINDOW-DEF.
```

This opens a new window using the definition of TESTPAN for window properties. The window will not be displayed until a DISPLAY-WINDOW or CONVERSE-PANEL call is made.

DISPLAY-WINDOW

The DISPLAY-WINDOW function displays the current window.

Parameter: NULL-PARM.

Example:

```
CALL "SP2" USING SP2-DISPLAY-WINDOW SP2-NULL-PARM.
```

This displays the current window. This call does not have to be made if you are using CONVERSE-PANEL.

ACTIVATE-WINDOW

The ACTIVATE-WINDOW function makes a previously opened window the current window.

Parameter: NAME-DEF.

Example:

```
MOVE LOW-VALUES TO PANELONE-DATA.
MOVE "PANELONE" TO PANELONE-NEXT-PANEL.
MOVE "y" TO PANELONE-NEW-WINDOW.
MOVE LOW-VALUES TO PANELONE-FIELDS.
CALL "SP2" USING SP2-CONVERSE-PANEL PANELONE-CONVERSE-DATA.
MOVE LOW-VALUE TO PANELONE-NEW-WINDOW.
MOVE LOW-VALUES TO PANELTWO-DATA.
MOVE "PANELTWO" TO PANELTWO-NEXT-PANEL.
MOVE "y" TO PANELTWO-NEW-WINDOW.
MOVE LOW-VALUES TO PANELTWO-FIELDS.
CALL "SP2" USING SP2-CONVERSE-PANEL PANELTWO-CONVERSE-DATA.
MOVE LOW-VALUE TO PANELTWO-NEW-WINDOW.
MOVE "PANELONE" TO SP2-ND-NAME.
CALL "SP2" USING SP2-ACTIVATE-WINDOW SP2-NAME-DEF.
CALL "SP2" USING SP2-CONVERSE-PANEL PANELONE-CONVERSE-DATA.
```

This displays and processes PANELONE in one window, then displays and processes PANELTWO in a second window, then activates the first window and accepts further input to PANELONE.

ACTIVATE-INTERNAL

The ACTIVATE-INTERNAL function activates a window internally without visibly activating it. This is useful if you want to update another window but do not need to accept input into it.

Parameter: NAME-DEF.

Example:

```
MOVE "PANEL002" TO SP2-ND-NAME.
CALL "SP2" USING SP2-ACTIVATE-INTERNAL SP2-NAME-DEF.
MOVE WS-COUNT TO PANEL002-COUNT.
CALL "SP2" USING SP2-DISPLAY-WINDOW SP2-NULL-PARM.
MOVE "PANEL001" TO SP2-ND-NAME.
CALL "SP2" USING SP2-ACTIVATE-INTERNAL SP2-NAME-DEF.
CALL "SP2" USING SP2-CONVERSE-PANEL PANEL001-CONVERSE-DATA.
```

CLEAR-WINDOW

The CLEAR-WINDOW function clears the current window so that the same window can be used for different panels. Fields from a previous panel may be still visible if this is not done.

Parameter: NULL-PARM.

Example:

```
MOVE LOW-VALUES TO PANELONE-DATA.
MOVE "PANELONE" TO PANELONE-NEXT-PANEL.
MOVE "y" TO PANELONE-NEW-WINDOW.
MOVE LOW-VALUES TO PANELONE-FIELDS.
CALL "SP2" USING SP2-CONVERSE-PANEL PANELONE-CONVERSE-DATA.
MOVE LOW-VALUE TO PANELONE-NEW-WINDOW.
CALL "SP2" USING SP2-CLEAR-WINDOW SP2-NULL-PARM.
MOVE LOW-VALUES TO PANELTWO-DATA.
MOVE "PANELTWO" TO PANELTWO-NEXT-PANEL.
MOVE "x" TO PANELTWO-NEW-WINDOW.
MOVE LOW-VALUES TO PANELTWO-FIELDS.
CALL "SP2" USING SP2-CONVERSE-PANEL PANELTWO-CONVERSE-DATA.
```

This displays and processes PANELONE in a new window, then clears the window, then displays and processes PANELTWO in the same window. You should notice the use of "x" in the new-window switch. This positions PANELTWO in the top left corner of the window rather than using the values for PD-TOP-ROW and PD-LEFT-COL.

GET-WINDOW-DEF

The GET-WINDOW-DEF function is used to retrieve a window definition. The key is WD-NAME. GET-WINDOW-DEF should be used prior to a SET-WINDOW-DEF call.

Parameter: WINDOW-DEF.

Example: see SET-WINDOW-DEF.

SET-WINDOW-DEF

The SET-WINDOW-DEF function is used to modify a window definition. The key is WD-NAME. SET-WINDOW-DEF can be used to change the size or position of a window or to set the window title. Use a GET-WINDOW-DEF call to correctly initialize all data items prior to modification.

Parameter: WINDOW-DEF.

Example:

```
MOVE LOW-VALUES TO SP2-WD-DATA.
MOVE "TESTPAN" TO WD-NAME.
CALL "SP2" USING SP2-GET-WINDOW-DEF SP2-WINDOW-DEF.
MOVE "Modified window title" TO SP2-WD-TITLE.
CALL "SP2" USING SP2-SET-WINDOW-DEF SP2-WINDOW-DEF.
```

This resets the title for the TESTPAN window. GET-WINDOW-DEF is used before SET-WINDOW-DEF so the window definition is not altered unintentionally.

DEACTIVATE-WINDOW

The DEACTIVATE-WINDOW function sends the currently active window to the bottom of the internal window stack. This is useful if you want to prevent this window from accidentally becoming the active window again.

Parameter: NULL-PARM.

Example:

```
CALL "SP2" USING SP2-DEACTIVATE-WINDOW SP2-NULL-PARM.
```

BRING-TO-FOREGROUND

The BRING-TO-FOREGROUND function is similar to the ACTIVATE-WINDOW function but it also ensures that the window being activated also becomes the topmost window in front of windows belonging to other processes.

Parameter: NAME-DEF.

Example:

```
MOVE "PANELONE" TO SP2-ND-NAME.
CALL "SP2" USING SP2-BRING-TO-FOREGROUND SP2-NAME-DEF.
```

CHAPTER D4 - Panel Functions

GET-INPUT

The GET-INPUT function accepts input into the current panel.

Parameter: CONVERSE-DATA (only first part is used).

Example:

```
MOVE LOW-VALUES TO SP2-WD-DATA.
MOVE "TESTPAN" TO SP2-WD-PANEL-NAME.
CALL "SP2" USING SP2-OPEN-WINDOW SP2-WINDOW-DEF.
MOVE LOW-VALUES TO TESTPAN-FIELDS.
CALL "SP2" USING SP2-SET-PANEL-FIELDS TESTPAN-FIELDS.
CALL "SP2" USING SP2-DISPLAY-WINDOW SP2-NULL-PARM.
MOVE LOW-VALUES TO TESTPAN-DATA.
MOVE "TESTPAN" TO TESTPAN-NEXT-PANEL.
CALL "SP2" USING SP2-GET-INPUT TESTPAN-CONVERSE-DATA.
```

This is essentially equivalent to a single CONVERSE-PANEL call for TESTPAN.

SET-PANEL-FIELDS

The SET-PANEL-FIELDS function sets a pointer to the program area used to hold field data.

Parameter: xxxxxxxx-FIELDS (see CONVERSE-DATA).

Example:

```
CALL "SP2" USING SP2-SET-PANEL-FIELDS TESTPAN-FIELDS.
```

This sets the panel Fields area for TESTPAN. This area must be maintained while the window containing TESTPAN is open. This processing is usually performed automatically using a CONVERSE-PANEL call.

SET-PANEL-COLORS

The SET-PANEL-COLORS function sets a pointer to the program area used to hold field colors.

Parameter: xxxxxxxx-COLRS (see CONVERSE-DATA).

Example:

```
CALL "SP2" USING SP2-SET-PANEL-COLORS TESTPAN-COLRS.
```

This sets the panel Colors area for TESTPAN. This area must be maintained while the window containing TESTPAN is open. This processing is usually performed automatically using a CONVERSE-PANEL call.

SET-PANEL-TYPES

The SET-PANEL-TYPES function sets a pointer to the program area used to hold field types.
Parameter: xxxxxxxx-TYPES (see CONVERSE-DATA).
Example:

```
CALL "SP2" USING SP2-SET-PANEL-TYPES TESTPAN-TYPES.
```

This sets the panel Types area for TESTPAN. This area must be maintained while the window containing TESTPAN is open. This processing is usually performed automatically using a CONVERSE-PANEL call.

GET-PANEL-DEF

The GET-PANEL-DEF function is used to retrieve panel properties. The key is PD-NAME.

Parameter: PANEL-DEF.

Example:

```
MOVE LOW-VALUES TO SP2-PD-DATA.
MOVE "TESTPAN" TO SP2-PD-NAME.
CALL "SP2" USING SP2-GET-PANEL-DEF SP2-PANEL-DEF.
```

This retrieves the properties for the TESTPAN panel. A window must be opened with TESTPAN in it before this code will work.

SET-PANEL-DEF

The SET-PANEL-DEF function is used to create a panel and/or set its properties. The key is PD-NAME.

Parameter: PANEL-DEF.
Example:

```
MOVE LOW-VALUES TO SP2-PD-DATA.
MOVE "TESTPAN" TO SP2-PD-NAME.
CALL "SP2" USING SP2-GET-PANEL-DEF SP2-PANEL-DEF.
MOVE "n" TO SP2-PD-CURS-SHOW.
CALL "SP2" USING SP2-SET-PANEL-DEF SP2-PANEL-DEF.
```

This retrieves a panel's properties using GET-PANEL-DEF, sets the switch for cursor display to off, then updates the panel properties. If you are modifying a panel's properties you should always call GET-PANEL-DEF first.

DISPLAY-PANEL

The DISPLAY-PANEL function displays a panel and makes this panel the active panel in the window. The panel will automatically be made into the foreground panel in the window. This function should only be used in conjunction with a multi-panel window.

Parameter: NAME-DEF.

Example:

```
MOVE "PANEL001" TO SP2-ND-NAME.
CALL "SP2" USING SP2-DISPLAY-PANEL SP2-NAME-DEF.
```

CLEAR-PANEL

The CLEAR-PANEL function removes a panel from a window. If this panel is also the active panel in the window, another panel will become the new active panel. This function should only be used in conjunction with a multi-panel window.

Parameter: NAME-DEF.

Example:

```
MOVE "PANEL001" TO SP2-ND-NAME.
CALL "SP2" USING SP2-CLEAR-PANEL SP2-NAME-DEF.
```

READ-PANEL

The READ-PANEL function retrieves a panel definition from the current panel file and makes this panel the active panel within the window.

Parameter: NAME-DEF.

Example:

```
MOVE LOW-VALUES TO SP2-WD-DATA.
MOVE 80 TO SP2-WD-WIDTH.
MOVE 25 TO SP2-WD-HEIGHT.
MOVE "workwin" TO SP2-WD-NAME.
MOVE "y" TO SP2-WD-HIDE-SW.
CALL "SP2" USING SP2-OPEN-WINDOW SP2-WINDOW-DEF.
MOVE "TESTPAN" TO SP2-ND-NAME.
CALL "SP2" USING SP2-READ-PANEL SP2-NAME-DEF.
```

This opens a hidden work window and then reads a panel definition ready for further processing.

WRITE-PANEL

The WRITE-PANEL function is used to write the current panel definition out to the current file. The definition in this case includes all fields, static fields, groups, and other panel definition properties:.

Parameter: NULL-PARM.

Example:

```
CALL "SP2" USING SP2-WRITE-PANEL SP2-NULL-PARM.
```

READ-NEXT-PANEL

The READ-NEXT-PANEL function allows you to get the names of all the panels in a file. This is an easy way of applying changes to all panels without using the editor. Initialize the parameter to low-values in order to get the first panel in the file. A non-zero return code signifies the end of the file. Initialize SP2-ND-NAME to high-values rather than low-values to get the panels in the current window rather than the current file.

Parameter: NAME-DEF

Example:

```
CALL "SP2" USING SP2-READ-NEXT-PANEL SP2-NAME-DEF.
IF SP2-ND-RET-CODE = 0
    MOVE LOW-VALUES TO SP2-PD-DATA
    MOVE SP2-ND-NAME TO SP2-PD-NAME
    CALL "SP2" USING SP2-GET-PANEL-DEF SP2-PANEL-DEF
    MOVE X"0B" TO SP2-PD-TEXT-OPTIONS
    CALL "SP2" USING SP2-SET-PANEL-DEF SP2-PANEL-DEF
    CALL "SP2" USING SP2-WRITE-PANEL SP2-NULL-PARM.
```

This code could be used to update all the panels in a file.

DELETE-RECORD

The DELETE-RECORD function is used to delete a record from a panel file. This record could be a panel record or a menu record. If you wish to delete a menu related to a particular panel, the name of the menu is held in the panel Menu property.

Parameter: NAME-DEF

Example:

```
MOVE "PANEL001" TO ND-NAME.
CALL "SP2" USING SP2-DELETE-RECORD SP2-NAME-DEF.
```

COPY-PANEL

The COPY-PANEL function is used to create a new panel definition from the current panel.

Parameter: NAME-DEF.

Example:

```
MOVE "PANELTWO" TO SP2-ND-NAME.
CALL "SP2" USING SP2-COPY-PANEL SP2-NAME-DEF.
```

This makes a copy of the panel in the current window.

CHAPTER D5 - Field Functions

DISPLAY-FIELD

The DISPLAY-FIELD function allows the contents of a single field to be refreshed without refreshing the whole window with a DISPLAY-WINDOW function. This is usually used in conjunction with the GET-INPUT function when the current window contains a panel with a large number of fields. If this is not the case, the CONVERSE-PANEL function should be quite fast enough.

Parameter: FIELD-DEF.

Example:

```
MOVE 123456 TO ACCOUNT-NUMBER.
MOVE LOW-VALUES TO SP2-FD-DATA.
MOVE ACCOUNT-NUMBER-I TO SP2-FD-ID.
CALL "SP2" USING SP2-DISPLAY-FIELD SP2-FIELD-DEF.
CALL "SP2" USING SP2-GET-INPUT ORDER-CONVERSE-DATA.
```

The particular field to be displayed is identified by FD-ID. If this item is zero, FD-NAME will be used. FD-OCCURRENCE will also be used if the field is a member of a repeat group.

If there is no Fields area allocated for the current panel or configuration variable SP2SAV is set to 2 (see Appendix B) or FD-PROG-SPEC is set to X"20", then place the data to be displayed in FD-INITIAL-VAL and set FD-INITIAL-LEN to the length of the data.

FD-OUTPUT may be set to the protection type for the field and FD-COLR to the color code for the field. FD-OUTPUT may also be set to "r" to force an icon field to be refreshed after its source image has been changed.

GET-FIELD-DATA

The get-field-data function allows the contents of a single field to be retrieved. The major benefit here is that a program field area is not required in order to retrieve data from fields. If you are using a multi-select list box, you must use this function to retrieve user selections.

Parameter: FIELD-DEF.

Example:

```
MOVE LOW-VALUES TO SP2-FD-VAR-LENS.
MOVE 2000 TO SP2-FD-VAR-LEN SP2-FD-INITIAL-LEN.
MOVE LOW-VALUES TO SP2-FD-DATA.
MOVE ACCOUNT-NUMBER-I TO SP2-FD-ID.
CALL "SP2" USING SP2-GET-FIELD-DATA SP2-FIELD-DEF.
MOVE SP2-FD-VAR-DATA (1 : SP2-FD-INITIAL-LEN) TO WS-FIELD-DATA.
```

The function takes field-def as a parameter in the same way as the DISPLAY-FIELD function. Set FD-ID or FD-NAME, FD-OCCURRENCE (if appropriate) and FD-INITIAL-LEN and the data will be returned in FD-VAR-DATA. Remember to set FD-INITIAL-LEN to the maximum length of data that your program can accept and the data will be returned in FD-VAR-DATA up to this maximum. FD-INITIAL-LEN will be reset to reflect the length of the data.

If FD-PROG-SPEC is set to x"20" then FD-PROG-NUM holds the number of the first selection to be returned from a multi-select list box. This allows multiple calls to be made to retrieve more than 32k of data. FD-INITIAL-LEN controls how many selections will be returned each time. FD-PROG-NUM is automatically reset to point to the next selection after the current block.

On return from the function call:

FD-PROG-OFF holds the index (0, 1, 2, …) of the selected item in a list box or combo box.

FD-OUTPUT holds the current Protection setting.

GET-FIELD-DEF

The GET-FIELD-DEF function is used to retrieve a field's properties. The key is FD-ID or FD-NAME. If you wish to use FD-NAME, as the key, FD-ID should be set to 0.

Parameter: FIELD-DEF.

Example:

```
MOVE 2000 TO SP2-FD-VAR-LEN.
MOVE LOW—VALUES TO SP2-FD-VAR-LENS.
MOVE LOW-VALUES TO SP2-FD-DATA.
MOVE TESTPAN-NAME-FIELD-I TO SP2-FD-ID.
CALL "SP2" USING SP2-GET-FIELD-DEF SP2-FIELD-DEF.
```

This retrieves the properties of the field named NAME-FIELD in the TESTPAN panel. The field ids are generated in the copy file for TESTPAN. Notice how FD-VAR-LEN and FD-VAR-LENS are initialized: FD-VAR-LEN should be set to the length of FD-VAR-DATA and all the other LEN items should be set to zero. If this is not done variable-length data items may be truncated.

SET-FIELD-DEF

The SET-FIELD-DEF function is used to create a field and/or set its properties. The key is FD-ID or FD-NAME. If you wish to use FD-NAME as the key, FD-ID should be set to 0.

Parameter: FIELD-DEF.

Example:

```
MOVE LOW-VALUES TO SP2-FD-DATA.
MOVE TESTPAN-NAME-FIELD-I TO SP2-FD-ID.
CALL "SP2" USING SP2-GET-FIELD-DEF SP2-FIELD-DEF.
MOVE "n" TO SP2-FD-CURS-SHOW.
CALL "SP2" USING SP2-SET-FIELD-DEF SP2-FIELD-DEF.
```

This retrieves a field's properties using GET-FIELD-DEF, sets the switch for cursor display to off, then updates the field properties. If you are modifying field properties you should always call GET-FIELD-DEF first.

GET-NEXT-FIELD-DEF

The GET-NEXT-FIELD-DEF function allows you to sequentially retrieve fields and their properties.

Parameter: FIELD-DEF

Set FD-RET-CODE to -1 in order to start the retrieval process at the first field. 0 will be returned in FD-RET-CODE until all fields have been retrieved when 1 will be returned.

Example:

```
MOVE -1 TO SP2-FD-RET-CODE.
PERFORM UNTIL SP2-FD-RET-CODE = 1
    MOVE 2000 TO SP2-FD-VAR-LEN
    MOVE LOW-VALUES TO SP2-FD-VAR-LENS
    MOVE LOW-VALUES TO SP2-FD-DATA
    CALL "SP2" USING SP2-GET-NEXT-FIELD-DEF SP2-FIELD-DEF
END-PERFORM.
```

DELETE-FIELD

The DELETE-FIELD function is used to delete a field. The key is FD-ID.

Parameter: FIELD-DEF.

Example:

```
MOVE LOW-VALUES TO SP2-FD-DATA.
MOVE TESTPAN-NAME-FIELD-I TO SP2-FD-ID.
CALL "SP2" USING SP2-DELETE-FIELD SP2-FIELD-DEF.
```

This deletes the name field on the TESTPAN panel.

MARK-FIELD

The MARK-FIELD function highlights a field or portion of a field. This is useful for showing the result of a search function, for example. The following items should be set in the FIELD-DEF parameter:

FD-ID	id of the field to be marked
FD-OCCURRENCE	occurrence of the field if in a repeat group
FD-PROG-OFF	offset (0=1st character) at which the marking is to start
FD-PROG-LEN	number of characters to be marked
FD-COLR	color to be used for marking

If the function is called a second time, the original marking will be
cleared and the new field marked instead. To remove the marking without
marking a new field, call function again with FD-COLR set to low-value.

Parameter: FIELD-DEF.

Example:

```
MOVE LOW-VALUES TO SP2-FD-DATA.
MOVE 1 TO SP2-FD-ID.
MOVE 5 TO SP2-FD-OCCURRENCE.
MOVE 3 TO SP2-FD-PROG-OFF.
MOVE 8 TO SP2-FD-PROG-LEN.
MOVE SP2-COLR-HIGHLIGHT TO SP2-FD-COLR.
CALL "SP2" USING SP2-MARK-FIELD SP2-FIELD-DEF.
```

CHAPTER D6 - Static Field Functions

GET-STATIC-DEF

The GET-STATIC-DEF is used to retrieve a static field's properties. The key is SD-ID or SD-ROW, SD-COL.

Parameter: STATIC-DEF.

Example:

```
MOVE LOW-VALUES TO SP2-SD-DATA.
MOVE 1 TO SP2-SD-ROW.
MOVE 1 TO SP2-SD-COL.
CALL "SP2" USING SP2-GET-STATIC-DEF SP2-STATIC-DEF.
```

This retrieves the properties for the static at row 1, column 1. If you need to access static text on a regular basis it is better to make the static a display-only custom field.

SET-STATIC-DEF

The SET-STATIC-DEF function is used to create a static field and/or set its properties. The key is SD-ID or SD-ROW, SD-COL.

Parameter: STATIC-DEF.

Example:

```
MOVE LOW-VALUES TO SP2-FD-DATA.
MOVE 1 TO SP2-SD-ROW.
MOVE 1 TO SP2-SD-COL.
CALL "SP2" USING SP2-GET-STATIC-DEF SP2-STATIC-DEF.
MOVE "new text" TO SP2-SD-TEXT.
CALL "SP2" USING SP2-SET-STATIC-DEF SP2-STATIC-DEF.
```

This retrieves a static definition using GET-STATIC-DEF, modifies the static text, then updates the static definition. You should normally use a display-only custom field to do this sort of processing.

GET-NEXT-STATIC-DEF

The GET-NEXT-STATIC-DEF function allows you to sequentially retrieve static fields and their properties.

Parameter: STATIC-DEF

Set SD-RET-CODE to -1 in order to start the retrieval process at the first static field. 0 will be returned in SD-RET-CODE until all static fields have been retrieved when 1 will be returned.

Example:

```
MOVE -1 TO SP2-SD-RET-CODE.
PERFORM UNTIL SP2-SD-RET-CODE = 1
    MOVE LOW-VALUES TO SP2-SD-DATA
    CALL "SP2" USING SP2-GET-NEXT-STATIC-DEF SP2-STATIC-DEF
END-PERFORM.
```

DELETE-STATIC

The DELETE-STATIC function is used to delete a static field. The key is SD-ROW and SD-COL or SD-ID.

Parameter: STATIC-DEF.

Example:

```
MOVE LOW-VALUES TO SP2-SD-DATA.
MOVE 1 TO SP2-SD-ROW.
MOVE 1 TO SP2-SD-COL.
CALL "SP2" USING SP2-DELETE-STATIC SP2-STATIC-DEF.
```

CHAPTER D7 - Group Functions

GET-GROUP-DEF

The GET-GROUP-DEF function is used to retrieve a group's properties. The key is GD-ID.

Parameter: GROUP-DEF.

Example:

```
MOVE LOW-VALUES TO SP2-GD-DATA.
MOVE SP2-FD-GROUP-ID TO SP2-GD-ID.
CALL "SP2" USING SP2-GET-GROUP-DEF SP2-GROUP-DEF.
```

This retrieves the properties of a group containing a field whose properties have been previously retrieved using GET-FIELD-DEF.

SET-GROUP-DEF

The SET-GROUP-DEF function is used to create a group and/or set its properties. The key is GD-ID.

Parameter: GROUP-DEF.

Example:

```
MOVE LOW-VALUES TO SP2-GD-DATA.
MOVE SP2-FD-GROUP-ID TO SP2-GD-ID.
CALL "SP2" USING SP2-GET-GROUP-DEF SP2-GROUP-DEF.
MOVE LOW-VALUE TO SP2-GD-SELECT-TYPE.
CALL "SP2" USING SP2-SET-GROUP-DEF SP2-GROUP-DEF.
```

This resets the Select type property which controls the functionality of the group. If you wish to make a change that affects the fields within the group you should delete the group then recreate it.

GET-NEXT-GROUP-DEF

The GET-NEXT-GROUP-DEF function allows you to sequentially retrieve groups and their properties.

Parameter: GROUP-DEF

Set GD-RET-CODE to -1 in order to start the retrieval process at the first group. 0 will be returned in GD-RET-CODE until all groups have been retrieved when 1 will be returned.
Example:

```
MOVE -1 TO SP2-GD-RET-CODE.
PERFORM UNTIL SP2-GD-RET-CODE = 1
    MOVE LOW-VALUES TO SP2-GD-DATA
    CALL "SP2" USING SP2-GET-NEXT-GROUP-DEF SP2-GROUP-DEF
END-PERFORM.
```

DELETE-GROUP

The DELETE-GROUP function is used to delete a group. The key is GD-ID.

Parameter: GROUP-DEF.

Example:

```
MOVE LOW-VALUES TO SP2-GD-DATA.
```

```
    MOVE SP2-FD-GROUP-ID TO SP2-GD-ID.
    CALL "SP2" USING SP2-DELETE-GROUP SP2-GROUP-DEF.
```

This deletes the group containing a field whose properties have been previously retrieved.

CHAPTER D8 - Repeat Group Functions

SET-REPEAT-EXT

The SET-REPEAT-EXT function allows the use of extended repeat processing to display large amounts of data within a repeat group without loading all the data at one time. It also provides an easy way of setting the number of occurrences in the repeat group and the occurrences that are to be displayed.

Parameter: REPEAT-EXT.

Example:

```
MOVE LOW-VALUES TO SP2-RX-DATA.
MOVE 1 TO SP2-RX-ID.
MOVE 3 TO SP2-RX-BLOCK-SW.
MOVE 0 TO SP2-RX-BLOCK-DISP.
MOVE 20 TO SP2-RX-BLOCK-OCCS.
MOVE 1 TO SP2-RX-TOTAL-SW.
MOVE 999 TO SP2-RX-TOTAL-OCCS.
PERFORM PROC-GET-BLOCK THRU PROC-GET-BLOCK-EXIT.
CALL "SP2" USING SP2-SET-REPEAT-EXT SP2-REPEAT-EXT.
```

This initially sets up a repeat group for extended processing. Only 20 records will be passed to the repeat group. The actual number of records is not known at this stage, so an estimate of 999 is made.

GET-REPEAT-EXT

The GET-REPEAT-EXT function allows extended repeat group long properties to be retrieved during processing of long extended repeat groups (more than 32k records).

Parameter: REPEAT-EXT.

Example:

```
MOVE LOW-VALUES TO SP2-RX-DATA.
MOVE 1 TO SP2-RX-ID.
CALL "SP2" USING SP2-GET-REPEAT-EXT SP2-REPEAT-EXT.
PERFORM PROC-GET-BLOCK THRU PROC-GET-BLOCK-EXIT.
```

GET-REPEAT-DEF

The GET-REPEAT-DEF function is used to retrieve repeat group properties. The key is RD-ID.

Parameter: REPEAT-DEF.

Example:

```
MOVE LOW-VALUES TO SP2-RD-DATA.
MOVE SP2-FD-REPEAT-ID TO SP2-RD-ID.
CALL "SP2" USING SP2-GET-REPEAT-DEF SP2-REPEAT-DEF.
```

This retrieves the properties for a repeat group whose properties have been previously retrieved using GET-FIELD-DEF.

SET-REPEAT-DEF

The SET-REPEAT-DEF function is used to create a repeat group and/or set its properties. The key is RD-ID.

Parameter: REPEAT-DEF.

Example:

```
MOVE LOW-VALUES TO SP2-RD-DATA.
MOVE SP2-FD-REPEAT-ID TO SP2-RD-ID.
CALL "SP2" USING SP2-GET-REPEAT-DEF SP2-REPEAT-DEF.
MOVE 200 TO SP2-RD-VERT-OCC.
CALL "SP2" USING SP2-SET-REPEAT-DEF SP2-REPEAT-DEF.
```

This modifies the total number of occurrences in a repeat group. If you wish to modify the visible number of occurrences you must delete the repeat group then recreate it.

DELETE-REPEAT

The DELETE-REPEAT function is used to delete a repeat group. The key is RD-ID.

Parameter: REPEAT-DEF.

Example:

```
MOVE LOW-VALUES TO SP2-RD-DATA.
MOVE SP2-FD-REPEAT-ID TO SP2-RD-ID.
CALL "SP2" USING SP2-DELETE-REPEAT SP2-REPEAT-DEF.
```

This deletes the repeat group containing a field whose properties have been previously retrieved using GET-FIELD-DEF.

CHAPTER D9 - Font Functions

GET-FONT-DEF

The GET-FONT-DEF function is used to retrieve font properties. The key is FO-ID.

Parameter: FONT-DEF.

Example:

```
MOVE LOW-VALUES TO SP2-FO-DATA.
MOVE SP2-FD-FONT-ID TO SP2-FO-ID.
CALL "SP2" USING SP2-GET-FONT-DEF SP2-FONT-DEF.
```

This retrieves the font properties for a field whose properties have been previously retrieved using GET-FIELD-DEF.

SET-FONT-DEF

The SET-FONT-DEF function is used to create a font and/or set its properties. The key is FO-ID.

Parameter: FONT-DEF.

Example:

```
MOVE LOW-VALUES TO SP2-FO-DATA.
MOVE 10 TO SP2-FO-WIDTH.
MOVE 20 TO SP2-FO-HEIGHT.
MOVE "Helv" TO SP2-FO-NAME.
CALL "SP2" USING SP2-SET-FONT-DEF SP2-FONT-DEF.
```

This creates a new Helvetica font. The font id will be placed in SP2-FO-ID upon return from the call.

QUERY-FONT

The QUERY-FONT function displays a Dialog Box that allows the user to pick from one of the available fonts. This function may not be available in all environments so you should always code substitute logic if you are running in multiple environments. If the function is not supported, a code of 1 will be returned in RET-CODE. The advantage of using this function is that your user will be presented with a Dialog Box that he or she is accustomed to seeing when a font needs to be picked.

Parameter: FONT-DEF
The returned font properties can be used with SET-FONT-DEF to actually create a font. FO-HEIGHT should be initialized to 0 - if the user failed to pick a font (cancel was selected), it will still be 0 on return.

Example:

```
MOVE LOW-VALUES TO SP2-FO-DATA.
CALL "SP2" USING SP2-QUERY-FONT SP2-FONT-DEF.
IF SP2-FO-RET-CODE NOT = 0
    PERFORM MY-OWN-QUERY-FONT
IF SP2-FO-HEIGHT NOT = 0
    CALL "SP2" USING SP2-SET-FONT-DEF SP2-FONT-DEF
    MOVE SP2-FO-ID TO SP2-PD-FONT-ID
    CALL "SP2" USING SP2-SET-PANEL-DEF SP2-PANEL-DEF.
```

This code gets the specifications for the font that the user wants to use, then creates the new font, then assigns the font code to the panel Font property so that panel fields will by default use this new font.

WRITE-FONTS

The WRITE-FONTS function is used to write all fonts to the fonts/colors file. This file will be created automatically if it does not exist.

Parameter: NULL-PARM.

Example:

```
CALL "SP2" USING SP2-WRITE-FONTS SP2-NULL-PARM.
```

CHAPTER D10 - Color Functions

GET-COLOR-DEF

The GET-COLOR-DEF function is used to retrieve color properties. The key is CO-ID.

Parameter: COLOR-DEF.

Example:

```
MOVE LOW-VALUES TO SP2-CO-DATA.
MOVE 1 TO SP2-CO-ID
IF SP2-CO-DATA (1 : 1) = LOW-VALUE
    MOVE SP2-FD-COLR TO SP2-CO-DATA (2 : 1)
ELSE
    MOVE SP2-FD-COLR TO SP2-CO-DATA (1 : 1)
END-IF
CALL "SP2" USING SP2-GET-COLOR-DEF SP2-COLOR-DEF.
```

This retrieves color properties for a field whose properties have been previously retrieved GET-FIELD-DEF. Notice the special code to move SP2-FD-COLR, which is a PIC X item containing the color id as a hex number, into SP2-CO-ID, which is a PIC S9(4) COMP item at the start of SP2-CO-DATA.

SET-COLOR-DEF

The SET-COLOR-DEF function is used to create a color and/or set its properties. The key is CO-ID.

Parameter: COLOR-DEF.

Example:

```
MOVE LOW-VALUES TO SP2-CO-DATA.
MOVE "t" TO SP2-CO-TYPE (1).
MOVE X"07" TO SP2-CO-TEXT (1).
MOVE "t" TO SP2-CO-TYPE (2).
MOVE X"01" TO SP2-CO-TEXT (2).
CALL "SP2" USING SP2-SET-COLOR-DEF SP2-COLOR-DEF.
```

This creates a color consisting of the text-mode shades of grey (low-intensity white) on blue. The color id will be placed in SP2-CO-ID upon return from the call.

QUERY-COLOR

The QUERY-COLOR function displays a Dialog Box that allows the user to pick from one of the available colors. This function may not be available in all environments so you should always code substitute logic if you are running in multiple environments. If the function is not supported, a code of 1 will be returned in RET-CODE. The advantage of using this function is that your user will be presented with a Dialog Box that he or she is accustomed to seeing when a color needs to be picked.
Parameter: COLOR-DEF

The returned color properties can be used with SET-COLOR-DEF to actually create the color. The appropriate occurrence of CO-TYPE should be initialized to "?" so that COBOL sp2 knows whether you are asking for a foreground or background color definition - if the user fails to pick a color (cancel was selected), it will still be "?" on return. If you need to get both a foreground and background color, you must make two calls.

Example:

```
MOVE LOW-VALUES TO SP2-CO-DATA.
MOVE "?" TO SP2-CO-TYPE (1).
CALL "SP2" USING SP2-QUERY-COLOR SP2-COLOR-DEF.
```

```
   IF SP2-CO-RET-CODE NOT = 0
       PERFORM MY-OWN-QUERY-COLOR
   ELSE
       IF SP2-CO-TYPE (1) NOT = "?"
           MOVE "?" TO SP2-CO-TYPE (2)
           CALL "SP2" USING SP2-QUERY-COLOR SP2-COLOR-DEF
           IF SP2-CO-TYPE (2) NOT = "?"
               CALL "SP2" USING SP2-SET-COLOR-DEF SP2-COLOR-DEF.
   IF SP2-CO-ID NOT = 0
       IF SP2-CO-DATA (1 : 1) NOT = LOW-VALUE
           MOVE SP2-CO-DATA (1 : 1) TO SP2-PD-COLR
       ELSE
           MOVE SP2-CO-DATA (2 : 1) TO SP2-PD-COLR
       END-IF
       CALL "SP2" USING SP2-SET-PANEL-DEF SP2-PANEL-DEF.
```

This code gets the specifications for the color that the user wants to use, then creates the new color, then assigns the color code to a panel definition so that it will be displayed with the new color. The reference modification code allows the binary color code to be converted to a single byte.

WRITE-COLORS

The WRITE-COLORS function is used to write all colors to the fonts/colors file. This file will be created automatically if it does not exist.

Parameter: NULL-PARM.

Example:

```
   CALL "SP2" USING SP2-WRITE-COLORS SP2-NULL-PARM.
```

CHAPTER D11 - Menu Functions

GET-MENU-OPTION

The GET-MENU-OPTION is used to retrieve the properties of a menu option. The key is MO-ID or MO-NAME if MO-ID is zero. GET-MENU-OPTION should be used prior to a SET-MENU-OPTION call if you need to modify an existing menu option.

Parameter: MENU-OPTION.

Example: see SET-MENU-OPTION.

SET-MENU-OPTION

The SET-MENU-OPTION function is used to add a menu option to a menu and/or set its properties. The key is MO-ID or MO-NAME if MO-ID is zero. If you are modifying a menu option, use GET-MENU-OPTION to retrieve existing properties before resetting them. SET-MENU-OPTION can be used to disable or check a menu option.

Parameter: MENU-OPTION.

Example:

```
MOVE LOW-VALUES TO SP2-MO-DATA.
MOVE TESTPAN-OPT-NEWFILE TO SP2-MO-ID.
PERFORM PROC-GET-MENU-OPTION THRU PROC-GET-MENU-OPTION-EXIT.
MOVE SP2-GREYED-OUT TO MO-STATE.
PERFORM PROC-SET-MENU-OPTION THRU PROC-SET-MENU-OPTION-EXIT.
```

This greys out the New file option in the menu for TESTPAN.

INSERT-MENU-OPTION

The INSERT-MENU-OPTION allows a new menu option to be inserted into an existing menu. This function contrasts with the SET-MENU-OPTION which can only be used to add an option to the END of a menu or submenu. The id of the option being added should be placed in MO-ID. The id of the option before which the new option is to be inserted should be placed in MO-OWNR-ID.

Parameter: MENU-OPTION.

Example:

```
MOVE LOW-VALUES TO SP2-MO-DATA.
MOVE 3 TO SP2-MO-ID.
MOVE 2 TO SP2-MO-OWNR-ID.
MOVE "s" TO SP2-MO-TYPE.
MOVE "New-pulldown" TO SP2-MO-TEXT.
CALL "SP2" USING SP2-INSERT-MENU-OPTION SP2-MENU-OPTION.
```

This inserts a new Menu Bar option in between the first and second options that already exist. Pulldown options can now be added using SET-MENU-OPTION with MO-OWNR-ID set to 3.

DELETE-MENU-OPTION

The DELETE-MENU-OPTION allows an existing menu option to be deleted from a menu. The id of the option being deleted should be placed in MO-ID.

Parameter: MENU-OPTION.

Example:

```
MOVE LOW-VALUES TO SP2-MO-DATA.
MOVE 21 TO SP2-MO-ID.
CALL "SP2" USING SP2-DELETE-MENU-OPTION SP2-MENU-OPTION.
```

CLEAR-MENU

The CLEAR-MENU function allows you to delete the Menu Bar from the current window. CLEAR-WINDOW on its own clears the window, but leaves the Menu Bar intact.

Parameter: NULL-PARM.

Example:

```
CALL "SP2" USING SP2-CLEAR-WINDOW SP2-NULL-PARM.
CALL "SP2" USING SP2-CLEAR-MENU SP2-NULL-PARM.
```

GET-MENU-DEF

The GET-MENU-DEF function is used to retrieve a menu definition. The key is MD-NAME.

Parameter: MENU-DEF.

Example:

```
MOVE LOW-VALUES TO SP2-MD-DATA.
MOVE SP2-WD-MENU-NAME TO SP2-MD-NAME.
CALL "SP2" USING SP2-GET-MENU-DEF SP2-MENU-DEF.
```

This retrieves the menu definition for a window whose properties have been previously retrieved GET-WINDOW-DEF.

SET-MENU-DEF

The SET-MENU-DEF function is used to create a menu and set its properties. The key is MD-NAME.

Parameter: MENU-DEF.

Example:

```
MOVE LOW-VALUES TO SP2-MD-DATA.
MOVE 3 TO SP2-MD-OPTION-CNT.
MOVE "TESTMENU" TO SP2-MD-NAME.
MOVE 1 TO SP2-MDO-ID (1).
MOVE "s" TO SP2-MDO-TYPE (1).
MOVE "~File" TO SP2-MDO-TEXT (1).
MOVE 21 TO SP2-MDO-ID (2).
MOVE 1 TO SP2-MDO-OWNR-ID (2).
MOVE "t" TO SP2-MDO-TYPE (2).
MOVE "~Open" TO SP2-MDO-TEXT (2).
MOVE 22 TO SP2-MDO-ID (3).
MOVE 1 TO SP2-MDO-OWNR-ID (3).
MOVE "t" TO SP2-MDO-TYPE (3).
MOVE "~New" TO SP2-MDO-TEXT (3).
```

This creates a new menu in the current window consisting of a single File pulldown with the Open and New options. Use GET/SET-MENU-OPTION to modify an existing menu.

READ-MENU

The READ-MENU function is used to read a menu from the current panel file into the current window.

Parameter: NAME-DEF.

Example:

```
MOVE "MYMENU" TO SP2-ND-NAME.
CALL "SP2" USING SP2-READ-MENU SP2-NAME-DEF.
```

WRITE-MENU

The WRITE-MENU function is used to write the menu in the current window to the current file.

Parameter: NULL-PARM.

Example:

```
CALL "SP2" USING SP2-WRITE-MENU SP2-NULL-PARM.
```

CHAPTER D12 - Message Functions

DISPLAY-MESSAGE

The DISPLAY-MESSAGE function displays a message box containing text and pushbuttons and waits for a reply from the user.

Parameter: MESSAGE-DATA

Example:

```
MOVE LOW-VALUES TO SP2-MS-DATA.
MOVE 1 TO SP2-MS-LINE-CNT.
MOVE "q" TO SP2-MS-ICON.
MOVE "y" TO SP2-MS-BUTTON.
MOVE "y" TO SP2-MS-CANCEL.
MOVE "File does not exist - create it?" TO SP2-MS-TEXT.
CALL "SP2" USING SP2-DISPLAY-MESSAGE SP2-MESSAGE-DATA.
IF SP2-MS-REPLY = "y"
    PERFORM CREATE-FILE
ELSE
IF SP2-MS-REPLY = "n"
    GOTO GET-FILE-NAME
ELSE
IF SP2-MS-REPLY = "c"
    GOTO GET-FILE-NAME-EXIT.
```

This displays a message box with a question mark icon as well as yes, no and cancel pushbuttons. If the user presses the yes pushbutton, the file is created.

SET-MOUSE-SHAPE

The SET-MOUSE-SHAPE function allows the shape of the mouse to be set to something other than the default. The new shape will be used until another SET-MOUSE-SHAPE function call is made.

Parameter: NULL-PARM.

NP-RET-CODE should be set to one of the following:

0 =	arrow (default)	6 =	wait
1 =	locate	7 =	size bottom right
2 =	size top left	8 =	size bottom
3 =	size top	9 =	size bottom left
4 =	size to right	10 =	size left
5 =	size right		

Example:

```
MOVE 6 TO SP2-NP-RET-CODE.
CALL "SP2" USING SP2-SET-MOUSE-SHAPE SP2-NULL-PARM.
```

SET-MOUSE-FILE

The SET-MOUSE-FILE function allows the shape of the mouse to be set something defined in an external file: ".cur" (static) and ".ani" (animated) files are supported. The new shape will be used until another SET-MOUSE-FILE or SET-MOUSE-SHAPE function call is made.

Parameter: BUFFER.

BF-DATA should be set to the name of the file to be used.

Example:

```
MOVE "\windows\cursors\aero_busy.ani" TO SP2-BF-DATA.
CALL "SP2" USING SP2-SET-MOUSE-FILE SP2-BUFFER.
```

CHAPTER D13 - Clipboard Functions

GET-CLIPBOARD

The GET-CLIPBOARD function allows you to retrieve text from the system clipboard. This function may not be supported in all environments.

Parameter: CLIPBOARD-DATA.

Example:

```
MOVE LOW-VALUES TO SP2-CB-DATA.
CALL "SP2" USING SP2-GET-CLIPBOARD SP2-CLIPBOARD-DATA.
MOVE SP2-CB-TEXT TO WS-TEXT.
```

SET-CLIPBOARD

The SET-CLIPBOARD function allows you to place text on the system clipboard. This function may not be supported in all environments.

Parameter: CLIPBOARD-DATA.

Example:

```
MOVE LOW-VALUES TO SP2-CB-DATA.
MOVE "Hello there" TO SP2-CB-TEXT.
CALL "SP2" USING SP2-SET-CLIPBOARD SP2-CLIPBOARD-DATA.
```

CHAPTER D14 - Toolbar Functions

OPEN-TOOLBAR

The OPEN-TOOLBAR function dynamically creates a toolbar in an existing "toolbar-ready" window. The parameter for this function is WINDOW-DEF with WD-PANEL-NAME set to the name of the toolbar panel or WD-HEIGHT set to the height of the toolbar (in scaled pixels) if the toolbar panel has not yet been created. "Toolbar-ready" means that the existing window must have been opened with the Toolbar rows property (WD-TOOLBAR-ROWS) set to non-zero.

Parameter: WINDOW-DEF

Example:

```
MOVE LOW-VALUES TO SP2-WD-DATA.
MOVE "$tbar001" TO SP2-WD-PANEL-NAME.
CALL "SP2" USING SP2-OPEN-TOOLBAR SP2-WINDOW-DEF.
```

ACTIVATE-TOOLBAR

The ACTIVATE-TOOLBAR function activates the toolbar so that its properties can be set or reset. A toolbar is just a regular panel.

Parameter: NULL-PARM

Example:

```
CALL "SP2" USING SP2-ACTIVATE-TOOLBAR SP2-NULL-PARM.
CALL "SP2" USING SP2-CLEAR-WINDOW SP2-NULL-PARM.
MOVE "$tbar002" TO SP2-ND-NAME.
CALL "SP2" USING SP2-READ-PANEL SP2-NAME-DEF.
```

DEACTIVATE-TOOLBAR

The DEACTIVATE-TOOLBAR function reactivates the main panel after the toolbar has been activated with ACTIVATE-TOOLBAR.

Parameter: NULL-PARM

Example:

```
CALL "SP2" USING SP2-ACTIVATE-TOOLBAR SP2-NULL-PARM.
CALL "SP2" USING SP2-CLEAR-WINDOW SP2-NULL-PARM.
MOVE "$tbar002" TO SP2-ND-NAME.
CALL "SP2" USING SP2-READ-PANEL SP2-NAME-DEF.
CALL "SP2" USING SP2-DEACTIVATE-TOOLBAR SP2-NULL-PARM.
```

DISPLAY-TOOLBAR

The DISPLAY-TOOLBAR function forces a refresh of the toolbar. The contents of a toolbar do not usually change as often as those of the main panel so toolbars are not automatically refreshed when you use DISPLAY-WINDOW or CONVERSE-PANEL for the main panel. If you are using CONVERSE-PANEL, you can also force a toolbar refresh by setting DISPLAY-SW to "t".

Parameter: NULL-PARM

Example:

```
CALL "SP2" USING SP2-ACTIVATE-TOOLBAR SP2-NULL-PARM.
CALL "SP2" USING SP2-CLEAR-WINDOW SP2-NULL-PARM.
MOVE "$tbar002" TO SP2-ND-NAME.
```

```
    CALL "SP2" USING SP2-READ-PANEL SP2-NAME-DEF.
    CALL "SP2" USING SP2-DEACTIVATE-TOOLBAR SP2-NULL-PARM.
    CALL "SP2" USING SP2-DISPLAY-TOOLBAR SP2-NULL-PARM.
```

CLOSE-TOOLBAR

The CLOSE-TOOLBAR functions closes the toolbar and shrinks the overall window. A new toolbar can be opened using OPEN-TOOLBAR.

Parameter: NULL-PARM

Example:

```
    CALL "SP2" USING SP2-CLOSE-TOOLBAR SP2-NULL-PARM.
```

CHAPTER D15 - File Functions

CREATE-FILE

The CREATE-FILE function is used to create a new panel file.

Parameter: FILE-DEF.

Example:

```
MOVE LOW-VALUES TO SP2-FI-DATA.
MOVE "w" TO SP2-FO-MODE.
MOVE "newfile.pan" TO SP2-FI-NAME.
CALL "SP2" USING SP2-CREATE-FILE SP2-FILE-DEF.
```

This creates a new file to be updated.

QUERY-FILE

The QUERY-FILE function displays a Dialog Box that allows the user to pick a file to be opened or a directory to be selected (FI-MODE = "d"). This function may not be available in all environments so you should always code substitute logic if you are running in multiple environments. If the function is not supported, a code of 1 will be returned in RET-CODE. The advantage of using this function is that your user will be presented with a Dialog Box that he or she is accustomed to seeing when a file needs to be opened.

Parameter: FILE-DEF

If you want to display only files of a certain type, you can initialize FI-NAME to a search pattern, for example "*.PAN". The returned file definition parameter could be used with OPEN-FILE to actually open the file if the file was a COBOL sp2 panel file. If the user fails to pick a file (cancel was selected), FI-NAME will be set to low-values.

Example:

```
MOVE LOW-VALUES TO SP2-FI-DATA.
CALL "SP2" USING SP2-QUERY-FILE SP2-FILE-DEF.
IF SP2-FI-RET-CODE NOT = 0
    PERFORM MY-OWN-QUERY-FILE.
IF SP2-FI-NAME NOT = LOW-VALUES
    PERFORM OPEN-FILE.
```

This code gets the name of a file to be opened and then uses regular COBOL code to actually open the file.

SET-ICON-FILE-NAME

The SET-ICON-FILE-NAME function sets the name of the file to be used for the application icon (omit the ".ico"). This overrides the default name which is the same as the first panel file to be opened. The function should be called prior to opening any windows. The function also sets the name of the system help file to be used.

Parameter: NAME-DEF.

Example:

```
MOVE "MYICON" TO SP2-ND-NAME.
CALL "SP2" USING SP2-SET-ICON-FILE-NAME SP2-NAME-DEF.
```

CHAPTER D16 - Miscellaneous Functions

SET-TEXT-MODE

The SET-TEXT-MODE function allows additional control over the screen in text mode and can also be used to detect the runtime environment.

Parameter: NULL-PARM.

The RET-CODE item should be set to the following:

0 =	do nothing (reset RET-CODE only)
1 =	clear screen
2 =	restore SP2 screen
4 =	reset tty to state before SP2 was started (ignored in MSDOS)
8 =	restore SP2 tty settings (ignored in MSDOS)

These values may be added together to perform multiple functions.

On return from the function call, RET-CODE is set to one of the following:

0 =	running in text-mode
1 =	running in Microsoft Windows
2 =	running in OS/2 Presentation Manager
3 =	running in Motif
4 =	running in Thin Client (Windows server)
5 =	running in Thin Client (Unix server)

This function is particularly useful when you need to invoke non-SP2 programs in Unix text mode. Typically you should set RET-CODE to 5 (1 + 4) before invoking another program and then 10 (2 + 8) before resuming normal SP2 operation.

SET-KEYBOARD-BUFFER

The SET-KEYBOARD-BUFFER function inserts keys into the internal keyboard buffer that will be used as input until no more characters remain. This allows an application to be run without active user input. The keys to be inserted should be defined in an occurs clause consisting of numeric key codes (see appendix A). The regular keyboard characters have codes of 32 through 126.

Parameter: BUFFER.

Example:

```
05  WS-KEY-CODES.
    10  FILLER   PIC S9(4) COMP-5 VALUE 65.
    10  FILLER   PIC S9(4) COMP-5 VALUE 66.
    10  FILLER   PIC S9(4) COMP-5 VALUE 67.
    10  FILLER   PIC S9(4) COMP-5 VALUE 13.

MOVE 8 TO SP2-BF-LEN.
MOVE WS-KEY-CODES TO SP2-BF-DATA.
CALL "SP2" USING SP2-SET-KEYBOARD-BUFFER SP2-BUFFER.
```

If the first key slot in bf-data is set to -1, then SET-KEYBOARD-BUFFER will set the system buffer rather than the sp2 internal buffer. This allows you to prime menu bars by loading the buffer up with an F10 key to activate the menu, followed by mnemonic keys to select menu options. Only F1 thru F10 and upper-case letters and numbers are supported. If -1 is put in the second slot as well as the first, a timeout will be generated.

GET-VERSION

The GET-VERSION function returns the runtime version number being used.

Parameter: BUFFER.

See the sample program "SP2VER". You can also use this function to force the generation of a memory dump file ("uib.dbg"). Set BF-TEXT to "DUMPMEM" to do this.

CHECK-OLD-VERSION

The CHECK-OLD-VERSION returns 0 if a version of the runtime is being used that does not support the GET-VERSION call.

Parameter: NULL-PARM.

See the sample program "SP2VER".

RESERVE-MEMORY

The RESERVE-MEMORY function is used to reserve a 64k block of memory in which a panel can grow. Make this call prior to SET-PANEL-DEF or READ-PANEL if you plan to define a large number of fields or a few fields with a lot of initial data. In versions 5 and greater this function does nothing because panel memory is allocated as needed.

Parameter: NULL-PARM.

Example:

```
CALL "SP2" USING SP2-RESERVE-MEMORY SP2-NULL-PARM.
```

SET-CONFIGURATION

The SET-CONFIGURATION function allows you to programatically set and/or override configuration variables normally found in the configuration file or in the system environment (see appendix B). This call should ideally be made before any other calls. BF-DATA may contain any number of configuration variable settings delimited by blanks. If you want to use a delimiter other than blank you can use any character with a value less than blank (tab for example) and pass this character as the first character in the string. Up to 512 bytes may be passed.

Parameter: BUFFER.

Example:

```
MOVE "SP2DIR=.;\sp2 SP2DBG=1" TO SP2-BF-DATA.
CALL "SP2" USING SP2-SET-CONFIGURATION SP2-BUFFER.
```

EXECUTE-PROGRAM-2

The EXECUTE-PROGRAM-2 function executes another program.

Parameter: BUFFER.

Example:

```
MOVE "PBRUSH.EXE" TO SP2-BF-DATA.
CALL "SP2" USING SP2-EXECUTE-PROGRAM-2 SP2-BUFFER.
```

GET-PROPERTY

The GET-PROPERTY function retrieves the value of a property for a panel, field, etc.

Parameter: PROPERTY.

Example:

```
MOVE LOW-VALUES TO SP2-PR-DATA.
MOVE "PC-0000000008" TO SP2-PR-KEY.
CALL "SP2" USING SP2-GET-PROPERTY SP2-PROPERTY.
MOVE SP2-PR-VALUE TO CURRENT-PANEL-NAME.
```

The above retrieves the name of the current panel.

SET-PROPERTY

The SET-PROPERTY function sets the value of a property for a panel, field, etc.

Parameter: PROPERTY.

Example:

```
MOVE LOW-VALUES TO SP2-PR-DATA.
MOVE LISTBOX-I TO SP2-PR-ID.
MOVE "FVC0000000050-RL" TO SP2-PR-KEY.
MOVE MY-LIST-ITEMS TO SP2-PR-VALUE.
CALL "SP2" USING SP2-SET-PROPERTY SP2-PROPERTY.
```

The above sets the items in a list box.

SET-NOTIFY-ICON

The SET-NOTIFY-ICON function places an icon in the notification area of the task bar. Pass parameter sp2-buffer with sp2-bf-data set to the text to appear in the icon hint. The icon itself will be the window icon set by the first panel file opened or the SET-ICON-FILE-NAME function. Use an initial hidden window with border set to "f" if you want to avoid a standard icon on the task bar and then define the main window with border set to "r" so that it's owned by the hidden window. See sample program notifyic for suggested code.

Parameter: BUFFER.

Example:

```
MOVE "Hint to be displayed" TO SP2-BF-DATA.
CALL "SP2" USING SP2-SET-NOTIFY-ICON SP2-BUFFER.
```

EXECUTE-FILE

The EXECUTE-FILE function opens a file using the default program for that type of file. If the file is a program it executes that program. The name of the file is placed in BF-DATA and may be preceded by the string "-w " if you want the call to wait until the new process has completed..

Parameter: BUFFER.

Example:
```
MOVE "MYFILE.DOC" TO SP2-BF-DATA.
CALL "SP2" USING SP2-EXECUTE-FILE SP2-BUFFER.
```

GET-CONFIGURATION

The GET-CONFIGURATION function allows you to programatically retrieve the setting of a configuration variable (see appendix B). BF-DATA should contain the name of a configuration variable and its value will be overwrite the name on return from the call.

Parameter: BUFFER.

Example:

```
MOVE "SP2DIR" TO SP2-BF-DATA.
CALL "SP2" USING SP2-GET-CONFIGURATION SP2-BUFFER.
```

PART E

COBOL sp2 Properties

CHAPTER E1 - Introduction to Properties

Part E describes the properties that control the appearance and behavior of sp2 objects (panels, fields, etc.). Most of these properties are set at design time using the panel editor's properties box. Select the appropriate object and the properties of that object will be automatically displayed in the properties box. Many properties can be set by bringing up a list of possible values. These values are described in detail in this chapter.

Part E also lays out the parameters used at runtime to set these properties programmatically. In most cases the first 2 bytes of a parameter are reserved for a return code. This will be set to 0 unless an error occurs. Following the return code is a group of data items defining the lengths of the actual data to be passed or received. These lengths are very important and should not be changed unless you are specifically instructed to do so. Data items relating to actual properties are grouped together, depending upon whether they are numeric, character or of variable length. There is always a length item (as described above) relating to each group. The length items corresponding to any variable length data items may need to be reset to suit a particular situation.

The following is a list of the property types that are covered here:

- Converse panel properties
- Window properties
- File properties
- Panel properties
- Field properties
- Static field properties
- Group properties
- Repeat group properties
- Font properties
- Color properties
- Menu properties
- Message properties
- Extended repeat properties
- Clipboard properties
- Miscellaneous parameters

CHAPTER E2 - Converse-panel Properties

Overview

These properties are used at runtime in order to control the display and processing of a panel. Some of the properties are used to override actual panel and field properties set at design time.

The properties are set and accessed using data items in the CONVERSE-DATA parameter before and after a CONVERSE-PANEL or GET-INPUT function call. This parameter is contained in an individual copy file generated using the code generator. The version of CONVERSE-DATA included in SP2.CPY should only be used if panels are being defined dynamically.

Parameter Layout

(xxxxxxxx represents the name of a panel):

```
   01  XXXXXXXX-CONVERSE-DATA.
       05  XXXXXXXX-RET-CODE          PIC S9(4) COMP-5.
       05  XXXXXXXX-LENS.
           10  XXXXXXXX-LEN-LEN       PIC S9(4) COMP-5 VALUE +20.
           10  XXXXXXXX-IP-NUM-LEN    PIC S9(4) COMP-5 VALUE +40.
           10  XXXXXXXX-IP-CHAR-LEN   PIC S9(4) COMP-5 VALUE +122.
           10  XXXXXXXX-OP-NUM-LEN    PIC S9(4) COMP-5 VALUE +6.
           10  XXXXXXXX-OP-CHAR-LEN   PIC S9(4) COMP-5 VALUE +2.
           10  XXXXXXXX-FIELD-LEN     PIC S9(4) COMP-5 VALUE +0.
           10  XXXXXXXX-COLR-LEN      PIC S9(4) COMP-5 VALUE +0.
           10  XXXXXXXX-TYPE-LEN      PIC S9(4) COMP-5 VALUE +0.
           10  FILLER                 PIC S9(4) COMP-5 VALUE +0.
           10  FILLER                 PIC S9(4) COMP-5 VALUE +0.
           10  FILLER                 PIC S9(4) COMP-5 VALUE +0.
           10  XXXXXXXX-LENL-LEN      PIC S9(4) COMP-5 VALUE -1.
           10  XXXXXXXX-NUML-LEN      PIC S9(4) COMP-5 VALUE +0.
           10  XXXXXXXX-FIELD-LEN-L   PIC S9(8) COMP-5 VALUE +0.
           10  XXXXXXXX-COLR-LEN-L    PIC S9(8) COMP-5 VALUE +0.
           10  XXXXXXXX-TYPE-LEN-L    PIC S9(8) COMP-5 VALUE +0.
       05  XXXXXXXX-DATA.
  ******** XXXXXXXX-IP-NUM-DATA *****
           10  XXXXXXXX-KEY           PIC S9(4) COMP-5.
           10  XXXXXXXX-NEXT-FLD-ID   PIC S9(4) COMP-5.
           10  XXXXXXXX-NEXT-FLD-NUM  PIC S9(4) COMP-5.
           10  XXXXXXXX-NEXT-TAB-NUM  PIC S9(4) COMP-5.
           10  XXXXXXXX-NEXT-OCCURS   PIC S9(4) COMP-5.
           10  XXXXXXXX-LAST-FLD-ID   PIC S9(4) COMP-5.
           10  XXXXXXXX-LAST-FLD-NUM  PIC S9(4) COMP-5.
           10  XXXXXXXX-LAST-TAB-NUM  PIC S9(4) COMP-5.
           10  XXXXXXXX-LAST-OCCURS   PIC S9(4) COMP-5.
           10  XXXXXXXX-MENU-ID       PIC S9(4) COMP-5.
           10  XXXXXXXX-CTRL-FIELD-KEY REDEFINES XXXXXXXX-MENU-ID
                                      PIC S9(4) COMP-5.
           10  XXXXXXXX-BUTTON-ID REDEFINES XXXXXXXX-MENU-ID
                                      PIC S9(4) COMP-5.
           10  XXXXXXXX-ROW-COL-SW    PIC S9(4) COMP-5.
           10  XXXXXXXX-CURSOR-ROW    PIC S9(4) COMP-5.
           10  XXXXXXXX-CURSOR-COL    PIC S9(4) COMP-5.
           10  XXXXXXXX-LAST-ROW      PIC S9(4) COMP-5.
           10  XXXXXXXX-LAST-COL      PIC S9(4) COMP-5.
           10  XXXXXXXX-DISP-SW       PIC S9(4) COMP-5.
           10  XXXXXXXX-NEXT-VERT     PIC S9(4) COMP-5.
           10  XXXXXXXX-LAST-VERT     PIC S9(4) COMP-5.
           10  XXXXXXXX-NEXT-HOR      PIC S9(4) COMP-5.
```

```
          10  XXXXXXXX-LAST-HOR      PIC S9(4) COMP-5.
 ******** XXXXXXXX-IP-CHAR-DATA ****
          10  XXXXXXXX-NEXT-PANEL    PIC X(8).
          10  XXXXXXXX-NEXT-FIELD    PIC X(30).
          10  XXXXXXXX-LAST-FIELD    PIC X(30).
          10  XXXXXXXX-MENU-OPTION   PIC X(30).
          10  XXXXXXXX-SWITCH-SW     PIC X.
          10  XXXXXXXX-SIZE-SW       PIC X.
          10  XXXXXXXX-MOUSE-SW      PIC X.
          10  XXXXXXXX-CAPTURE-SW    PIC X.
          10  XXXXXXXX-WAIT-SW       PIC X.
          10  XXXXXXXX-CURS-SW       PIC X.
          10  XXXXXXXX-CHG-SW        PIC X.
          10  XXXXXXXX-TIMEOUT       PIC X.
          10  XXXXXXXX-NEXT-SUBPANEL PIC X.
          10  XXXXXXXX-LAST-SUBPANEL PIC X.
 ******** XXXXXXXX-OP-NUM-DATA *****
          10  XXXXXXXX-PAN-POS-SW    PIC S9(4) COMP-5.
          10  XXXXXXXX-PAN-ROW       PIC S9(4) COMP-5.
          10  XXXXXXXX-PAN-COL       PIC S9(4) COMP-5.
 ******** XXXXXXXX-OP-CHAR-DATA ****
          10  XXXXXXXX-NEW-WINDOW    PIC X.
          10  XXXXXXXX-DISPLAY-SW    PIC X.
 ******** XXXXXXXX-FIELDS *******
 ******** XXXXXXXX-COLRS ********
 ******** XXXXXXXX-TYPES ********
```

xxxxxxxx-FIELD-LEN is the length of xxxxxxxx-FIELDS (FIELD-LEN-L if greater than 9999).
xxxxxxxx-COLR-LEN is the length of xxxxxxxx-COLRS (COLR-LEN-L if greater than 9999).
xxxxxxxx-TYPE-LEN is the length of xxxxxxxx-TYPES (TYPE-LEN-L if greater than 9999)..

Control key pressed

Program name : **xxxxxxxx-KEY**

The key or action that caused control to be returned to your program. This will normally be one of the keys assigned to Push Buttons or set up specifically as control keys using the panel Control keys property. See Appendix A for a list of keys and their corresponding values. This property is set by the runtime.

Next field ID

Program name: **xxxxxxxx-NEXT-FLD-ID**

The id of the field where the cursor is to be positioned. The ids assigned to fields are listed in the copy file output by the code generator. Field ids do not change even if other fields are added to a panel. This is the best way to set cursor position. A field id of zero means use the first tab field unless the cursor position is set by some other means (see below). This property is reset by the runtime.

Next field number

Program name: **xxxxxxxx-NEXT-FLD-NUM**

The number of the field where the cursor is to be positioned. Field numbers are assigned left to right, top to bottom and include display-only fields. Field numbers may change if other fields are added to a panel. Next field ID and next tab number must be set to zero if you wish to use this property. This property is updated by the runtime.

Next tab number

Program name: **xxxxxxxx-NEXT-TAB-NUM**

The tab number of the field where the cursor is to be positioned. Tab numbers are assigned left to right, top to bottom not including display-only fields. The default tabbing order may be overridden using the panel Tab number property. Tab numbers may change if other fields are added to a panel. Next field ID must be set to zero if you wish to use this property. This property is updated by the runtime.

Next occurrence

Program name: **xxxxxxxx-NEXT-OCCURS**

The occurrence of the field where the cursor is to be positioned. Only meaningful if Next field ID, Next field number or Next tab number refers to a field in a repeat group. A value of zero means use the first occurrence. This property is updated by the runtime.

Last field id

Program name: **xxxxxxxxx-LAST-FLD-ID**

The id of the field where the cursor was positioned before control was returned. The ids assigned to fields are listed in the copy file output by the code generator. An id relating to a Push Button will not normally be returned. Instead, the id of the last non-Push Button field will be returned. This property is set by the runtime.

Last field number

Program name: **xxxxxxxxx-LAST-FLD-NUM**

The number of the field where the cursor was positioned before control was returned. Field numbers are assigned left to right, top to bottom and include display-only fields. This property is set by the runtime.

Last tab number

Program name **xxxxxxxxx-LAST-TAB-NUMBER**

The tab number of the field where the cursor was positioned before control was returned. Tab numbers are assigned left to right, top to bottom not including display-only fields. This property is set by the runtime.

Last occurrence

Program name: **xxxxxxxxx-LAST-OCCURS**

The occurrence of the field where the cursor was positioned before control was returned. Only meaningful if Last field ID, Last field number or Last tab number refers to a field in a repeat group. Use this item to detect cursor position in a repeat group or detect which row was selected in a repeat group. This property is set by the runtime.

Menu option id

Program name: **xxxxxxxx-MENU-ID**

The id of the menu option that was selected. This property will be set when Control key pressed is set to SP2-KEY-MENU (-6). Menu ids are assigned in the panel editor when defining a menu. This property is set by the runtime.

This property is also set to the actual key pressed when Control key pressed is set to SP2-KEY-CTRL-FIELD (-4) signifying that a "return on exit" field has been exited or SP2-KEY-ENTER-FIELD (-31) signifying that a "return on entry" field has been entered.

This property is also set to the id of a display-only Push Button or Icon if such a Push Button or Icon causes control to be returned. It will also be set to the id of a Push Button if the Push Button's mnemonic (alt+letter) is used.

This property is also set to the id of the toolbar key code when an item on the toolbar is selected (Control key pressed = SP2-KEY-TOOLBAR (-15)). This key code will normally be the key code of one of the toolbar icon fields.

This property is also set to the id of a repeat group if key pressed is set to SP2-KEY-MORE (-4).

If Menu option id is 0, it means that either a key itself was pressed or the Push Button used was not display-only (and therefore Next field id is set to its ID).

Row column switch

Program name: **xxxxxxxx-ROW-COL-SW**

Switch to activate cursor positioning by row and column.

1 = Cursor row and Cursor column represent the position of the cursor within the panel in terms of cells. Actual cursor position will always be at the beginning of a field.
2 = Cursor column is the offset of the cursor within the field in terms of characters. This is the only way to cause the cursor to be positioned other than at the start of a field.
8 = Mouse row and column will be returned in Last row and Last column if the mouse is clicked.

Cursor row

Program name: **xxxxxxxx-CURSOR-ROW**

The row where the cursor is to be positioned in terms of cells (only meaningful if Row column switch is set to 1). This property is updated by the runtime.

Cursor column

Program name: **xxxxxxxx-CURSOR-COL**

The column where the cursor is to be positioned in terms of cells (Row column switch set to 1) or number of characters offset into field (Row column switch set to 2). This property is updated by the runtime.

Last row

Program name: **xxxxxxxx-LAST-ROW**

The row (in terms of cells) of the field where the cursor was positioned before control was returned except:
- if Row column switch is set to 2, not used.
- if Row column switch is set to 8, set to the row where mouse is clicked.
- if Mouse switch is "r" or "o", set to the row where mouse is right-clicked.

This property is set by the runtime.

Last column

Program name: **xxxxxxxx-LAST-COL**

The column (in terms of cells) of the field where the cursor was positioned before control was returned except:
- if Row column switch is set to 2, set to the character offset of the cursor within the field where the cursor was positioned.
- if Row column switch is set to 8, set to the column where mouse is clicked.
- if Mouse switch is "r" or "o", set to the column where mouse is right-clicked.
- if field Options-5 is set to X"01", set to the character offset of the cursor within the field where the cursor was positioned and also controls the position of the cursor on the next call.

This property is set by the runtime.

Displacement switch

Program name: **xxxxxxxx-DISP-SW**

Switch to activate display of a repeat group by displacement or selection of text in an edit field.

1 = Next vertical displacement is the offset in rows of the first row to be displayed.
2 = Next vertical displacement is the offset in characters of the text to be selected.

Next vertical displacement

Program name: **xxxxxxxx-NEXT-VERT**

If Displacement switch is 1, this is the offset in rows of the first row to be displayed in the repeat group where the cursor is to be positioned. If Displacement switch is 2, this is the offset in characters of the text to be selected in the edit field where the cursor is to be positioned. This property is updated by the runtime.

Last vertical displacement

Program name: **xxxxxxxx-LAST-VERT**

If Displacement switch is not 2, this is the offset in terms of rows of the first row displayed in the repeat group where the cursor was positioned before control was returned. If Displacement switch is 2, this is the offset of the text that was selected in the edit field where the cursor was positioned. This property is set by the runtime.

Next horizontal displacement

Program name: **xxxxxxxx-NEXT-HOR**

If Displacement switch is not 2, this is the offset in characters of the first character column to be displayed in the repeat group where the cursor is to be positioned. If Displacement switch is 2, this is the length of the text to be selected in the edit field where the cursor is to be positioned. This property is updated by the runtime.

Last horizontal displacement

Program name: **xxxxxxxx-LAST-HOR**

If the displacement switch is not 2, this is the offset in characters of the first character column displayed in the repeat group where the cursor was positioned before control was returned. If Displacement switch is 2, this is the length of the text that was selected in the edit field where the cursor was positioned. This property is set by the runtime.

Next panel

Program name: **xxxxxxxx-NEXT-PANEL**

The panel to be processed. If this is the name of a container panel, then the actual panel to be displayed is specified by the Next subpanel property. This property is updated by the runtime if the user switches to another window (Switch switch is "y" and Control key pressed is SP2-KEY-SWITCH (-3)).

Next field

Program name: **xxxxxxxx-NEXT-FIELD**

The name of the field where the cursor is to be positioned. Next field ID, Next field number and Next tab number must be 0 if you wish to use this property. This property is updated by the runtime.

Last field

Program name: **xxxxxxxx-LAST-FIELD**

The name of the field where the cursor was positioned before control was returned. This property is set by the runtime.

Switch switch

Program name : **xxxxxxxx-SWITCH-SW**

Switch to allow the user to activate a different window with a mouse click.

y = allow the user to switch to a different window with a mouse click. If this switch is set you must be prepared to respond to Control key pressed set to SP2-KEY-SWITCH (-3). Next panel will be set to the name of the panel in the new window.

x = same as "y" except that if the switch is denied because of errors in the current panel's data, control will be returned to the program with Control key pressed set to SP2-KEY-SWITCH-DENIED (-19).

Size switch

Program name: **xxxxxxxx-SIZE-SW**

Switch to allow you to detect if the window is resized or moved.

y = return if the window is resized or moved. If this switch is set you must be prepared to respond to Control key pressed set to SP2-KEY-SIZE (-9). You may use the GET-WINDOW-DEF function to retrieve the new size or position of the window.

Mouse switch

Program name: **xxxxxxxx-MOUSE-SW**

Switch to allow you to detect if the mouse has been clicked.

y = return if left mouse button is clicked. If this value is used you must be prepared to respond to Control key pressed set to SP2-KEY-MOUSE (-10).
r = return if right mouse button is clicked. If this value is used you must be prepared to respond to Control key pressed set to SP2-KEY-CLICK-RIGHT (-11) or SP2-KEY-DOUBLE-RIGHT (-13).
o = return if left or right mouse button is clicked (as above).

Capture switch

Program name: **xxxxxxxx-CAPTURE-SWITCH**

Switch used to allow you to capture the mouse.

y = capture mouse while within current window
x = capture mouse and do not release even if outside window
n = turn off mouse capture if currently active

When the mouse is captured panel fields do not respond to user input because all mouse input is returned directly to your program.

Wait switch

Program name: **xxxxxxxx-WAIT-SW**

Switch used to override regular control key processing.

n = do not wait for input
1 = wait only for first key pressed
k = if a key has been pressed, process this key as normal. If at any time during this call there is no key outstanding, return with Control key pressed set to SP2-KEY-TIMEOUT (-1). This allows your program to check for an interrupt key during batch-type processing.
s = resume regular input but do not attempt to take the input focus. This allows another application which has been initiated with an SP2-EXECUTE-PROGRAM call to retain the input focus.
c = check if a key has been pressed (as for "k") but don't activate the window (as for "s"). This allows a key check to be made by a task which is to be run in the "background". This value must be set specifically before each CONVERSE-PANEL or GET-INPUT call.
t = the panel will timeout (Control key pressed = SP2-KEY-TIMEOUT (-1)) after a certain period, regardless of user input. The duration of the timeout period is specified by Timeout period (below). If you need to turn the timeout facility off during panel processing, set Timeout period to low-value, otherwise it will remain active until the window is closed.
a = same as t but the panel will only timeout if there is no user activity for the amount of the timeout period.
u = same as t but window is not given focus (as for s and c)
b = same as a but window is not given focus (as for s and c)
2 = cause control to be returned after every character entered in a system entry field. Configuration variable SP2EDT=1 or 2 switch must also be turned on. Set Row column switch to 2 to retrieve the offset of the cursor within the field - this offset is returned in Last column and should be used to set the value of Cursor column before the next converse-panel or get-input call. Note that this setting can only be used with system fields and is different to Wait switch = "1" which immediately returns control for all keys entered but does not trap characters entered into system fields.
h = same as t except that Timeout period is specified in 1/100 seconds rather than seconds

Cursor switch

Program name: **xxxxxxxx-CURS-SW**

Switch to control cursor display.

n = do not display cursor. Even though the cursor is not displayed, cursor positions will still be processed as normal. This overrides the panel and field Cursor display properties.
s = do not scroll just to make the cursor visible.

Change switch

Program name: **xxxxxxxxx-CHG-SW**

Set to "y" by the runtime to indicate that the user has entered data into the current panel. This includes modifying the settings of Radio Buttons, Check Boxes and other fields.

Timeout period

Program name: **xxxxxxxxx-TIMEOUT**

The amount of time in seconds after which the panel is to timeout. This is a hexadecimal value between X"01" and to X"ff" representing number of seconds (1-255). X"00" means turn off the timeout. If Wait switch is set to "h", the time is specified in 1/100 seconds.

Next subpanel

Program name: **xxxxxxxxx-NEXT-SUBPANEL**

The next subpanel to be displayed. Set this to low-values to display the container panel or an external panel.

Last subpanel

Program name: **xxxxxxxxx-NEXT-SUBPANEL**

The subpanel just displayed. This property is set by the runtime.

Panel position switch

Program name: **xxxxxxxxx-PAN-POS-SW**

Switch used to allow repositioning of current panel.

1 = absolute position on screen
2 = offset from original position

Panel row

Program name: **xxxxxxxxx-PAN-ROW**

The new panel row in terms of screen cells which are 4 pixels high. This property overrides the actual panel Row property for as long as the panel is in memory.

Panel column

Program name: **xxxxxxxxx-PAN-COL**

The new panel column in terms of screen cells which are 2 pixels wide. This property overrides the actual panel Column property for as long as the panel is in memory.

New window switch

Program name: **xxxxxxxxx-NEW-WINDOW**

Switch to allow you to open a new window for a panel without using the OPEN-WINDOW function or to use an existing window for a new panel.

y = open new window to fit panel
x = use current window for panel and position panel in top left hand corner of window

Use x when you are using the same window for multiple panel displays. This setting causes the actual panel Row and Column properties to be set to 0.

Display switch

Program name: **xxxxxxxx-DISPLAY-SW**

Switch to control window refresh.

n =	suppress panel refresh. The CONVERSE-PANEL function will essentially function the same as GET-INPUT, except that the Fields area property can be set without using SET-PANEL-FIELDS, etc.
t =	refresh toolbar as well as main panel

Fields area

Program name: **xxxxxxxx-FIELDS**

Data items used to override the Value property of each field in the panel that has its Program length property set to non-zero. If a data item is set to low-values, the Initial value of the field will be displayed, otherwise the value of the data item will be displayed. The runtime will update these data items to reflect data entered into the fields by the user.

Program length is usually derived from the field Format property at design time and the code generator will generate data items based on this. The order of the data items is dependent on the field Program offset property which is usually calculated by the Editor based on the order of fields in the panel and their length.

The format of the data for an item in the Fields area corresponds in most cases to that of the Value property data. The following are exceptions to this rule:

For list box and combo box fields (Control types "l" and "o"), the Fields area item holds the option selected whereas the Value property holds the entire list of options or a list identifier.

For icon fields (Control type " i"), the Fields area item holds the name of the image to be displayed whereas the Value property holds the image file preceded by an 8-byte area of text. If the icon field is a multi-image icon (Miscellaneous property X'20' switch set), the Fields area item is divided into 4 slots each Item length long as follows:

Slot 1 =	icon file (default image)
Slot 2 =	icon file 2 (image to be displayed when the mouse is over the button)
Slot 3 =	icon file 3 (image to be displayed when the button is grayed out)
Slot 4 =	icon hint text (text to be displayed when mouse is over the button)

The Value property for a multi-image icon holds only the first 3 slots, with the icon hint text held in the Message text property.

Colors area

Program name: **xxxxxxxx-COLRS**

Data items used to override the color property of each field in the panel that has its Program number property set to non-zero. If a data item is set to low-values, the color of the field will be unchanged.

The order of the data items is dependent on the field Program number property which is usually calculated by the Editor based on the order of the fields in the panel.

Each data item can be set to a hexadecimal number representing a numeric color code, as follows:

low-value =	no change
x'01' =	window default
x'02' =	highlight
x'03' =	menu
x'0B' + =	user-defined color

Types area

Program name: **xxxxxxxx-TYPES**

Data items used to override the Protection property of each field in the panel that has its Program number property set to non-zero. If a data item is set to low-values, the protection of the field will be unchanged.

The order of the data items is dependent on the field Program number property which is usually calculated by the Editor based on the order of the fields in the panel.

Each data item can be set to one of the following codes:

low-value =	use original field definition
'o' =	make field display-only
'p' =	make field protected
'i' =	make field unprotected
'g' =	make field grayed-out
'h' =	make field hidden
'z' =	if this is a subwindow field, suppress display of subwindow

CHAPTER E3 - Window properties

Overview

These properties define the appearance and behavior of a window. Many of the properties can be derived from related panel properties if the window is opened based on a panel definition. This happens when you make a CONVERSE-DATA call with the New window property set to "y" or an OPEN-WINDOW call with the Width property set to 0 and the Panel name property set to a valid panel name. Such properties are marked as such below.

Window properties are set and accessed programmatically either by using the SET-PROPERTY and GET-PROPERTY functions or by using data items in the WINDOW-DEF parameter before an OPEN-WINDOW or SET-WINDOW-DEF call and after a GET-WINDOW-DEF call. To set a property using the SET-WINDOW DEF call, you would normally make a GET-WINDOW-DEF call first. The WINDOW-DEF parameter is contained in the copy file SP2.CPY.

Parameter layout

```
 01  SP2-WINDOW-DEF.
     05  SP2-WD-RET-CODE         PIC S9(4) COMP-5.
     05  SP2-WD-LENS.
         10  SP2-WD-LEN-LEN      PIC S9(4) COMP-5 VALUE +10.
         10  SP2-WD-NUM-LEN      PIC S9(4) COMP-5 VALUE +36.
         10  SP2-WD-CHAR-LEN     PIC S9(4) COMP-5 VALUE +38.
         10  SP2-WD-VAR-LEN      PIC S9(4) COMP-5 VALUE +80.
         10  SP2-WD-TITLE-LEN    PIC S9(4) COMP-5 VALUE +80.
     05  SP2-WD-DATA.
******** SP2-WD-NUM-DATA ********
         10  SP2-WD-WINDOW-ID    PIC S9(4) COMP-5.
         10  SP2-WD-OWNR-ID      PIC S9(4) COMP-5.
         10  SP2-WD-GUI-ID       PIC S9(4) COMP-5.
         10  SP2-WD-GUI-ID2      PIC S9(4) COMP-5.
         10  SP2-WD-WIDTH        PIC S9(4) COMP-5.
         10  SP2-WD-HEIGHT       PIC S9(4) COMP-5.
         10  SP2-WD-ROW          PIC S9(4) COMP-5.
         10  SP2-WD-COL          PIC S9(4) COMP-5.
         10  SP2-WD-TOT-WIDTH    PIC S9(4) COMP-5.
         10  SP2-WD-TOT-HEIGHT   PIC S9(4) COMP-5.
         10  SP2-WD-HOR-DISP     PIC S9(4) COMP-5.
         10  SP2-WD-VERT-DISP    PIC S9(4) COMP-5.
         10  SP2-WD-TITLE-ROWS   PIC S9(4) COMP-5.
         10  SP2-WD-MENU-ROWS    PIC S9(4) COMP-5.
         10  SP2-WD-MENU-ID      PIC S9(4) COMP-5.
         10  SP2-WD-MENU-ID2     PIC S9(4) COMP-5.
         10  SP2-WD-CELL-WIDTH   PIC S9(4) COMP-5.
         10  SP2-WD-CELL-HEIGHT  PIC S9(4) COMP-5.
         10  SP2-WD-TOOLBAR-ROWS PIC S9(4) COMP-5.
******** SP2-WD-CHAR-DATA *******
         10  SP2-WD-NAME         PIC X(8).
         10  SP2-WD-PANEL-NAME   PIC X(8).
         10  SP2-WD-MENU-NAME    PIC X(8).
         10  SP2-WD-COLR         PIC X.
         10  SP2-WD-BOR-TYPE     PIC X.
         10  SP2-WD-INIT-SW      PIC X.
         10  SP2-WD-PAINT-SW     PIC X.
         10  SP2-WD-OPTS-SW      PIC X.
         10  SP2-WD-HIDE-SW      PIC X.
         10  SP2-WD-SBAR-SW      PIC X.
         10  SP2-WD-NO-TABS-SW   PIC X.
         10  SP2-WD-MORE-OPTIONS PIC X.
         10  SP2-WD-CELL-SIZE    PIC X.
```

```
          10  SP2-WD-DIV-WIDTH    PIC X.
          10  SP2-WD-DIV-HEIGHT   PIC X.
          10  SP2-WD-OPTIONS-3    PIC X.
          10  SP2-WD-SYSTEM-MENU  PIC X.
********* SP2-WD-VAR-DATA *******
          10  SP2-WD-TITLE        PIC X(80).
```

WD-VAR-LEN is the length of WD-TITLE.
WD-TITLE-LEN is the length of WD-TITLE.

Id

Program name: **WD-WINDOW-ID**

The id of the window. This property is automatically set by the runtime on an OPEN-WINDOW call. On a GET-WINDOW-DEF or SET-WINDOW-CALL you can use WD-WINDOW-ID to identify the window whose properties are being accessed as long as WD-NAME is set to low-values.

Owner id

Program name: **WD-OWNR-ID**

The id of the parent window. This property must be set on an OPEN-WINDOW call if you wish to open a child window. After a window is opened, this property cannot be changed.

Set this to -2 to force the new window to be owned by the window that currently has focus.

Set this to -3 to force the new window to become the foreground window if there are no other sp2 windows currently open..

GUI id

Program name: **WD-GUI-ID**

The first part of the handle used by the operating system to identify the window.

GUI id (2)

Program name: **WD-GUI-ID-2**

The second part of the handle used by the operating system to identify the window.

Width

Program name: **WD-WIDTH**
Derived from: **Panel Width property**

The width of the visible portion of the window in terms of cells. This will normally be derived from the panel Width property. This property may change if the user changes the size of the window.

Height

Program name: **WD-HEIGHT**
Derived from: **Panel Height property**

The height of the visible portion of the window in terms of cells. This will normally be derived from the panel Height property. This property may change if the user changes the size of the window.

Row

Program name: **WD-ROW**
Derived from: **Panel Row property**

The position on the screen of the top row of the window in terms of screen cells 4 pixels high. This will normally be derived from the panel ROW property. This property may change if the user repositions the window. Set this value to -9999 to center the window vertically on the screen or -9998 to center the window in the parent or owner window

Column

Program name: **WD-COL**
Derived from: **Panel Column property**

The position on the screen of the left column of the window in terms of screen cells 2 pixels wide. This property may change if the user repositions the window. Set this value to -9999 to center the window horizontally on the screen or -9998 to center the window in the parent or owner window

Total width

Program name: **WD-TOT-WIDTH**
Derived from: **Panel Width or Total width property**

The total width of the window in terms of cells. This property will remain constant even if the user changes the size of the window.

Total height

Program name: **WD-TOT-HEIGHT**
Derived from: **Panel Height or Total height property**

The total height of the window in terms of cells. This property will remain constant even if the user changes the size of the window.

Horizontal displacement

Program name: **WD-HOR-DISP**

The number of cells by which the window contents have been shifted in order to display data which would otherwise be hidden to the right of the window. This property will be set by the runtime when Total width is greater than Width and the user scrolls the window to the right.

Vertical displacement

Program name: **WD-VERT-DISP**

The number of cells by which the window contents have been shifted in order to display data which would otherwise be hidden below the window. This property will be set by the runtime when Total height is greater than Height and the user scrolls the window down.

Title rows

Program name: **WD-TITLE-ROWS**
Derived from: **Panel Title rows property**

The number of rows occupied by the window title. This is set to -1 if the window has a default title bar.

Menu rows

Program name: **WD-MENU-ROWS**
Derived from: **Panel Menu rows property**

The number of rows occupied by the menu. This is set to -1 if the window has a default menu bar.

Menu GUI id

Program name: **WD-MENU-ID**

The first part of the handle used by the operating system to identify the menu.

Menu GUI id (2)

Program name: **WD-MENU-ID-2**

The second part of the handle used by the operating system to identify the menu.

Cell width

Program name:	**WD-CELL-WIDTH**
Derived from:	**Panel Cell width property**

The width of a cell in pixels.

Cell height

Program name:	**WD-CELL-HEIGHT**
Derived from:	**Panel Cell height property**

The height of a cell in pixels.

Toolbar rows

Program name:	**WD-TOOLBAR-ROWS**
Derived from:	**Panel Toolbar property**

The height of the toolbar in terms of pixels. This item is normally derived from the panel Height property for the panel whose name is assigned to the panel Toolbar name property. If this property is set to –1, the window is ready to hold a toolbar but does not hold one currently.

Name

Program name:	**WD-NAME**
Derived from:	**Panel Name property**

The name of the window. On a GET-WINDOW-DEF or SET-WINDOW-CALL you should use WD-NAME to identify the window whose properties are being accessed. On a GET-WINDOW-DEF call, if WD-NAME is low-values and WD-WINDOW-ID is zero, the properties of the current window will be retrieved.

Panel name

Program name:	**WD-PANEL-NAME**

The name of the current panel within the window. On an OPEN-WINDOW call, you may initialize WD-DATA to low-values and set WD-PANEL-NAME to indicate the name of the panel from which the window properties are to be derived. All other items will be ignored except WD-HIDE-SW which may still be used to indicate the visibility of the window.

Menu name

Program name:	**WD-MENU-NAME**
Derived from:	**Panel Menu property**

The name of the menu assigned to the window.

Color

Program name:	**WD-COLR**
Derived from:	**Panel Color property**

The id of the color of the window.

low-value = use default color

Border type

Program name:	**WD-BOR-TYPE**
Derived from:	**Panel Window border type property**

The border type to be used for the window. The following is a list of the valid border types:

m =	main window border
d =	dialog box border

n =	no border
low-value =	default border
p =	single line border
r =	resizable window (owned by main window)
o =	non-owned dialog box
t =	tab window
e =	raised border
s =	sunken border
f =	floating toolbar (skinny title bar)
v =	vertical splitter
h =	horizontal splitter

If Border type is "t" and window is a child window, the window will be displayed as a tab window. The Title property will be used for the text for the tab. The width of the window will automatically be set so that it fits the parent window. If the parent window is a dialog box, the height of the window will be as specified in the Height property, otherwise the height will be automatically set as for the width. This allows push buttons, etc, to be put at the bottom of the parent window if the parent is a dialog box.

All windows are by default owned windows except for Window border type = 'm' and 'o'. Owned windows are not included in the task list and always appear in front of their owner (the previously opened window).

Initial switch

Program name: **WD-INIT-SW**

Used internally but WD-INIT-SW may be set to "y" on a SET-WINDOW-DEF call to force a complete window refresh.

Paint switch

Used internally.

Options

Program name: **WD-OPTS-SW**
Derived from: **Panel Text options, Miscellaneous properties**

Switch used to control various aspects of window behavior:

X'01' - used internally
X'02' - used internally
X'04' - autosize window to fill screen in text mode
X'08' - no title bar in text mode
X'10' - no system menu in text mode
X'20' - ignore cell size in text mode
X'40' - no top border in text mode
X'80' - used internally

Hide switch

Program name: **WD-HIDE-SW**

Switch used to control a window’s visibility.

low-value =	visible.
y =	not visible. A hidden window is used to process panel definitions without displaying them. This type of window will normally remain hidden but can be changed to a regular window by setting WD-HIDE-SW to "n" or "x" at run time.
x =	not visible initially but can be made visible by setting WD-HIDE-SW to low-value.
g =	grayed out.
m =	popup menu (derived from panel Options-3 property). The menu associated with this window will be displayed as a popup rather than a menubar (the actual window will not be displayed at all). When the panel is conversed menu selections will be returned as normal with Control key pressed set to SP2-KEY-MENU (-6) and Menu id set to id of

	option selected. If no option is selected Control key pressed will be SP2-KEY-CLOSE (-5) or SP2-KEY-SWITCH (-3). A window switch will only be allowed if Switch switch is set to "y".
a =	maximized.
i =	minimized.
q =	remove min/max options.
r =	restore min/max options.
n =	change a hidden window to a regular visible window at run time.

Scrollbar switch

Program name: **WD-SBAR-SW**
Derived from: **Panel Options-5 property (Value="r")**

Switch used to control whether Scroll Bars should be displayed if part of the total window is not visible.

low-value =	display Scroll Bars
n =	do not display Scroll Bars
x =	force check to see if Scroll Bars are needed
r =	link vertical window scrollbar to repeat group. Only single-column repeat groups will scroll horizontally - this facility is useful for text entry, etc. To scroll a multi-column repeat group, you have to place the repeat group in a scrolling window and allow it to scroll horizontally (set Total width and height properties to activate window scrolling) while the repeat group itself scrolls only vertically. The problem with this technique is that the vertical scrollbar for the repeat group usually starts off outside the visible portion of the window. You can get around this problem by omitting the repeat group scrollbar and setting the Scrollbar switch property = r - this causes the window vertical scroll bar to control vertical scrolling for the repeat group rather than the window.
s =	scroll window based on new values in Horizontal displacement and Vertical displacement.

No tab fields switch

Program name: **WD-NO-TABS-SW**

Used internally.

More options

Program name: **WD-MORE-OPTIONS**
Derived from: **Panel More options, Options-3, Options-4 properties**

Switch used to control various aspects of window behavior:

X'01' - use 3d effects as the default
X'02' - window Total width and height properties are specified in 1/100 inches
X'04' - automatically resize child window to fit within this window
X'08' - used internally
X'10' - window will not resize less than its original size
X'20' - use statusbar for message line
X'40' - used internally
X'80' - window Row and Column properties are specified in pixels, not 2 by 4 cells

Cell size

Program name: **WD-CELL-SIZE**
Derived from: **Panel Cell size property**

Sets the units used for the window Cell width and height properties.

low-value =	pixels
x'03' =	large stock font
x'04' =	small/bold stock font
x'05' =	small stock font
x'07' =	fixed stock font

Cell width division

Program name:	**WD-DIV-WIDTH**
Derived from:	**Panel Cell width division property**

The horizontal cell division if the Cell size property is not low-value. If the division is 1/2 cell, a field can be positioned on a 1/2 cell boundary and its horizontal position (Column property) will be recorded in 1/2 cells.

X'01' - 1/2 cell divisions
X'02' - 1/4 cell divisions
X'04' - 1/8 cell divisions

Cell height division

Program name:	**WD-DIV-HEIGHT**
Derived from:	**Panel Cell height division property**

The vertical cell division if the Cell size property is not low-value. If the division is 1/2 cell, a field can be positioned on a 1/2 cell boundary and its vertical position (Row property) will be recorded in 1/2 cells.

X'01' - 1/2 cell divisions
X'02' - 1/4 cell divisions
X'04' - 1/8 cell divisions
X'08' - 1/16 cell divisions

Options-3

Program name:	**WD-OPTIONS-3**
Derived from:	**Panel Options-3 property**

Switch used to control various aspects of window behavior:

X'01' -	window help. A question mark will be placed in the titlebar of the window. If this question mark is clicked, the mouse pointer will become a question mark and you can click on a field to bring up field-level help (panel-level if no field-level help available). This feature only works for dialogboxes and non-custom fields.
X'02' -	clip children. Prevents the panel background overwriting system fields and child windows. This option should be set for any window containing Activex controls.
X'04' -	toolbar sunken window. Display the window beneath a toolbar with a sunken border.
X'08' -	used internally to track window scrollbars.
X'10' -	used internally to track window scrollbars.
X'20' -	used internally.
X'40' -	window contains anchor fields.

System menu

Program name:	**WD-SYSTEM-MENU**

Switch used to suppress items on the system menu:

X'01' -	suppress Restore
X'02' -	suppress Move
X'04' -	suppress Size
X'08' -	suppress Minimize
X'10' -	suppress Maximize
X'20' -	suppress Close button

Title

Program name:	**WD-TITLE**
Derived from:	**Panel Title property**

The text to be displayed in the title bar.

CHAPTER E4 - Open File Properties

Overview

These properties are used at runtime in order to control access to panel files.

The properties are set using data items in the FILE-DEF parameter on an OPEN-FILE or NEW-FILE call. This parameter is contained in the copy file SP2.CPY.

The FILE-DEF parameter is also used on a QUERY-FILE function.

Parameter layout

```
  01  SP2-FILE-DEF.
      05  SP2-FI-RET-CODE         PIC S9(4) COMP-5.
      05  SP2-FI-LENS.
          10  SP2-FI-LEN-LEN      PIC S9(4) COMP-5 VALUE +10.
          10  SP2-FI-NUM-LEN      PIC S9(4) COMP-5 VALUE +0.
          10  SP2-FI-CHAR-LEN     PIC S9(4) COMP-5 VALUE +2.
          10  SP2-FI-VAR-LEN      PIC S9(4) COMP-5 VALUE +80.
          10  SP2-FI-NAME-LEN     PIC S9(4) COMP-5 VALUE +80.
      05  SP2-FI-DATA.
 ******** SP2-FI-CHAR-DATA *******
          10  SP2-FI-MODE         PIC X.
          10  SP2-FI-SHARE        PIC X.
 ******** SP2-FI-VAR-DATA ********
          10  SP2-FI-NAME         PIC X(80).
```

FI-VAR-LEN is the length of FI-NAME.
FI-NAME-LEN is the length of FI-NAME.

Access mode

Program name: **FI-MODE**

Specifies how the file is to be accessed.

low-value =	read-only
w =	read/write

On a QUERY-FILE function call, FI-MODE is set as follows:

n =	window title is "New"
a =	window title is "Save as"
t =	window title is taken from second half of FI-NAME
d =	display list of directories rather than files. Additional title text or special instructions for the user can be placed in the second half of FI-NAME
s =	push button text is "Save" rather than "Open". Title is taken from second half of FI-NAME as for "t".

Share switch

Program name: **FI-SHARE**

Specifies how the file will be shared between processes.

low-value =	allow any number of users read-only access or one user read/write access (but not both)
y =	allow any number of users full access

If another process already has or may need read/write access, you must set this property even if you only need read-only access. However, you should avoid setting it if nobody will require read/write access because there are performance penalties involved in sharing the file for possible updating. The panel editor opens panel files with Share switch set to low-value unless you have the network version.

On a QUERY-FILE function call, FI-SHARE is set as follows (values may be combined):

X'01' =	return short file name (8.3 format)
X'02' =	do not change current directory
X'04' =	FI-NAME holds initial selection rather than search string
X'08' =	allow multiple files to be selected. Selections are returned in FI-NAME with directory first followed by a low-value, then first file followed by a low-value, then second file followed by a low-value and so on. Use the UNSTRING command delimited by low-value to split out the names.
X'10' =	force current folder to be listed rather than the folder last listed.

File name

Program name: **FI-NAME**

The name of the panel file to be accessed.

On a QUERY-FILE function call, FI-NAME returns the name of the file or directory selected. It may also be initialized to a search string that controls the list of files to be displayed. If FI-MODE is set to "t", the second half of FI-NAME must contain a custom title for the window. If FI-SHARE is set to X'04', FI-NAME must contain the initial file selection rather than a search string.

CHAPTER E5 - Panel Properties

Overview

These properties define the appearance and behavior of a panel.

Panel properties are set and accessed programmatically either by using the SET-PROPERTY and GET-PROPERTY functions or by using data items in the PANEL-DEF parameter before a SET-PANEL-DEF call and after a GET-PANEL-DEF call. To set a property using SET-PANEL-DEF after a panel is already defined you would normally make a GET-PANEL-DEF call followed by a SET-PANEL-DEF call. The PANEL-DEF parameter is contained in the copy file SP2.CPY.

Parameter layout

```
 01  SP2-PANEL-DEF.
     05  SP2-PD-RET-CODE         PIC S9(4) COMP-5.
     05  SP2-PD-LENS.
         10  SP2-PD-LEN-LEN      PIC S9(4) COMP-5 VALUE +30.
         10  SP2-PD-NUM-LEN      PIC S9(4) COMP-5 VALUE +48.
         10  SP2-PD-CHAR-LEN     PIC S9(4) COMP-5 VALUE +76.
         10  SP2-PD-VAR-LEN      PIC S9(4) COMP-5 VALUE +132.
         10  SP2-PD-DESC-LEN     PIC S9(4) COMP-5 VALUE +20.
         10  SP2-PD-TITLE-LEN    PIC S9(4) COMP-5 VALUE +20.
         10  SP2-PD-CURS-KEY-LEN PIC S9(4) COMP-5 VALUE +50.
         10  SP2-PD-CTRL-KEY-LEN PIC S9(4) COMP-5 VALUE +40.
         10  FILLER              PIC S9(4) COMP-5 VALUE +0.
         10  SP2-PD-MSG-TEXT-LEN PIC S9(4) COMP-5 VALUE +1.
         10  SP2-PD-USER-LEN     PIC S9(4) COMP-5 VALUE +1.
         10  SP2-PD-HELP-LEN     PIC S9(4) COMP-5 VALUE +1.
         10  FILLER              PIC S9(4) COMP-5.
         10  SP2-PD-LENL-LEN     PIC S9(4) COMP-5 VALUE -1.
         10  SP2-PD-NUML-LEN     PIC S9(4) COMP-5 VALUE +40.
     05  SP2-PD-DATA.
******** SP2-PD-NUM-DATA ********
         10  SP2-PD-WIDTH        PIC S9(4) COMP-5.
         10  SP2-PD-HEIGHT       PIC S9(4) COMP-5.
         10  SP2-PD-ROW          PIC S9(4) COMP-5.
         10  SP2-PD-COL          PIC S9(4) COMP-5.
         10  SP2-PD-FLD-CNT      PIC S9(4) COMP-5.
         10  SP2-PD-PROG-CNT     PIC S9(4) COMP-5.
         10  SP2-PD-PROG-LEN     PIC S9(4) COMP-5.
         10  SP2-PD-HELP-KEY     PIC S9(4) COMP-5.
         10  SP2-PD-EDIT-OV-KEY  PIC S9(4) COMP-5.
         10  SP2-PD-MSG-REFRESH  PIC S9(4) COMP-5.
         10  SP2-PD-TITLE-ROWS   PIC S9(4) COMP-5.
         10  SP2-PD-DEF-PB       PIC S9(4) COMP-5.
         10  FILLER              PIC S9(4) COMP-5.
         10  SP2-PD-MENU-ROWS    PIC S9(4) COMP-5.
         10  SP2-PD-TOT-WIDTH    PIC S9(4) COMP-5.
         10  SP2-PD-TOT-HEIGHT   PIC S9(4) COMP-5.
         10  SP2-PD-MSG-LEN      PIC S9(4) COMP-5.
         10  SP2-PD-CELL-WIDTH   PIC S9(4) COMP-5.
         10  SP2-PD-CELL-HEIGHT  PIC S9(4) COMP-5.
         10  SP2-PD-FONT-ID      PIC S9(4) COMP-5.
         10  SP2-PD-PROG-LEN-L   PIC S9(8) COMP-5.
         10  SP2-PD-PROG-CNT-L   PIC S9(8) COMP-5.
******** SP2-PD-CHAR-DATA *******
         10  SP2-PD-NAME         PIC X(8).
         10  SP2-PD-MENU-NAME    PIC X(8).
         10  FILLER              PIC X.
```

```
          10  SP2-PD-CUR-FLD-COLR PIC X.
          10  SP2-PD-CURS-SKIP    PIC X.
          10  SP2-PD-CURS-SHOW    PIC X.
          10  SP2-PD-CURS-IN-FLD  PIC X.
          10  SP2-PD-SHIFT-NUMS   PIC X.
          10  SP2-PD-BLANK-NUMS   PIC X.
          10  SP2-PD-ASSUME-DEC   PIC X.
          10  SP2-PD-FORMAT-NUMS  PIC X.
          10  SP2-PD-CURS-WRAP    PIC X.
          10  SP2-PD-INIT-NUMS    PIC X.
          10  SP2-PD-OVERRIDE-REQ PIC X.
          10  SP2-PD-CELL-SIZE    PIC X.
          10  FILLER              PIC X.
          10  SP2-PD-MISC-OPTIONS PIC X.
          10  SP2-PD-DIV-WIDTH    PIC X.
          10  SP2-PD-DIV-HEIGHT   PIC X.
          10  FILLER              PIC X.
          10  FILLER              PIC X.
          10  FILLER              PIC X.
          10  FILLER              PIC X.
          10  FILLER              PIC X.
          10  SP2-PD-COLR         PIC X.
          10  SP2-PD-PROG-DATE    PIC X.
          10  SP2-PD-HELP         PIC X(8).
          10  SP2-PD-TEXT-OPTIONS PIC X.
          10  FILLER              PIC X(8).
          10  SP2-PD-TOOLBAR-NAME PIC X(8).
          10  SP2-PD-MSG-COLR     PIC X.
          10  SP2-PD-MORE-OPTIONS PIC X.
          10  SP2-PD-OPTIONS-3    PIC X.
          10  SP2-PD-OPTIONS-4    PIC X.
          10  SP2-PD-OPTIONS-5    PIC X.
          10  SP2-PD-TAB-OPTIONS  PIC X.
          10  SP2-PD-OPTIONS-6    PIC X.
          10  SP2-PD-OPTIONS-7    PIC X.
          10  SP2-PD-SYSTEM-MENU  PIC X.
          10  SP2-PD-WIN-BOR-TYPE PIC X.
          10  SP2-PD-INITIAL-SW   PIC X.
 ******** SP2-PD-VAR-DATA ********
          10  SP2-PD-DESCRIPTION  PIC X(20).
          10  SP2-PD-TITLE        PIC X(20).
          10  SP2-PD-CURS-KEYS.
              15  SP2-PD-LEFT     PIC S9(4) COMP-5.
              15  SP2-PD-RIGHT    PIC S9(4) COMP-5.
              15  SP2-PD-UP       PIC S9(4) COMP-5.
              15  SP2-PD-DOWN     PIC S9(4) COMP-5.
              15  SP2-PD-TAB      PIC S9(4) COMP-5.
              15  SP2-PD-BACKTAB  PIC S9(4) COMP-5.
              15  SP2-PD-TB-ERASE PIC S9(4) COMP-5.
              15  SP2-PD-BT-ERASE PIC S9(4) COMP-5.
              15  SP2-PD-DELETE   PIC S9(4) COMP-5.
              15  SP2-PD-BACKSPAC PIC S9(4) COMP-5.
              15  SP2-PD-ERASE    PIC S9(4) COMP-5.
              15  SP2-PD-INSERT   PIC S9(4) COMP-5.
              15  SP2-PD-HOME     PIC S9(4) COMP-5.
              15  SP2-PD-END      PIC S9(4) COMP-5.
              15  SP2-PD-SCRL-UP  PIC S9(4) COMP-5.
              15  SP2-PD-SCRL-DN  PIC S9(4) COMP-5.
              15  SP2-PD-SCRL-LT  PIC S9(4) COMP-5.
              15  SP2-PD-SCRL-RT  PIC S9(4) COMP-5.
```

```
            15  FILLER          PIC S9(4) COMP-5.
            15  FILLER          PIC S9(4) COMP-5.
            15  FILLER          PIC S9(4) COMP-5.
            15  FILLER          PIC S9(4) COMP-5.
            15  FILLER          PIC S9(4) COMP-5.
            15  SP2-PD-HOME-PAN PIC S9(4) COMP-5.
            15  SP2-PD-END-PAN  PIC S9(4) COMP-5.
        10  SP2-PD-CTRL-KEYS.
            15  SP2-PD-CTRL-KEY OCCURS 20
                                PIC S9(4) COMP-5.
        10  SP2-PD-MSG-TEXT     PIC X.
        10  SP2-PD-USER-DATA    PIC X.
        10  SP2-PD-HELP-KEYWORD PIC X.
```

PD-VAR-LEN is the sum of the lengths of PD-DESCRIPTION, PD-TITLE, PD-CURS-KEYS, PD-CTRL-KEYS, PD-MSG-TEXT, PD-USER-DATA and PD-HELP-KEYWORD.
PD-DESC-LEN is the length of PD-DESCRIPTION.
PD-TITLE-LEN is the length of PD-TITLE.
PD-CURS-KEY-LEN is the length of PD-CURS-KEYS.
PD-CTRL-KEY-LEN is the length of PD-CTRL-KEYS.
PD-MSG-TEXT-LEN is the length of PD-MSG-TEXT
PD-USER-LEN is the length of PD-USER-DATA
PD-HELP-LEN is the length of PD-HELP-KEYWORD

Width

Program name:	**PD-WIDTH**
Used to derive:	**Window Width property**

The width of the panel in cells. On a GET-PANEL-DEF call, set this to -1 to retrieve the panel details from any window.

Height

Program name:	**PD-HEIGHT**
Used to derive:	**Window Height property**

The height of the panel in cells.

Row

Program name:	**PD-ROW**
Used to derive:	**Window Row property**

The position of the top row of the panel either on the screen or within the current window. On an OPEN-WINDOW or CONVERSE-PANEL call when a new window is being derived from a panel, this property represents the position of the panel on the screen in terms of screen cells 4 pixels high. Set this value to -9999 to center the window vertically on the screen or -9998 to center the window in the parent or owner window. If a new window is not being created, it represents the position of the panel within the current window in terms of window cells.

Column

Program name:	**PD-COL**
Used to derive:	**Window Column property**

The position of the left column of the panel either on the screen or within the current window. On an OPEN-WINDOW or CONVERSE-PANEL call when a new window is being derived from a panel, this property represents the position of the panel on the screen in terms of screen cells 2 pixels wide. Set this value to -9999 to center the window horizontally on the screen or -9998 to center the window in the parent or owner window. If a new window is not being created, it represents the position of the panel within the current window in terms of window cells.

Field count

Program name: **PD-FLD-CNT**

The count of visible fields on the panel. This property is maintained automatically as fields are added and deleted.

Program field count

Program name: **PD-PROG-CNT**

The count of fields accessible by your program if this count is not greater than 9999 (otherwise Long program field count is used). This property is maintained automatically as fields are added and deleted.

Program field length

Program name: **PD-PROG-LEN**

The length of fields accessible by your program if this length is not greater than 9999 (otherwise Long program field length is used). This property is maintained automatically as fields are added and deleted.

Help key

Program name: **PD-HELP-KEY**

The key that the user must press to invoke panel-level help. This help information could be in the form of standard system help or internal help. See chapter B32 for more details of this.

Edit override key

Program name: **PD-EDIT-OV-KEY**

The key used to exit from the panel regardless of any errors detected in the data entered.

Message refresh switch

Program name: **PD-MSG-REFRESH**

Switch to cause the message line to be refreshed (set switch to zero). Only used at runtime.

Title rows

Program name: **PD-TITLE-ROWS**
Used to derive: **Window Title rows property**

The number of rows occupied by the window title. Set to -1 if the window for this panel has a default title bar. This property is automatically set in the panel editor when you set the panel Title property.

Default pushbutton

Program name: **PD-DEF-PB**

The id of the default pushbutton. This property is automatically set when a default pushbutton field is set up.

Menu rows

Program name: **PD-MENU-ROWS**
Used to derive: **Window Menu rows property**

The number of rows occupied by the window menu. Set to -1 if the window for this panel has a default menu bar. This property is automatically set in the panel editor when you set the panel Menu property.

Total width

Program name: **PD-TOT-WIDTH**
Used to derive: **Window Total width property**

The total width of the panel in cells. If this is larger than Width, the window derived from the panel will scroll horizontally to allow the whole panel to be viewed.

Total height

Program name: **PD-TOT-HEIGHT**
Used to derive: **Window Total height property**

The total height of the panel in cells. If this is larger than Height, the window derived from the panel will scroll vertically to allow the whole panel to be viewed.

Message length

Program name: **PD-MSG-LEN**

Switch to indicate that a message line is to be displayed at the bottom of the panel window

-1 =	regular message line.
-2 =	display text in statusbar of parent window (see also Options-3 property).

Cell width

Program name: **PD-CELL-WIDTH**
Used to derive: **Window Cell width property**

The width of a cell in pixels.

Cell height

Program name: **PD-CELL-HEIGHT**
Used to derive: **Window Cell height property**

The height of a cell in pixels.

Font ID

Program name: **PD-FONT-ID**

The id of the font to be used for text in the panel (unless overridden at the field level). Fonts can be established in the panel editor or by using the SET-FONT-DEF function. Zero means use the default font.

Long program field length

Program name: **PD-PROG-LEN-L**

The length of fields accessible by your program if this length exceeds 9999. This property is maintained automatically as fields are added and deleted.

Long program field count

Program name: **PD-PROG-CNT-L**

The count of fields accessible by your program if this count exceeds 9999. This property is maintained automatically as fields are added and deleted.

Name

Program name: **PD-NAME**
Used to derive: **Window Name property**

The name of the panel. Set in the panel editor using the File/Save or File/Save as menu bar option. If a panel is assigned a name that cannot be used as part of a COBOL data name, a warning message is issued. Set PD-NAME before a SET-PANEL-DEF or GET-PANEL-DEF call to indicate which panel is to be accessed.

Menu

Program name: **PD-MENU-NAME**
Used to derive: **Window Menu property**

The name of the menu to be displayed in the window containing the panel. This property can be set in the panel editor directly or indirectly by defining a new menu by clicking the menu bar icon.

Current field color

Program name: **PD-CUR-FLD-COLR**

The id of the color to be used to highlight the field currently set up to receive user input. Colors can be established in the panel editor or by using the SET-COLOR-DEF function. Low-value means no special color for current field.

Cursor skip

Program name: **PD-CURS-SKIP**

Used to control whether the cursor should automatically jump to the next field when no more input is possible in the current field. To enable this feature for system entry fields, configuration variable SP2EDT must be set to 1 or 2 (see Appendix B).

low-value =	jump to next field
n =	do not jump to next field

Cursor show

Program name: **PD-CURS-SHOW**

Used to control the appearance of the cursor while in custom fields within the panel.

low-value =	underline cursor
n =	no cursor
b =	block cursor
v =	vertical (I-beam) cursor

Cursor movement

Program name: **PD-CURS-IN-FLD**

Used to control movement of cursor between fields in the panel.

low-value =	normal movement
v =	allow up/down cursor keys to be used to move vertically between fields. This means that the up/down cursor keys cannot be used to move within a group.

Shift numerics

Program name: **PD-SHIFT-NUMS**

Used to control whether numerics should be shifted left to make input into large numeric fields easier. This data item only applies to custom fields. This option also applies to non-numeric input fields if those fields have their Justify property set to "r" (right) or "c" (center).

low-value =	do not shift
y =	shift left

Blank numerics

Program name: **PD-BLANK-NUMS**

Used to control whether numeric fields should be blanked out if an entry is made into the first position of that field before any other positions. This allows numeric fields to be easily changed. This property only applies to custom fields.

low-value =	blank out numeric if first position entered

n =	do not blank out numeric

Assume decimal

Program name: **PD-ASSUME-DEC**

Used to control whether a decimal point should be assumed if it is not specifically entered in a numeric field.

low-value =	do not assume decimal
y =	assume decimal

Format numerics

Program name: **PD-FORMAT-NUMS**

Used to control whether values entered into numeric fields should be reformatted to a COBOL de-edited numeric item with an assumed decimal point and sign combined with the last digit.

Low-value =	reformat numerics
n =	do not reformat

Cursor wrap

Program name: **PD-CURS-WRAP**

Used to control the behavior of the cursor when the user tries to tab out of the last field or backtab out of the first field.

low-value =	cursor wraps to the first or last field
n =	cursor stays in the current field

Initialize numerics

Program name: **PD-INIT-NUMS**

Used to control whether numeric fields are initialized to zero. If they are not initialized, numeric fields may return a non-numeric value.

low-value =	initialize numerics
n =	numeric field will be initialized to blanks. Blanks will also be treated as valid keyboard input

Btab overrides reqd.

Program name: **PD-OVERRIDE-REQ**

Used to control required field processing.

low-value =	no overrides, so all required fields must be entered unless the Edit override key is hit.
c =	same as Low value
b =	allow Backtab out of current field even if field is empty
a =	same as b

Cell size

Program name: **PD-CELL-SIZE**
Used to derive: **Window Cell size property**

Sets the units used for cell size.

low-value =	pixels
x'03' -	large stock font
x'04' -	small/bold stock font
x'05' -	small stock font
x'07' -	fixed stock font

Miscellaneous

Program name: **PD-MISC-OPTIONS**
Used to derive: **Window Options property**

Switch used to control various aspects of panel behavior. The following values are combined to form the final value:

X'01' -	not used
X'02' **Ignore edits** -	ignore edits and input.
X'04' **Gui - no current color**-	use current field color only in text mode.
X'08' **Text - ignore cell size**-	ignore cell size in text mode. This switch allows a little more flexibility when handling special cell sizes in text mode. Normally, for example, if a field was defined in graphical mode to be positioned on the third row of a panel with a cell height of 24, then that field would be positioned on the fourth row of the panel in text mode. This is because text mode cells are assumed to have a height of 16, the field is supposed to start at pixel row 48 (2 * 24), and 48 / 16 equals row 3 (the fourth row). This behavior may not be desirable and the cell size switch alters this behavior. If the switch is set, the cell size of a panel will always be set to 16 by 8 in text mode - if a field is positioned on the third row in graphical mode, it will be positioned on the third row in text mode.
X'10' **Text - no top border** -	allow title bar to replace the top window border in text mode. This is useful for saving space in text mode.
X'20' **System color for customs** -	force custom fields to be colored with the same default color as system entry fields.
X'40' **Double-click not enter** -	distinguish between mouse double-clicks and the Enter key. Double mouse clicks are normally translated into Enter keys and return a value of 13 in Control key pressed. If this option is set, double clicks will return a value of -12 to the program (-12 does not have to be set up as a control key).
X'80' **Overlaid icons** -	allow icon fields to be overlaid with other fields.

Cell width division

Program name: **PD-DIV-WIDTH**
Used to derive: **Window Cell width division property**

Specifies the horizontal cell division that is to be used for positioning and sizing fields. If the division is 1/2 cell, a field can be positioned on a 1/2 cell boundary and its Column property will be in terms of 1/2 cells.

X'01' - 1/2 cell divisions
X'02' - 1/4 cell divisions
X'04' - 1/8 cell divisions

Cell height division

Program name: **PD-DIV-HEIGHT**
Used to derive: **Window Cell height division property**

Specifies the vertical cell division that is to be used for positioning and sizing fields. If the division is 1/2 cell, a field can be positioned on a 1/2 cell boundary and its Row property will be in terms of 1/2 cells.

X'01' - 1/2 cell divisions
X'02' - 1/4 cell divisions
X'04' - 1/8 cell divisions
X'08' - 1/16 cell divisions

Color

Program name: **PD-COLR**
Used to derive: **Window Color property**

The id of the color of the panel.

low-value = use default color

Program date format

Program name: **PD-PROG-DATE**

The format of dates in your program's Fields area.

d =	DMY
m =	MDY
low-value =	YMD
'/' (slash)=	M/D/Y

Separators are normally removed to leave a length of either 6 or 8, depending on whether the date includes the century.

Help panel

Program name: **PD-HELP**

The name of the object containing the internal help information. This might be the name of another panel or it could be the name of a file (with an "SP2" file extension.) See chapter B32 for more details of this.

Text options

Program name: **PD-TEXT-OPTIONS**
Used to derive: **Window Options property**

Switch that controls behavior in text mode. The following values are combined to form the final value:

X'01' - no field borders
X'02' - no pushbutton borders
X'04' - used internally
X'08' - do not display icons
X'10' - autosize panel to fill screen
X'20' - no window border
X'40' - no titlebar
X'80' - no system menu

Toolbar

Program name: **PD-TOOLBAR-NAME**

The name of the toolbar object which is actually another panel defining the layout of the toolbar. This property can be set in the panel editor directly or indirectly by defining a new toolbar by clicking the toolbar icon.

Message line color

Program name: **PD-MSG-COLR**

The code that defines the color of the message line.

More options

Program name: **PD-MORE-OPTIONS**
Used to derive: **Window More options property**

A switch controling various aspects of panel behavior. The following values are combined to form the final value:

X'01' **Use 3d effects** -	use 3d effects as the default
X'02' **Panel size in 1/100 ins** -	total size (Total width and height properties) is specified in 1/100 inches
X'04' **Size children to fit** -	automatically resize child window to fit within this window
X'08' **2.x scrolling panel** -	2.x scrolling panel
X'10' **2.x window title** -	2.x window title
X'40' **2.x mnemonics** -	2.x mnemonic panel
X'80' **Always pass data** -	return data from field even if override key pressed

Options-3

Program name: **PD-OPTIONS-3**
Used to derive: **Window More options, Options-3 properties**

Switch used to control various aspects of panel behavior:

X'01' **Minimum window size** -	panel window will not resize less than its original size
X'02' **No display** -	panel will not be displayed
X'04' **Messageline in statusbar** -	use status bar for message line (see also Message text property)
X'08' **Popup menu** -	the menu associated with this panel will be displayed as a popup rather than a menubar (the actual panel window will not be displayed at all). When the panel is conversed menu selections will be returned as normal with Control key pressed set to SP2-KEY-MENU (-6) and Menu id set to id of option selected. If no option is selected Control key pressed will be SP2-KEY-CLOSE (-5) or SP2-KEY-SWITCH (-3). A window switch will only be allowed if the converse panel Switch switch property is set to "y".
X'10' **Window help** -	window help. A question mark will be placed in the titlebar of the panel window. If this question mark is clicked, the mouse pointer will become a question mark and you can click on a field to bring up field-level help (panel-level if no field-level help available). This feature only works for dialogboxes and non-custom fields.
X'20' **Clipchildren** -	clip children. Prevents the panel background overwriting system fields and child windows. This option should be set for any panel containing Activex controls.
X'40' **Toolbar sunken border** -	toolbar sunken window. Display the window beneath a toolbar with a sunken border.
X'80' **2.x data problem** -	2.x data problem.

Options-4

Program name: **PD-OPTIONS-4**

Switch used to control various aspects of panel behavior:

X'01' **Force control field** -	force control field. Causes a key=-4 to be returned when a control key is hit on a field with Program control = "a" (return on exit) or "o" (return on exit or select). Control key will be returned on next converse-panel. Normally just the control key would be returned in this situation. This rule does not apply if key to be returned is -2 (selection key).
X'02' **Row and column in pixels** -	window position in pixels rather than cells if new window being derived from panel.
X'04' **Static text mnemonics** -	mnemonics for static text. For static text items, if "~" is included in the text, the following character will be underlined, indicating that this character is a mnemonic. For fields, set the Mnemonic property for all fields apart from push buttons, radio boxes and check boxes (for which it's set automatically.) This will allow the appropriate alt+letter key to be used to switch focus to this field.
X'08' **Anchor fields** -	panel contains anchor fields
X'10' **Generate popup html** -	used by Web Client
X'20' **Panel icon** -	search for an icon file named ppppppp.ico where pppppppp is the name of the panel. This icon will be displayed in the title bar instead of the standard system icon. The icon will be set or reset when the panel is initially displayed or when a SET-WINDOW-DEF call is made. The icon that is normally displayed for a window is controlled in the same way as the application icon - see the SET-ICON-FILE-NAME function.
X'40' **Anchor existing window** -	force anchor processing for a panel loaded into an existing window i.e. panel will be resized to fit into existing window.
X'80' **Container panel** -	this panel contains subwindow fields that will cause windows to be opened automatically.

Options-5

Program name: **PD-OPTIONS-5**

Switch used to control various aspects of panel behavior:

X'01' **Return on open** -	return control to program when a subwindow is opened.
X'02' **Return on entry** -	return control to program when a subwindow is given focus.
X'04' **Return on exit** -	return control to program when a subwindow loses focus.
X'08' **Return on close** -	return control to program when a subwindow is closed.

X'10' **Vertical scroll repeat**	allow window vertical scroll bar to control the repeat group contained in the window - same effect as setting window Scrollbar switch to "r".
X'20' **Program info in field data**	field User data holds the names of program-defined items used by the code generator.
X'40' **Hide panel**	panel is not visible.
X'80' **Grid panel**	panel is to be formatted as a grid.

Tabs options

Program name: **PD-TAB-OPTIONS**

A switch controling how tabs are displayed and handled. The following values are combined to form the final value:

X'01' -	allow tabs to take focus (same as SP2TAB=1 - see Appenix B).
X'02' -	allow multi-line tabs (same as SP2TAB=2 - see Appendix B).
X'04' -	display tabs vertically.
X'08' -	display tabs on the right or bottom.
X'10' -	disable themes for tabs. Vertical and bottom tabs are always non-themed.
X'20' -	handle tabs manually i.e. program responsible for displaying tab windows.

Options-6

Program name: **PD-OPTIONS-6**

Switch used to control various aspects of panel behavior:

X'01' **Entry field margins** -	insert a 2 pixel margin in system entry fields
X'02' **WC/X enhanced display** -	WC/X should use extra script to enhance the page display
X'04' **Same poff for dup fields** -	fields duplicated across panels within a container should share the same Fields area item i.e. have the same program offset
X'08' **Same pnum for dup fields** -	fields duplicated across panels within a container should share the same Colors/Types area item i.e. have the same program number
X'10' **All keys control keys** -	any key will return control (apart from input to system fields)

Options-7

Program name: **PD-OPTIONS-7**

Switch used to control various aspects of panel behavior:

X'01' **No visibility adj** -	do not attempt to adjust a window's position when it is first opened in order to make it wholly visible

Window border type

Program name: **PD-WIN-BOR-TYPE**
Used to derive: **Window Border Type property**

The border to be used for the window containing this panel.

low-value =	default border
m =	main window border
d =	dialog box border
n =	no border
p =	single line border
r =	resizable border (owned main window)
o =	dialog box border (non-owned dialog box)
t =	tab window (see below)
e =	raised border
s =	sunken border

f =	floating toolbar (skinny title bar)
v =	vertical splitter
h =	horizontal splitter

If Window border type is "t" and panel is in a child window, the window will be displayed as a tab window. The Title property will be used as the text for the tab. The width of the window will automatically be set so that it fits the parent window. If the parent window is a dialog box, the height of the window will be as specified in the Height property, otherwise the height will be automatically set as for the width. This allows push buttons, etc, to be put at the bottom of the parent window if the parent is a dialog box.

All windows are by default owned windows except for Window border type = 'm' and 'o'. Owned windows are not included in the task list and always appear in front of their owner (the previously opened window).

Initial switch

Program name: **PD-INITIAL-SW**

Used internally.

Description

Program name: **PD-DESCRIPTION**

Descriptive text for the panel. This property may also be used as a keyword for standard help. See chapter B32 for more information on this topic.

Title

Program name: **PD-TITLE**
Used to derive: **Window Title property**

The title of the panel to be displayed in the title bar of the panel window.

Cursor keys

Program name: **PD-CURS-KEYS**

The keys to be used to move the cursor within the panel, etc.

Cursor left

Program name: **PD-LEFT**

The key used to move the cursor to the left within a custom field or to the previous field in a field group (see group Tab within property).

Cursor right

Program name: **PD-RIGHT**

The key used to move the cursor to the right within a custom field or to the next field in a field group (see group Tab within property).

Cursor up

Program name: **PD-UP**

The key used to move the cursor up in a repeat group or if the panel Cursor movement property is set for vertical movement, or the key to move to the previous field in a field group (see group Tab within property).

Cursor down

Program name: **PD-DOWN**

The key used to move the cursor down in a repeat group or if the panel Cursor movement property is set for vertical movement, or the key to move to the next field in a field group (see group Tab within property).

Tab

Program name: **PD-TAB**

The key used to move to the next field or to the next field outside the current group (see the group Tab within property and the repeat group Tabbing type property).

Tab erase

Program name: **PD-TB-ERASE**

Alternative key used to move to the next field.

Backtab

Program name: **PD-BACKTAB**

The key used to move to the previous field or to the previous field outside the current group (see the group Tab within property and the repeat group Tabbing type property).

Backtab erase

Program name: **PD-BT-ERASE**

Alternative key used to move to the next field.

Delete

Program name: **PD-DELETE**

The key used to delete a character within a custom field.

Backspace

Program name: **PD-BACKSPAC**

The key used to delete the previous character within a custom field.

Erase

Program name: **PD-ERASE**

The key used to erase all subsequent characters within a custom field.

Insert

Program name: **PD-INSERT**

The key used to turn insert mode on/off within custom fields.

Home

Program name: **PD-HOME**

The key used to move to the first character within a custom field.

End

Program name: **PD-END**

The key used to move to the last character within a custom field.

Scroll up

Program name: **PD-SCRL-UP**

The key used to scroll up the current repeat group or the panel window if window Height is less than window Total height.

Scroll down

Program name: **PD-SCRL-DN**

The key used to scroll down the current repeat group or the panel window if window Height is less than window Total height.

Scroll left

Program name: **PD-SCRL-LT**

The key used to scroll left the current repeat group or the panel window if window Width is less than window Total width.

Scroll right

Program name: **PD-SCRL-RT**

The key used to scroll right the current repeat group or the panel window if window Width is less than window Total width.

Home panel

Program name: **PD-HOME-PAN**

The key used to move to the first tab field in the panel.

End panel

Program name: **PD-END-PAN**

The key used to move to the last tab field in the panel.

Control keys

Program name: **PD-CTRL-KEYS**

The keys to be used to return control to your program. In the panel editor, keys may be set directly or indirectly by setting the Help key property for a pushbutton or icon field. If a negative value is assigned to a key, then the key corresponding to the positive value will be a control key, but it will be an edit override control key, having the same functionality as the Edit override key.

Message text

Program name: **PD-MSG-TEXT**

The text that is to be displayed in the message line of the panel window. This text will only be displayed if a message line exists for this window (see Message length property) and there is no field-level message text for the current field (see field Message text property).

If message text is to be displayed in a status bar (see Options-3 property), it may be formatted as follows:

/width/border/text/width/border/text/.....

width=	width of this part in 1/10 cells (-1 means extend to right of window)
border=	y/n to display/suppress the border for this part
text=	text to be displayed in this part

For example:

/200/y/hello/-1/y/from my program/

This would display "hello" in the first part of the status bar (for a length of 20 cells) and "from my program" in the remaining part.

The first part can be easily changed at runtime by resetting Message text to the new text to be displayed - the format details are not required unless the format needs to be changed. Also set Message refresh switch to zero to force a refresh of the status bar (use the set-property function to do this). To change multiple parts at runtime use the format "/first text/second text/third text/".

It is also possible to embed special text in the statusbar using the variables $CAP, $NUM, $INS, $DAT and $TIM - use these in the statusbar format instead of a text string, for example:

```
/200/y/hello/100/y/$DAT/
```

$CAP displays "CAP" if Caps Lock is on
$NUM displays "NUM" if Num Lock is on
$INS displays "INS" if in insert mode or "OVR" if in overwrite mode
$DAT displays the current date
$TIM displays the current time

Dollar variables should be included on only one window in your system (the main window) but the display will be updated even if this main window is not active.

User data

Program name: **PD-USER-DATA**

Allows you to store your own data within the panel definition.

For a grid panel (Options-5 = X"80"), this property is used to hold options for the grid as follows:

rc -	total number of rows (default = 100)
dw -	divider width (default = 2)
dh -	divider height (default = 2)
hd -	headings y/n (default = y)
sz -	column resize allowed y/n (default = y)

For example: rc=500,dw=1,dh=1

Help keyword

Program name: **PD-HELP-KEYWORD**

The keyword used in conjunction with standard help (see chapter B32). This takes priority over the implied panel-level keyword in the panel Description property. The keyword can be prefixed with the name of the help file to be used, enclosed within forward slashes. For example "/myhelp/help-for-account/" would cause a file named "myhelp.hlp" to be searched for a keyword of "help-for-account". If this item is set to "HELP_CONTENTS", the contents page of the help file will be displayed. If the keyword text is prefixed with a "!", it will be interpreted as an application to run, for example:

```
!winhlp32 -P -Icontext-string help-file
```

If the keyword text is prefixed with an "@", it will be interpreted as a macro to run, for example:

```
@JumpId(`help-file',`context-string')
```

CHAPTER E6 - Field Properties

Overview

These properties define the appearance and behavior of a field.

Field properties are set and accessed programmatically either by using the SET-PROPERTY and GET-PROPERTY functions or by using data items in the FIELD-DEF parameter before a SET-FIELD-DEF call and after a GET-FIELD-DEF call. To set a property using SET-FIELD-DEF after a field is already defined you would normally make a GET-FIELD-DEF call followed by a SET-FIELD-DEF call. The FIELD-DEF parameter is contained in the copy file SP2.CPY.

Parameter layout

```
  01  SP2-FIELD-DEF.
      05  SP2-FD-RET-CODE         PIC S9(4) COMP-5.
      05  SP2-FD-LENS.
          10  SP2-FD-LEN-LEN      PIC S9(4) COMP-5 VALUE +30.
          10  SP2-FD-NUM-LEN      PIC S9(4) COMP-5 VALUE +52.
          10  SP2-FD-CHAR-LEN     PIC S9(4) COMP-5 VALUE +74.
          10  SP2-FD-VAR-LEN      PIC S9(4) COMP-5 VALUE +2000.
          10  SP2-FD-VAR-LENS.
              15  SP2-FD-FORMAT-LEN
                                  PIC S9(4) COMP-5 VALUE +0.
              15  SP2-FD-CAPTION-LEN
                                  PIC S9(4) COMP-5 VALUE +0.
              15  SP2-FD-INITIAL-LEN
                                  PIC S9(4) COMP-5 VALUE +0.
              15  SP2-FD-CLASS-LEN
                                  PIC S9(4) COMP-5 VALUE +0.
              15  SP2-FD-RANGE-LEN
                                  PIC S9(4) COMP-5 VALUE +0.
              15  SP2-FD-DISCRETE-LEN
                                  PIC S9(4) COMP-5 VALUE +0.
              15  SP2-FD-MSG-TEXT-LEN
                                  PIC S9(4) COMP-5 VALUE +0.
              15  SP2-FD-USER-LEN PIC S9(4) COMP-5 VALUE +0.
              15  SP2-FD-HELP-LEN PIC S9(4) COMP-5 VALUE +0.
          10  SP2-FD-LENL-LEN     PIC S9(4) COMP-5 VALUE -1.
          10  SP2-FD-NUML-LEN     PIC S9(4) COMP-5 VALUE +44.
      05  SP2-FD-DATA.
 ******** SP2-FD-NUM-DATA ********
          10  SP2-FD-ID           PIC S9(4) COMP-5.
          10  SP2-FD-GUI-ID       PIC S9(4) COMP-5.
          10  SP2-FD-GUI-ID-2     PIC S9(4) COMP-5.
          10  SP2-FD-OCCURRENCE   PIC S9(4) COMP-5.
          10  SP2-FD-BASE-ID      PIC S9(4) COMP-5.
          10  SP2-FD-ROW          PIC S9(4) COMP-5.
          10  SP2-FD-COL          PIC S9(4) COMP-5.
          10  SP2-FD-PROG-OFF     PIC S9(4) COMP-5.
          10  SP2-FD-FLD-NUM      PIC S9(4) COMP-5.
          10  SP2-FD-TAB-NUM      PIC S9(4) COMP-5.
          10  SP2-FD-PROG-NUM     PIC S9(4) COMP-5.
          10  SP2-FD-WIDTH        PIC S9(4) COMP-5.
          10  SP2-FD-HEIGHT       PIC S9(4) COMP-5.
          10  SP2-FD-MAX-LEN      PIC S9(4) COMP-5.
          10  SP2-FD-PROG-LEN     PIC S9(4) COMP-5.
          10  SP2-FD-ITEM-LEN     PIC S9(4) COMP-5.
          10  FILLER              PIC S9(4) COMP-5.
          10  SP2-FD-HELP-KEY     PIC S9(4) COMP-5.
          10  FILLER              PIC S9(4) COMP-5.
          10  SP2-FD-GROUP-ID     PIC S9(4) COMP-5.
          10  SP2-FD-REPEAT-ID    PIC S9(4) COMP-5.
          10  SP2-FD-FONT-ID      PIC S9(4) COMP-5.
          10  SP2-FD-PROG-OFF-L   PIC S9(8) COMP-5.
          10  SP2-FD-PROG-NUM-L   PIC S9(8) COMP-5.
 ******** SP2-FD-CHAR-DATA *******
          10  SP2-FD-NAME         PIC X(30).
          10  SP2-FD-TYPE         PIC X.
          10  SP2-FD-OUTPUT       PIC X.
          10  SP2-FD-PROG-DEC     PIC X.
          10  FILLER              PIC X(4).
          10  SP2-FD-INIT-NUMS    PIC X.
          10  SP2-FD-MISC-OPTIONS PIC X.
          10  FILLER              PIC X.
          10  SP2-FD-HELP         PIC X(8).
          10  SP2-FD-MORE-OPTIONS PIC X.
          10  SP2-FD-BOR-COLR     PIC X.
          10  SP2-FD-ANCHOR       PIC X.
          10  SP2-FD-OPTIONS-3    PIC X.
          10  SP2-FD-OPTIONS-4    PIC X.
          10  FILLER              PIC X(3).
          10  SP2-FD-REQUIRED     PIC X.
          10  SP2-FD-PROG-CTRL    PIC X.
          10  SP2-FD-JUSTIFY      PIC X.
          10  SP2-FD-FILL         PIC X.
          10  SP2-FD-ASSUME-DEC   PIC X.
          10  SP2-FD-SPEC-FMT     PIC X.
          10  SP2-FD-CASE         PIC X.
          10  SP2-FD-IMBED-BLANKS PIC X.
          10  SP2-FD-CUR-COLR     PIC X.
          10  SP2-FD-CURS-SKIP    PIC X.
          10  SP2-FD-CURS-SHOW    PIC X.
          10  SP2-FD-BLANK-FIRST  PIC X.
          10  SP2-FD-BLANK-ZERO   PIC X.
          10  SP2-FD-CTRL-TYPE    PIC X.
          10  SP2-FD-COLR         PIC X.
          10  SP2-FD-MNEMONIC     PIC X.
          10  SP2-FD-BOR-TYPE     PIC X.
          10  SP2-FD-PROG-SPEC    PIC X.
 ******** SP2-FD-VAR-DATA ********
          10  SP2-FD-VAR-DATA     PIC X(2000).
 ******** SP2-FD-FMT *************
 ******** SP2-FD-CAPTION *********
 ******** SP2-FD-INITIAL-VAL *****
 ******** SP2-FD-CLASS ***********
 ******** SP2-FD-RANGE-VALS ******
 ******** SP2-FD-DISC-VALS *******
 ******** SP2-FD-MSG-TEXT ********
 ******** SP2-FD-USER-DATA *******
 ******** SP2-FD-HELP-KEYWORD ****
```

FD-VAR-LEN is the sum of the lengths of FD-FMT, FD-CAPTION, FD-INITIAL-VAL, FD-CLASS, FD-RANGE-VALS, FD-DISC-VALS, FD-MSG-TEXT, FD-USER-DATA and FD-HELP-KEYWORD.
FD-FORMAT-LEN is the length of FD-FMT.
FD-CAPTION-LEN is the length of FD-CAPTION.
FD-INITIAL-LEN is the length of FD-INITIAL-VAL.
FD-CLASS-LEN is the length of FD-CLASS.
FD-RANGE-LEN is the length of FD-RANGE-VALS.
FD-DISCRETE-LEN is the length of FD-DISC-VALS.

FD-MSG-TEXT-LEN is the length of FD-MSG-TEXT
FD-USER-LEN is the length of FD-USER-DATA
FD-HELP-LEN is the length of FD-HELP-KEYWORD

Id

Program name: **FD-ID**

The id used to identify the field. Set automatically in the panel editor when a new field is defined. If FD-ID is set to zero before a SET-FIELD-DEF call, a new field will be created with an automatically assigned id. In order to find out the id used in this case, set FD-ID to -1 and make a GET-FIELD-DEF call and on return FD-ID will be reset to the id used. Otherwise, FD-ID should be set to the id of the field to be accessed by GET-FIELD-DEF or SET-FIELD-DEF. The maximum value for Id is 5000.

GUI id

Program name: **FD-GUI-ID**

The first part of the handle used by the operating system to identify the field.

GUI id (2)

Program name: **FD-GUI-ID-2**

The second part of the handle used by the operating system to identify the field.

Occurrence

Program name: **FD-OCCURRENCE**

The occurrence number of the field if it is part of a repeat group. Repeat fields are generated automatically when repeat groups are defined so field definitions with an occurrence number greater than 1 should not be accessed directly.

Base id

Program name: **FD-BASE-ID**

The id of the original field if this is a repeat field. Repeat fields are generated automatically when repeat groups are defined so field definitions with an occurrence number greater than 1 should not be accessed directly.

Row

Program name: **FD-ROW**

The row in cells within the panel where the field is positioned. May be set directly in the panel editor or indirectly by selecting and dragging the field.

Column

Program name: **FD-COL**

The column in cells within the panel where the field is positioned. May be set directly in the panel editor or indirectly by selecting and dragging the field.

Program offset

Program name: **FD-PROG-OFF**

The offset within the program Fields area of the field if this offset is not greater than 9999 (otherwise Long program offset is used). Automatically set in the panel editor prior to code generation or saving the panel. May be calculated outside the editor by running the external code generator - see Appendix E.

If this item is set to -1 it means that the offset of the field is held in Long program offset (see below).

On return from a GET-FIELD-DATA call for a list box or combo box, this item holds the index (0, 1, 2, etc.) of the item selected.

Field number

Program name: **FD-FLD-NUM**

The number of the field in a left-to-right, top-to-bottom sequence including display-only fields. Automatically recalculated as fields are added and deleted.

Tab number

Program name: **FD-TAB-NUM**

The number of the field within the panel tabbing sequence. Automatically set when a field is created but may be manually overridden.

Program number

Program name: **FD-PROG-NUM**

The occurrence number of the field within the program Colors and Types areas if this number is not greater than 9999 (otherwise Long program number is used). Automatically set by the panel editor prior to code generation or saving the panel.

Width

Program name: **FD-WIDTH**

The width of the field in terms of 1/10 cells. May be set directly in the panel editor or indirectly by selecting the field and dragging the left or right border.

Height

Program name: **FD-HEIGHT**

The height of the field in terms of 1/10 cells. May be set directly in the panel editor or indirectly by selecting the field and dragging the top or bottom border. May be set to 0 which means use a default height for the field.

Maximum length

Program name: **FD-MAX-LEN**

The maximum number of characters that can be displayed or entered in the field. Only required for entry fields (Control type = low-value or "e"). Automatically set in the panel editor based on the Format property. If FD-MAX-LEN is set to -1 on a SET-FIELD-DEF call, the maximum number of characters will be calculated automatically based on the Format property.

Program length

Program name: **FD-PROG-LEN**

The length of the field as it appears in the program Fields area. Automatically set in the panel editor based on the Format property. May be calculated outside the editor by running the external code generator - see Appendix E.

Item length

Program name: **FD-ITEM-LEN**

For a List Box or Combination Box (Control type = "l" or "o"), this is the length of an item in the list. Automatically set in the panel editor based on the Format property.

For an Icon field (Control type = "i"), if Item length is not zero it specifies the length of the name of the icon file in the Value property (see Value property). Automatically set in the panel editor based on the Format property. This must be set as well as Program length in order to use the program Fields area to change an icon at runtime. If the Miscellaneous X'20' option is also set, Item length will be calculated as Format length divided by 4.

Help key

Program name: **FD-HELP-KEY**

The key that the user will press to invoke field-level help. See chapter B32 for more details of this.

For a Push Button or Icon (Control type = "p" or "i"), specifies the key code that will be returned if the Push Button or Icon is clicked.

For Subwindow field (Control type = "w"), specifies the key that will be used to open the new window.

Group id

Program name: **FD-GROUP-ID**

The id of the group to which this field belongs. Set automatically when the group is defined.

Repeat id

Program name: **FD-REPEAT-ID**

The id of the repeat group to which this field belongs. Set automatically when the repeat group is defined.

Font id

Program name: **FD-FONT-ID**

The id of the font to be used for text in the field. Fonts can be established in the panel editor or by using the SET-FONT-DEF function. Zero means use the default font.

Long program offset

Program name: **FD-PROG-OFF-L**

The offset within the program Fields area of the field if this offset is greater than 9999. Automatically set in the panel editor prior to code generation or saving the panel. May be calculated outside the editor by running the external code generator - see Appendix E.

Long program number

Program name: **FD-PROG-NUM-L**

The occurrence number of the field within the program Colors and Types areas (see Converse panel properties) if this occurrence is greater than 9999. Automatically set by the panel editor prior to code generation or saving the panel.

Name

Program name: **FD-NAME**

The name of the field.

Type

Program name: **FD-TYPE**

For an entry field (Control type = low-value or "e"), Type controls the sort of data that will be accepted into the field:

low-value =	any data (subject to range validation, etc.)
n =	numeric
d =	date

For a push button field (Control type = "p"), Type specifies the type of push button:

low-value =	regular push button
X"01"=	default push button

For an icon field (Control type = "i"), Type specifies the type of "icon":

low-value =	regular bitmap (file name is in the Initial value property)

X"01" =	bitmap-exact. This setting is used to display 256-color bitmaps using the exact colors that were used when the bitmap was created. This is usually necessary when displaying photographs, for instance. It is not advisable to use this setting if you need to display more than one bitmap within the same window.
X"02" =	bitmap-pushbutton. Field will act like a pushbutton but will display the bitmap (indented when selected) as specified by the Value property.
X"03" =	bitmap-radiobutton. Field will act like a radiobutton but will display the bitmap (indented when selected) as specified by the Value property.
X"04" =	bitmap-checkbox. Field will act like a checkbox but will display the bitmap (indented when selected) as specified by the Value property.
X"05" =	Activex control. Field will act as a container for the activex control as identified by the Value property.
X"06" =	splitter field. Field can be used to track mouse movement. When mouse button is released, panel-key is set to -32 and panel-last-col is set to the column location of the mouse pointer.
a =	audio-play (file name is specified by the Value property). The audio clip will play as soon as the window is displayed.
b =	audio-hold. The audio clip will not play without program intervention.
c =	audio-replay. Use this setting in FD-TYPE with a SET-FIELD-DEF call in your program to play an audio-hold icon.
v =	video-play (file name is specified by the Value property). The video clip will play as soon as the window is displayed.
w =	video-hold. The video clip will not play without program intervention.
x =	video-replay. Use this setting in FD-TYPE with a SET-FIELD-DEF call in your program to play a video-hold icon.
y =	video-repeat. The video clip will play and repeat itself until the window is closed.

Protection

Program name: **FD-OUTPUT**

Used to control whether user input will be accepted into the field.

low-value =	input accepted
y =	no input accepted, display-only field (except mouse input to Push Buttons and icons)
p =	no input accepted but user may tab to the field
s =	secure, input accepted but not displayed
g =	greyed out, absolutely no input accepted (same as y except for Push Buttons and icons). g may cause a visible "greying out" in GUI mode.
h =	invisible
r =	refresh image (DISPLAY-FIELD function only)

Decimals

Program name: **FD-PROG-DEC**

Specifies the number of decimals in the program's definition of this field. This allows the program definition to be independent of the display definition. This property is only referenced if the Program Data property (FD-PROG-SPEC) is set appropriately.

Initialize if numeric

Program name: **FD-INIT-NUMS**

Specifies if a numeric field should be initialized to zero.

low-value =	default to panel Initialize numerics property
n =	do not initialize to zero

Miscellaneous

Program name: **FD-MISC-OPTIONS**

Controls various aspects of field behavior. The following values are combined to form the final value:

X'01' **2.x auto-input** =	accept data into this field based on input to support panel (panel Help property).
X'02' **2.x disabled in group** =	2.x compatibility option.
X'04' **No delete on resize** =	allows field to be resized without losing its values.

X'08' **No bring window to top** =	prevents this field overwriting another field which may be overlaying it.
X'10' **Dynamic html values** =	used in conjunction with the Web Client product to allow dynamic values to be passed to HTML listboxes and comboboxes.
X'20' **Multiple image button** =	see Value property.
X'80' **Dynamic icon mods** =	for icons with Border property = "d", no border will be drawn but "mouse over" features will still be active. For other icons, background color will be reset as if Border property = "d". (See Border property.)

Help

Program name: FD-HELP

The name of the object containing the COBOL sp2 internal help information. This might be the name of another panel or it could be the name of a file (with an "SP2" file extension.) See chapter B32 for more details of this.

More options

Program name: FD-MORE-OPTIONS

Controls various aspects of field behavior. The following values are combined to form the final value:

X'01' **Exact listbox height** =	for a List Box, height will be exactly the value of the Height property even if this means partial items are displayed. For a Combo Box, height of the dropdown will be determined by the value of the Height property.
X'02' **Return right click** =	specify that control is to be returned to the program when user right clicks on a system field, if Mouse switch is "r" or "o". If the system field is a list box, the item which was clicked can be retrieved by setting fd-more-options to x"02" and then making a get-field-data call for the list box.
X'04' **Allow color change** =	for a system field, specify that color might be changed at runtime.
X'08' **Return on combo dropdown** =	for a Combo Box, specify that control is to be returned to the program when the drop down is about to be displayed. This allows you to reset the contents of the drop down or display another window instead of the regular drop down. KEY-VBX is returned and Menu-option is set to "ComboDropDown". See sample program comboret for details.
X'10' **Hold multi-scroll offset** =	for a scrolling multi-line entry field, specify that displacement of the text is to be preserved when the field is redisplayed..
X'20' **Combo dropdown on focus** =	for a combo box field, specify that the drop down is to be displayed as soon as the field receives focus.
X'80' **Refresh only on change** =	for a custom entry field, special handling for a transparent background.

Border color

Program name: FD-BOR-COLR

The id of the color to be used for the field border if the Border type property is set to "L" (line border). Low-value means use the default border color.

Anchor

Program name: FD-ANCHOR

Causes a field to change size or position relative to the containing window size. The following values are combined to form the final value:

X'01' =	change size relative to right window border
X'02' =	change size relative to bottom window border
X'04' =	free left allows a field anchored to the right (X'01') to move horizontally rather than change size
X'08' =	free top allows a field anchored to the bottom (X'02') to move vertically rather than change size

The following properties should also be set:

Panel Options-4

X'08' =	panel contains anchor fields

Window Options-3 (usually derived from panel property above)

X'40' =	window contains anchor fields

Panel Options-3

X'01' =	minimum size for containing window

Window More options (usually derived from panel property above)

X'10' =	minimum size for window

Options-3

Program name: **FD-OPTIONS 3**

Controls various aspects of field behavior. The following values are combined to form the final value:

X'01' **No icon bottom border** =	for a bitmap button with a line border, specifies that the bottom border is omitted. This allows these buttons to be used in conjunction with tab panels.
X'02' **Delayed combo select** =	for a Combo Box with Program control = return on select, specifies that control will only be returned when the drop down has been closed.
X'04' **No tab stop** =	specifies that this field should not be included in the tab sequence so that a mouse click must be used to make it current.
X'08' **Message as hint** =	for all fields except custom fields, specifies that message text assigned to the field Message text property will be displayed as an hint next to the field when the mouse pointer is within the field.
X'10' **Multi-line pushbutton** =	for a push button, specifies that text should wrap to another line if there is not enough roon on one line..
X'20' **No return in multi-line** =	for a multi-line entry field,, specifies that the Return key should not be used as a new line key but rather as a regular Control key.
X'80' **Special text** =	for a multi-line custom field, allow imbedded control characters to set the font and color of specific test: \fxxx - set font where xxx is font id \cxxx - set color where xxx is color id \n - new line \sx - use "x" instead of "\" \a"xxx" – set class xxx

Options-4

Program name: **FD-OPTIONS-4**

Controls various aspects of field behavior. The following values are combined to form the final value:

X'01' **Default value if blank** =	use initial value if program area set to blank.
X'02' **Return on entry** =	return control to program when focus is given to the field. Key is set to SP2-KEY-ENTER-FIELD (31).
X'04' **Help as icon href** =	if field is clicked, the address held in Help keyword will be loaded into the browser.
X'08' **Windowed static** =	specifies that a system entry field is to be treated as a static item. This allows labels to be contained in their own window but also to be transparent if Color is set appropriately - regular system entry fields do not support transparency.
X'10' **No icon bg clear** =	background of windowed icon is not to be cleared before image is drawn - mainly for use with transparent images.
X'20' **Push button fields area** =	specifies that the program Fields area should override the Value property for setting push button text if a push button field has a Fields area item allocated (usually by setting the Format property).
X'40' **ActiveX protection** =	specifies that the program Types area should control if input is allowed into ActiveX fields just as for regular fields.
X'80' **ActiveX program value** =	specifies that the program Fields value should be used to set/get the control's Text property. Targeting the Text property can be changed by setting the field Caption to the name of a different property.

Options-5

Program name: **FD-OPTIONS-5**

Controls various aspects of field behavior. The following values are combined to form the final value:

X'01' **Return on change** =	causes KEY-CHANGE-FIELD (-33) to be returned on any change to a custom or system entry field. The Converse-panel Last column property will be set to the offset of the cursor within the field and this property will control the offset of the cursor on the next call.
X'02' **Suppress return on exit** =	same as setting Usage option = "i" for a pushbutton – allows this feature to be enabled for an icon button.
X'04' **Default button** =	same as setting Type = X"01" for a pushbutton – allows this feature to be enabled for an icon button.
X'08' **Secure** =	same as setting Protection = "s" but allows a protected field to also be secure.

Required

Program name: **FD-REQUIRED**

Controls whether data must be entered into this field. If a non-numeric field is initialized to a non-blank value either by your program or when defining the field, the requirement test will be satisfied. A numeric field must be initialized to a non-zero value to pass the test unless the panel Initialize numerics property is set to "n".

low-value =	not required
y =	required

Program control

Program name: **FD-PROG-CTRL**

Control whether the field will cause control to be returned to your program once the user exits it or selects it. If control is returned as a result of this property, the Control key pressed property will be set to SP2-KEY-CTRL-FIELD (-4).

low-value =	control not returned
a =	control returned on exit out of field
m =	control returned if contents of field are modified
s =	control returned if field is selected
o =	control returned on exit out of field or if field is selected

Justify

Program name: **FD-JUSTIFY**

Controls whether data will be justified in any way before it is displayed within the field. Right justification is useful for lining up numeric fields in neat columns. If right or center justification is set, data will by default be entered from the right, calculator-style. To avoid this behavior (for Custom fields only), set the panel Shift numerics property "y" which causes right- and center-justified fields (as well as just numeric fields) to be shifted to the left for input.

low-value =	no justification
l =	justify to the left
r =	justify to the right
c =	center
s =	shift existing contents to the left prior to input
j =	justify to the left and right (Options-3 = X'40' Special text must also be set).

Fill character

Program name: **FD-FILL**

Specifies the character that will be used to fill blank spaces after user input.

low-value =	zero for numeric fields, blank for non-numeric fields.

Assume decimal

Program name: **FD-ASSUME-DEC**

Used to control whether a decimal point should be assumed if it is not specifically entered in a numeric field.

low-value =	refer to panel Assume decimal property
y =	assume decimal
n =	do not assume decimal

Special format

Program name: **FD-SPEC-FMT**

For a custom or system entry field (Control type property is low-value or "e"), specifies that the field format contains special separator characters that are fixed and non-enterable. Only X, 9 and A in the field format will be treated as an enterable character slot. Do not use parentheses to specify multiple X's, 9's or A's in the field format.

low-value =	regular format
y =	special format

For a bitmap icon field (Control type property is "i" and Type property is low-value or X"01"), specifies if the field or image is to be automatically resized.

low-value =	clip image to field size
i =	resize field to image size
f =	resize bitmap image to field size
a =	resize bitmap image to field size but maintain aspect ratio

For a bitmap button field (Control type property is "i" and Type property is X"02", X"03" or X"04"), specifies if the button contains an image or just text. For text buttons, the Color property defines the default color and the Current color poperty defines the color when clicked if it's a display-only button or when given focus if it's a protected button. Text buttons can be used for tabs with a different look.

low-value =	image
t =	text

For bitmap push buttons only (Type property is X"02"), also specifies that the button contains text with an image - the image in the background and the text in the foreground. Miscellaneous options X"20" must also be set to allow for a slot for the text and a slot for the image (see Value property). The image will be stretched to fit the button by dividing it into 3 parts, the first used for the left side of button, the second stretched to fill the middle of button and the third for the right side of button.

w =	text with image

For a video icon field (Control type property is "i" and Type property is "v" or "w"), specifies if the image should be automatically resized to fit the field.

low-value =	ignore field size
f =	stretch image to fit field size

For a multiline system entry field (Control type property is "e" and Usage option property is m or s or h), specifies if end-of-lines should be marked.

low-value =	do not mark end-of-lines
e =	marker characters will be placed at the end of every line even if the return key has not been used to start a new line. The characters used are X"0D0D0A". If you scan for X"0D0A", you will be able to detect end-of-lines in the data entered into multiline edits.

For a listbox (Control type property is "l"), controls horizontal scrolling (amount of scroll is based on the Item length property and the size of font being used) and tab expansion (two spaces are expanded into a tab):

low-value =	no horizontal scrolling and no tab expansion
h =	horizontal scrolling and tab expansion
t =	tab expansion only
n =	horizontal scrolling only

Case

Program name: **FD-CASE**

Controls whether user input will have its case converted if necessary.

low-value =	no case conversion
u =	uppercase
l =	lowercase
f =	first character uppercase
n=	no case conversion even if SP2CAS (see Appendix B) is set

Imbedded blanks

Program name: **FD-IMBED-BLANKS**

Controls whether imbedded blanks will be allowed in user input.

low-value =	imbedded blanks allowed
n =	imbedded blanks not allowed

Current color

Program name: **FD-CUR-COLR**

The id of the color to be used to identify the field if it is currently set up to receive user input. Colors can be established in the panel editor or by using the SET-COLOR-DEF function. Low-value means refer to the panel Current field color property.

If this property is set to a value other than low-value for a windowed icon field (Control type property is "i" and Usage option is "w"), the mouse shape will be changed when the mouse moves over the icon. The shape used will be either SP2-MOUSE-LOCATE or the shape defined by the xxxxxxxx.cur file where xxxxxxxx is the name of the first panel file opened or as set using the SET-ICON-FILE-NAME function. A .ani (animated cursor) file may be used instead of a .cur file.

Usage option

Program name: **FD-CURS-SKIP**

For a custom or system entry field (Control type property is low-value or "e"), specifies a single- or multi-line entry field.

low-value =	single-line entry field
m =	multi-line entry field (display-only for custom entry field)
s =	multi-line entry field with vertical scroll bar (system entry field only)
h =	horizontally scrolling multi-line entry field (system entry field only)

For a single-line custom or system entry field (Control type property is low-value or "e"), controls whether the cursor should automatically jump to the next field when no more input is possible in this field.

low-value =	refer to panel Cursor skip property
y =	jump to next field
n =	do not jump to next field

For a single-line system entry field (Control type property is "e"), specifies that field is to be displayed as a spin button. Type property must be set to "n" and Format property must be set to a valid numeric format.

low-value =	regular system entry field
b =	spin button

For a push button field (Control type property is "p"), this item is used to indicate how control is returned to the program if previous field was a "return-on-exit" field (Program control property is "a", "m" or "o").

low-value =	return normally after appropriate "return-on-exit" processing (Control key pressed property set to SP2-KEY-CTRL-FIELD). This may result in return being returned twice for a single pushbutton click.
i =	return immediately when the pushbutton is clicked even if the cursor is currently positioned on a "return-on-exit" field.

For a bitmap icon field (Control type property is "i" and Type property is low-value, X"01", X"02", X"03" or X"04"), specifies a windowed icon field. Windowed icons allow the system to easily track mouse movement so that a depressed button can be raised when the mouse pointer is moved off it and icon hints (see field Message text property) can be displayed when the mouse pointer is moved over it.

low-value =	non-windowed icon
w =	windowed icon.

For an activex control field (Control type property is "i" and Type property is X"05"), specifies how keyboard input is handled.

low-value =	keys handled normally
t =	tab and enter keys handled by activex control
a =	all keys handled by activex control
e =	enter key handled by activex control but not tab

For a list box field (Control type property is "l"), specifies how entries should be sorted and how entries may be selected.

low-value =	sort entries
o =	do not sort entries
m =	multi-select/sorted
n =	multi-select/unsorted

Selections in a multi-select list box must be retrieved using the GET-FIELD-DATA function.

For a combo box field (Control type property is "o"), specifies how entries should be sorted.

low-value =	do not sort entries
o =	sort entries

For a subwindow field (Control type property is "w"), specifies the type of subwindow..

low-value =	child window
t =	child window on top
p =	popup window
n =	panel only (no child window)

Cursor show

Program name: **FD-CURS-SHOW**

For custom entry fields (Control type property is low-value), controls the appearance of the cursor while in field

low-value =	refer to panel Cursor show property
y =	underline cursor
n =	no cursor
b =	block cursor
v =	vertical (I-beam) cursor

For non-display only bitmap icon fields (Control type property is "i" and Protection property is low-value or "p"), controls the focus rectangle.

low-value =	reverse-video focus rectangle
n =	no focus rectangle

Blank first

Program name: **FD-BLANK-FIRST**

Controls whether the field should be blanked out if an entry is made into the first position before any other positions. This allows fields, particularly numeric ones, to be easily changed. This property only applies to custom and system entry fields.

For custom entry fields:

low-value =	if numeric field, refer to panel Blank numerics property, otherwise do not blank
y =	blank out field

For system entry fields:

low-value =	highlight field on initial entry
n =	do not highlight field on initial entry

Blank if zero

Program name: **FD-BLANK-ZERO**

Controls whether a numeric field should be displayed as blank if its value is zero.

low-value =	do not display blank
y =	blank if zero

Control type

Program name: **FD-CTRL-TYPE**

The type of the field in terms of the control-types found in windowing environments. Normally set in the panel editor by creating a new field with one of the field icons in the toolbar.

low-value =	Custom entry field
e =	System entry Field
p =	Push Button
r =	Radio Button
c =	Check Box
l =	List Box
o =	Combination Box
i =	Icon
v =	Vertical Scroll Bar
h =	Horizontal Scroll Bar
w =	Subwindow field

Color

Program name: **FD-COLR**

The id of the color of the field. If field Type is "i" (icon) and Color is set to a color with a transparent background, all the white pixels in the bitmap image will be transparent.

low-value =	refer to panel Color property

Mnemonic

Program name: **FD-MNEMONIC**

The character that allows the user to immediately jump to or select this field. Automatically set as a result of including a "~" or "&" character in the Caption property of a Radio button or Check Box or the Value property of a Push Button. The mnemonic will normally only be useable if the field is included in a group, except for Push Buttons when the (ALT) key may be used in conjunction with the mnemonic to select the Push Button.

If the panel Options-4 X'04' switch is set on, the mnemonic character may be manually set for fields other than the above and the (ALT) key in conjunction with the mnemonic may be used to shift the focus to all field types. The mnemonic should correspond to the character underlined in the preceding static text item using the "~" character.

low-value = no mnemonic

Border type

Program name: **FD-BOR-TYPE**

For entry fields, list boxes and combo boxes (Control type property is low-value, "e", "l" or "o"), specifies the type of border for the field.

low-value =	no border
l (lower case "L") =	default border
f =	flat graphical border, usually light blue.
3 =	old-style 3-dimensional border.
t =	thin 3d border. This border is only one pixel wide and is therefore particularly useful if you are cramped for space. The border looks best if the field which it surrounds has a background color of light-grey and the window has a color other than bright-white.
L=	single-line, 2-dimensional border (entry fields only). This can be used in conjunction with a repeat group to create a flat grid. Set panel Cell width/height to 1/1, repeat Vertical/Horizontal gap to -1, repeat Cursor movement to -2, repeat Tabbing type to a or v, and make sure that field Height is a specific number (i.e. not zero).
a =	(Control type = low-value only) no border but align text as if a border were present. This is useful for aligning variable labels with bordered fields.

For bitmap buttons (Control type property is "i" and Type property is X"02", X"03" or X"04"),
specifies the type of border for the button.

Low-value=	raised frame
n=	no border
m=	no border rounded
l=	line border
k=	rounded line border
r =	rounded frame border
s =	square frame border
p =	same as "r" except that border is included within button area
q =	same as "s" except that border is included within button area
d =	border is only displayed when mouse is moved over button area. Also allows an alternative image to be displayed when the mouse is over the button area (see Miscellaneous property – multi-image bitmap button). Also causes all pixels matching the top left pixel to be reset to match the field or panel Color.

Program data

Program name: **FD-PROG-SPEC**

Controls whether the Program offset, Program length, Program number and Decimals properties can be specified manually rather than be calculated automatically. The following values are combined to form the final value:

low-value =	calculate these properties automatically
X'01' =	set Program offset property
X'02' =	set Program length property

X'04' =	set Program number property
X'08' =	set Decimals property
X'10' =	ignore groups when ordering this field and do not generate Program number property for this field if Control type property is "v" or "h" (2.x compatibility option)
X'20' =	on a GET-FIELD-DATA call, this causes FD-PROG-NUM to be used for the number of the first selection to be returned from a multi-select list box.
X'40' =	on a DISPLAY-FIELD call, this causes FD-PROG-OFF to be treated as the zero-based offset of the item to be selected in a list box.

Format

Program name: FD-FMT

The COBOL format of the field i.e. display format. This may be a non-standard format if the Special format property is set to "y". If the field is a Date Field, use:

'm'	for month
'd'	for day
'y'	for year

For example, mm/dd/yy would be used for the format of Month/Day/Year. Only regular alphanumeric formats (PIC X) are supported for Combination Boxes and List Boxes.

Caption

Program name: FD-CAPTION

The caption for a Radio Button or Check Box. Use a '~' (tilde character) to represent the mnemonic character.

For an ActiveX field, this can be used to specify the property to be targeted by the Program Fields area value.

Value

Program name: FD-INITIAL-VAL

The data to be displayed in the field if not modified by the user or your program.

For a date field (Type property is "d"), this property can be set to "tt/tt/tt" (set FD-INITIAL-VAL to "tttttt" in your program) to indicate that today's date should be automatically displayed in the field.

For list boxes and combination boxes (Control type property is "l" or "o"), the data should be formatted into a list of values each with a length specified by the Item length property. For combination boxes, the first value is used to fill the entry field portion of the field so this value should be repeated in the list which follows. A blank entry can be inserted in a list box by setting the value to X"FF".

For a list box (Control type property is "l"), this property can also be set to a special value to cause the List Box to be automatically filled with specific data. These special values are as follows (the values must be in upper-case):

FILELIST -	a file listing will be displayed.
DIRLIST -	a directory listing will be displayed, including all available disk drives.
FONTLIST -	a list of available fonts will be displayed.
OCXLIST -	a list of installed activex controls will be displayed.
DRVLIST -	a list of all available disk drives will be displayed.
TEXTLIST -	the contents of a text file can be displayed - see below.

In the case of the file and directory listings, the actual directory used for the listing is dependent on the contents of the Fields area value for the List Box, as set by your program. If it is low-values, the current directory will be used; if it is set to a directory name, then that directory will be used. If a file or directory is selected by the user, the Fields area value will be modified to reflect the selection. See the sample program FILELIST for an example of using directory list boxes. If a drive is selected by the user, the selection will be returned as "c: ", "d: ", etc.

To display a text file, set Value to "TEXTLIST=myfile" where myfile is the name of a text file. Item length (derived from Format) must be at least as long as Value otherwise the editor will truncate the file name. Use the Fields area item to set and receive selections as for regular listboxes. Use GET-FIELD-DATA to retrieve multiple selections using multiple calls if necessary - see Chapter D5.

For an icon field (Control type property is "i" and Item length property is zero), this property is set as follows:

Bytes 1-8 =	text icon. This can be either text to display in the icon area or "NULL" to indicate that nothing should be displayed. Not used in a graphical environment.
Bytes 9-20 =	icon file (e.g. "myimage.bmp").
Bytes 21+ =	icon details. This varies depending on the type of icon, as follows:

For a bitmap icon (Type property is low-value or X"01"), icon details can contain a second bitmap filename which will be displayed when the user clicks on the icon. This can be used to simulate "pressing" the icon. If the icon file specified does not have an extension of "bmp", the dll named "sp2ima32.dll" will be loaded to process the non-bmp image. Support for this dll is contained in an add-on package - call Flexus for details of this.

For a bitmap button icon (Type property is X"02", X"03" or X"04"), icon details is used to store the number and width of the bitmap image to be used. If "1/16" is entered, the first 16 pixels (horizontally) in the bitmap file will be used. If "2/16" is entered, the second 16 pixels (horizontally) will be used. This allows multiple images to be stored in a single bitmap file which means quicker access and easier distribution. See the file "icons.bmp" for an example of this.

For an activex control icon (Type property is X"05"), icon details is used to store the activex control name and property and event data in the form "name/event/event/property=value/property=value/etc.". Also activex control icons are used to interface with additional control types such as win32 custom controls which are supported using external dll's - the same basic technique as activex support. The icon file name extension controls which dll will be used, as follows:

ocx/dll -	sp2ocx32.dll (activex controls)
lvw -	sp2lvw32.dll (listview controls)
tvw -	sp2tvw32.dll (treeview controls)

Support for these dll’s is contained in an add-on package - call Flexus for details of this. If icon file name is set to "nohatch", no control will be loaded but the hatch that represents the missing control will not be displayed - this is useful for fields that are used for controls that are not defined until run time.

If Item length is set for an Icon field then the Value property does not contain icon details but rather contains the text icon for 8 bytes followed by the icon file for Item length. If Type is X"02", the bitmap number and width (e.g. "1/16") is included as part of the icon file, following the name and separated by a blank. If Type is X"05", the activex control name, etc. are stored in a slot immediately following the file name.

If the Miscellaneous X"20" option is set (see field Miscellaneous property) as well as Item Length, and Type is X"02", then the Value property contains three slots (each Item length long) following the text icon:

Slot 1 =	icon file (default image)
Slot 2 =	icon file 2 (image to be displayed when the mouse is over the button)
Slot 3 =	icon file 3 (image to be displayed when the button is grayed out)

If the Miscellaneous X"20" option is set (see field Miscellaneous property) as well as Item Length, and Type is X"02" and Special format is "w" (text with image), then the Value property contains two slots (each Item length long) following the text icon:

Slot 1 =	text for button
Slot 2 =	icon file for background image

For a subwindow field (Control type property is "w"), Value is used to hold the name of a panel which will be displayed in a child or popup window.

For a multi-line system entry field, Value can be set to "TEXTFILE=myfile.txt" to automatically display a text file in the field.

Class

Program name: **FD-CLASS**

The name of a class defined externally in a CSS file referenced by the configuration variable SP2CSS. This class can be used to style text buttons (Control type = "i", Type = X"02", Special format = "t"). The following styles are supported:

font-family
font-size
font-weight
color
background-color
cursor
border-color
border-style
border-radius

The following selectors are supported:

:hover
:active
:disabled

The corresponding field properties (Font, etc.) should be set to default in order for the CSS styles to take effect.

Range

Program name: **FD-RANGE-VALS**

The range of values within which any data entered into the field must fall. The values are held in de-edited form. The bottom of the range is first for a length specified by the Program length property followed by the top of the range for the same length. Low-values means infinitely small or infinitely large depending on whether this value is first or second. This also controls the values that are displayed in a spin button (Usage option = "b").

Discrete values

Program name: **FD-DISC-VALS**

The list of values in which any data entered into the field must be included. Each value is held in the list in de-edited form for a length specified by the Program length property. For a Radio Button, the list should be just one value which is the value to be returned when the Radio Button is selected. For a Check Box, the list should be two items, the first value being the value to be returned when the Check Box is selected, the second when it is de-selected.

Message text

Program name: **FD-MSG-TEXT**

The text that is to be displayed in the message line or status bar of the panel window. This text will only be displayed if a message line exists for this window (see the panel Message length property).

For a windowed icon (Control type property is "i" and Usage option property is "w"), the text will be displayed as an icon hint which will appear automatically when the mouse pointer is moved over the icon. This also applies to any system field that has the Message as hint property set (Options-3 X"08"). If the text is set to the string "$return", a hint will not be displayed but rather control will be returned to the program with key set to KEY-MOUSE-MOVE. LAST-FLD-ID will either identify the field if the mouse is entering the field or be set to zero to indicate that it is leaving the field.

User data

Program name: **FD-USER-DATA**

Allows you to store your own data within the field definition. This data must be updated using the SET-FIELD-DEF function and retrieved using the GET-FIELD-DEF function.

If panel Options-5 property is set to X"20" then User data is formatted in a special way for use by the code generator (all values are delimited by the @ character):

1. Program item holding field data
2. Program item holding type (protection) data
3. Not used
4. Paragraph to be performed when exception occurs
5. Paragraph to be performed when event occurs
6. Paragraph to be performed on leaving field
7. Paragraph to be performed on entering field
8. Program item holding visibility state
9. Data columns within program data for list box
10. Popup menu to be displayed

Help keyword

Program name: **FD-HELP-KEYWORD**

The keyword used in conjunction with standard help (see chapter B32). This takes priority over the implied keyword held in FD-NAME. The keyword can be prefixed with the name of the help file to be used, enclosed within forward slashes. For example "/myhelp/help-for-account/" would cause a file named "myhelp.hlp" to be searched for a keyword of "help-for-account". If this property is set to "HELP_CONTENTS", the contents page of the help file will be displayed. If the keyword text is prefixed with a "!", it will be interpreted as an application to run, for example:

```
!winhlp32 -P -Icontext-string help-file
```

If the keyword text is prefixed with an "@", it will be interpreted as a macro to run, for example:

```
@JumpId(`help-file',`context-string')
```

If the field Options-4 property is set to X"04" then the keyword text is assumed to be an address to be loaded into the browser when the field is clicked.

CHAPTER E7 - Static Field Properties

Overview

These properties define the appearance and behavior of a static field.

Static field properties are set and accessed programmatically either by using the SET-PROPERTY and GET-PROPERTY functions or by using data items in the STATIC-DEF parameter before a SET-STATIC-DEF call and after a GET-STATIC-DEF call. To set a property using SET-STATIC-DEF after a static field is already defined you would normally make a GET-STATIC-DEF call followed by a SET-STATIC-DEF call. The STATIC-DEF parameter is contained in the copy file SP2.CPY.

Parameter layout

```
01  SP2-STATIC-DEF.
      05  SP2-SD-RET-CODE         PIC S9(4) COMP-5.
      05  SP2-SD-LENS.
          10  SP2-SD-LEN-LEN      PIC S9(4) COMP-5 VALUE +10.
          10  SP2-SD-NUM-LEN      PIC S9(4) COMP-5 VALUE +12.
          10  SP2-SD-CHAR-LEN     PIC S9(4) COMP-5 VALUE +4.
          10  SP2-SD-VAR-LEN      PIC S9(4) COMP-5 VALUE +80.
          10  SP2-SD-TEXT-LEN     PIC S9(4) COMP-5 VALUE +80.
      05  SP2-SD-DATA.
 ******** SP2-SD-NUM-DATA ********
          10  SP2-SD-ID           PIC S9(4) COMP-5.
          10  SP2-SD-ROW          PIC S9(4) COMP-5.
          10  SP2-SD-COL          PIC S9(4) COMP-5.
          10  SP2-SD-WIDTH        PIC S9(4) COMP-5.
          10  SP2-SD-HEIGHT       PIC S9(4) COMP-5.
          10  SP2-SD-FONT-ID      PIC S9(4) COMP-5.
 ******** SP2-SD-CHAR-DATA *******
          10  SP2-SD-COLR         PIC X.
          10  SP2-SD-TYPE         PIC X.
          10  SP2-SD-JUSTIFY      PIC X.
          10  SP2-SD-MISC-OPTIONS PIC X.
 ******** SP2-SD-VAR-DATA ********
          10  SP2-SD-TEXT         PIC X(80).
```

SD-VAR-LEN is the length of SD-TEXT.
SD-TEXT-LEN is the length of SD-TEXT.

Id

Program name: **SD-ID**

The id for the static field. The panel editor uses the Row and Column properties to index static fields so this property is usually set to zero. Setting this property manually however, may make it easier to access a static field programmatically using GET-STATIC-DEF and SET-STATIC-DEF.

Row

Program name: **SD-ROW**

The row in cells within the panel where the static field is positioned. This property may be set directly in the panel editor or indirectly by selecting and dragging the static field.

Column

Program name: **SD-COLUMN**

The column in cells within the panel where the static field is positioned. This property may be set directly in the panel editor or indirectly by selecting and dragging the static field.

Width

Program name: **SD-WIDTH**

The width of the static field in 1/10 cells. This property may be set directly in the panel editor or indirectly by selecting and resizing the static field. If SD-WIDTH is set to 0 on a SET-STATIC-DEF call, Width will be calculated based on the value of the Text property.

Height

Program name: **SD-HEIGHT**

The height of the static field in 1/10 cells. This property may be set directly in the panel editor or indirectly by selecting and resizing the static field. If this property is set to 0, a default height will be used.

Font ID

Program name: **SD-FONT-ID**

The id of the font to be used for static text. Fonts can be established in the panel editor or by using the SET-FONT-DEF function. Zero means use the panel font.

Color

Program name: **SD-COLR**

The id of the color of the field. Colors can be established in the panel editor or by using the SET-COLOR-DEF function.

low-value = use panel color (panel Color property)

Type

Program name: **SD-TYPE**

The type of the static item. It can be one of the following:

low-value = regular text

'a' = aligned text. This causes the text to be positioned as if it were contained in a rectangular border, allowing the text to line up with an adjacent entry field with a border.

'l' = line. This causes a line to be drawn. If the Width property is less than the Height property, the line will be vertical for a length of Height; if the Height property is less than the Width property, the line will be horizontal for a length of Width. The first byte of the Text property must be set to one of the following:

X'DA' =	top left corner
X'C4' =	horizontal
X'BF' =	top right corner
X'B3' =	vertical
X'C0' =	bottom left
X'D9' =	bottom right
X'C3' =	left-T
X'B4' =	right-T
X'C2' =	top-T
X'C1' =	bottom-T
X'C5' =	cross

'm' = Multi-line text with word-wrap

Justify

Program name: **SD-JUSTIFY**

Specifies how text will be justified within a static. Right justified text is useful for right-aligning a series of labels adjacent to a corresponding series of left-aligned entry fields.

low-value =	no justification
l =	justify to the left
r =	justify to the right
c =	center horizontally
v =	center horizontally and vertically
j =	justify to the left and right

Miscellaneous

Program name: **SD-MISC-OPTIONS**

Controls various aspects of a static field's appearance. The following values are combined to form the final value:

X'01' **3d line**	draw a 3d line
X'20' **Recalulate width**	recalculate the width of the static every time it is displayed

Text

Program name: **SD-TEXT**

The text to be displayed in the static field. If the panel Options-4 X'04' switch is set on and "~" is included in the text, the following character will be underlined, indicating that this character is a mnemonic.

If the Type property is set to "l" (line), the first byte of the Text property should be set to one of the following:

X'DA' =	top left corner
X'C4' =	horizontal
X'BF' =	top right corner
X'B3' =	vertical
X'C0' =	bottom left
X'D9' =	bottom right
X'C3' =	left-T
X'B4' =	right-T
X'C2' =	top-T
X'C1' =	bottom-T
X'C5' =	cross

CHAPTER E8 - Group Properties

Overview

These properties define the appearance and behavior of a group.

Group properties are set and accessed programmatically either by using the SET-PROPERTY and GET-PROPERTY functions or by using data items in the GROUP-DEF parameter before a SET-GROUP-DEF call and after a GET-GROUP-DEF call. To set a property using SET-GROUP-DEF after a group is already defined you would normally make a GET-GROUP-DEF call followed by a SET-GROUP-DEF call. The GROUP-DEF parameter is contained in the copy file SP2.CPY.

Parameter layout

```
  01  SP2-GROUP-DEF.
      05  SP2-GD-RET-CODE          PIC S9(4) COMP-5.
      05  SP2-GD-LENS.
          10  SP2-GD-LEN-LEN       PIC S9(4) COMP-5 VALUE +8.
          10  SP2-GD-NUM-LEN       PIC S9(4) COMP-5 VALUE +14.
          10  SP2-GD-CHAR-LEN      PIC S9(4) COMP-5 VALUE +48.
          10  SP2-GD-VAR-LEN       PIC S9(4) COMP-5 VALUE +50.
          10  SP2-GD-ID-LEN        PIC S9(4) COMP-5 VALUE +20.
          10  SP2-GD-TITLE-LEN     PIC S9(4) COMP-5 VALUE +30.
      05  SP2-GD-DATA.
 ******** SP2-GD-NUM-DATA ********
          10  SP2-GD-ID            PIC S9(4) COMP-5.
          10  SP2-GD-ROW           PIC S9(4) COMP-5.
          10  SP2-GD-COL           PIC S9(4) COMP-5.
          10  SP2-GD-WIDTH         PIC S9(4) COMP-5.
          10  SP2-GD-HEIGHT        PIC S9(4) COMP-5.
          10  SP2-GD-FLD-CNT       PIC S9(4) COMP-5.
          10  SP2-GD-FONT-ID       PIC S9(4) COMP-5.
 ******** SP2-GD-CHAR-DATA *******
          10  SP2-GD-NAME          PIC X(30).
          10  SP2-GD-TAB-WITHIN    PIC X.
          10  SP2-GD-CUR-COLR      PIC X.
          10  FILLER               PIC X.
          10  SP2-GD-SELECT-TYPE   PIC X.
          10  FILLER               PIC X.
          10  FILLER               PIC X.
          10  FILLER               PIC X(3).
          10  FILLER               PIC X(3).
          10  FILLER               PIC X.
          10  SP2-GD-MISC-OPTIONS  PIC X.
          10  SP2-GD-COLR          PIC X.
          10  SP2-GD-ANCHOR        PIC X.
          10  SP2-GD-BOR-TYPE      PIC X.
          10  FILLER               PIC X.
 ******** SP2-GD-VAR-DATA ********
          10  SP2-GD-FLD-ID OCCURS 10
                                   PIC S9(4) COMP-5.
          10  SP2-GD-TITLE         PIC X(30).
```

GD-VAR-LEN is the combined length of all occurrences of GD-FLD-ID plus the length of GD-TITLE.

Id

Program name: GD-ID

The id of the group. Automatically set in the panel editor when a new group is defined. If GD-ID is set to zero before a SET-GROUP-DEF call, a new group will be created with an automatically assigned id. Otherwise, GD-ID should be set to the id of the group to be accessed by GET-GROUP-DEF or SET-GROUP-DEF.

Row

Program name: GD-ROW

The row in cells where the top of the group is located within the panel. May be set directly in the panel editor or indirectly by selecting and then dragging the group. The row does not allow for the group border (if present) which is drawn outside the actual group.

Column

Program name: GD-COL

The column in cells where the left of the group is located within the panel. May be set directly in the panel editor or indirectly by selecting and then dragging the group. The column does not allow for the group border (if present) which is drawn outside the actual group.

Width

Program name: GD-WIDTH

The width of the group in 1/10 cells. May be set directly in the panel editor or by selecting and dragging the left or right hand border of the group. The width does not allow for the group border (if present) which is drawn outside the actual group.

Height

Program name: GD-HEIGHT

The height of the group in 1/10 cells. May be set directly in the panel editor or by selecting and dragging the top or bottom border of the group. The height does not allow for the group border (if present) which is drawn outside the actual group.

Field count

Program name: GD-FLD-CNT

The number of fields included in the group. Set automatically by the panel editor when the group is sized based on the number of fields within the area occupied by the group.

Font ID

Program name: GD-FONT-ID

The id of the font to be used for title text. Fonts can be established in the panel editor or by using the SET-FONT-DEF function. Zero means use the panel font.

Name

Program name: GD-NAME

The name of the group.

Tab within

Program name: GD-TAB-WITHIN

Controls how the user can move the cursor within the group.

low-value =	tab key can be used to move within group
n =	tab key used to exit the group, use arrow keys to move within group
a =	tab key or arrow keys can be used to move within group

Current color

Program name: **GD-CUR-COLR**

Id of color used to highlight the group if the Select type property is "c" or "t" or "h". If Current color is set to low-value, the default highlight color will be used.

Select type

Program name: **GD-SELECT-TYPE**

Specifies the functionality of the group.

low-value =	no special action
a =	auto-select, field within group will be automatically selected as soon as the cursor is moved to it and all other fields in the group will be de-selected (normally used for Radio Buttons - if other fields are included in the group make sure all fields have Program data set to X"10" otherwise the copy file will be generated incorrectly.)
s =	select, all other fields within will be de-selected when a field is selected (normally used for Radio Buttons)
m =	multiple-select, multiple fields in the group may be selected, fields are de-selected by selecting again (normally used for Check Boxes)
c =	color if current, fields within the group will all be colored with the highlight color when any field in the group becomes the current field
t =	color contiguous, same as "c" except that any blanks between the fields are also colored.
h =	same as "t," except that the color will be held even when the user tabs out of the group
n =	no member, useful for drawing boxes without having any effect on fields and other groups lying within the boundaries of the box.

Title color

Program name: **GD-TITLE-COLR**

Id of color of group title.

low-value = use panel color (panel Color property)

Miscellaneous

Program name: **GD-MISC-OPTIONS**

Miscellaneous options for a group.

X'02' **Shift title up** =	adjust position of title
X'04' **Title in border** =	make sure title overlays border

Color

Program name: **GD-COLR**

The id of the color of the group background.

low-value = use panel color (panel Color property)

Anchor

Program name: **GD-ANCHOR**

Causes a group to change size or position relative to the containing window size. The following values are combined to form the final value:

X'01' =	change size relative to right window border
X'02' =	change size relative to bottom window border
X'04' =	free left allows a group anchored to the right (X'01') to move horizontally rather than change size
X'08' =	free top allows a group anchored to the bottom (X'02') to move vertically rather than change size

Border type

Program name: **GD-BOR-TYPE**

The border to be drawn around the group.

low-value =	no border
l (lower-case L) =	default border
3 =	3d border
t =	thin 3d border. This border is only one pixel wide and is therefore particularly useful if you are cramped for space.
P =	system 3d group border (like system entry field)
a =	single line border (uses Color property)
A =	thick line border (uses Color property)
b =	raised border
B =	raised and thick
c =	lowered border
C =	lowered and thick
d =	engraved border
D =	engraved and thick
e =	rimmed border
E =	rimmed and thick
f =	flat border
g =	rounded border
h =	flat rounded

Field ids

Program name: **GD-FLD-ID**

The ids of the fields included in the group. Set automatically by the panel editor when the group is sized based on the fields within the area occupied by the group.

Title

Program name: **GD-TITLE**

The text to appear above the group describing its contents.

CHAPTER E9 - Repeat Group Properties

Overview

These properties define the appearance and behavior of a repeat group.

Repeat group properties are set and accessed programmatically either by using the SET-PROPERTY and GET-PROPERTY functions or by using data items in the REPEAT-DEF parameter before a SET-REPEAT-DEF call and after a GET-REPEAT-DEF call. To set a property using SET-REPEAT-DEF after a repeat group is already defined you would normally make a GET-REPEAT-DEF call followed by a SET-REPEAT-DEF call. The REPEAT-DEF parameter is contained in the copy file SP2.CPY.

Parameter layout

```
 01  SP2-REPEAT-DEF.
     05  SP2-RD-RET-CODE         PIC S9(4) COMP-5.
     05  SP2-RD-LENS.
         10  SP2-RD-LEN-LEN      PIC S9(4) COMP-5 VALUE +10.
         10  SP2-RD-NUM-LEN      PIC S9(4) COMP-5 VALUE +44.
         10  SP2-RD-CHAR-LEN     PIC S9(4) COMP-5 VALUE +4.
         10  SP2-RD-VAR-LEN      PIC S9(4) COMP-5 VALUE +104.
         10  SP2-RD-BASE-LEN     PIC S9(4) COMP-5 VALUE +20.
     05  SP2-RD-DATA.
******** SP2-RD-NUM-DATA ********
         10  SP2-RD-ID           PIC S9(4) COMP-5.
         10  SP2-RD-ROW          PIC S9(4) COMP-5.
         10  SP2-RD-COL          PIC S9(4) COMP-5.
         10  SP2-RD-WIDTH        PIC S9(4) COMP-5.
         10  SP2-RD-HEIGHT       PIC S9(4) COMP-5.
         10  SP2-RD-VERT-OCC     PIC S9(4) COMP-5.
         10  SP2-RD-VERT-VIS     PIC S9(4) COMP-5.
         10  SP2-RD-VERT-GAP     PIC S9(4) COMP-5.
         10  SP2-RD-VERT-DISP    PIC S9(4) COMP-5.
         10  SP2-RD-VERT-SHIFT   PIC S9(4) COMP-5.
         10  SP2-RD-VERT-BAR-ID  PIC S9(4) COMP-5.
         10  SP2-RD-HOR-OCC      PIC S9(4) COMP-5.
         10  SP2-RD-HOR-VIS      PIC S9(4) COMP-5.
         10  SP2-RD-HOR-GAP      PIC S9(4) COMP-5.
         10  SP2-RD-HOR-DISP     PIC S9(4) COMP-5.
         10  SP2-RD-HOR-SHIFT    PIC S9(4) COMP-5.
         10  SP2-RD-HOR-BAR-ID   PIC S9(4) COMP-5.
         10  SP2-RD-LAST-OCCURS  PIC S9(4) COMP-5.
         10  SP2-RD-BEG-LINE-NUM PIC S9(4) COMP-5.
         10  SP2-RD-PROG-NUM     PIC S9(4) COMP-5.
         10  SP2-RD-PROG-LEN     PIC S9(4) COMP-5.
         10  SP2-RD-FLD-CNT      PIC S9(4) COMP-5.
******** SP2-RD-CHAR-DATA *******
         10  SP2-RD-BOR-TYPE     PIC X.
         10  SP2-RD-TAB-SW       PIC X.
         10  SP2-RD-MISC-OPTIONS PIC X.
         10  SP2-RD-ANCHOR       PIC X.
******** SP2-RD-VAR-DATA ********
         10  SP2-RD-BASE-ID OCCURS 10
                                 PIC S9(4) COMP-5.
         10  SP2-RD-FILE-IND               PIC S9(4) COMP-5.
         10  SP2-RD-FILE-LEN     PIC S9(4) COMP-5.
         10  SP2-RD-FILE-NAME    PIC X(80).
```

RD-VAR-LEN is the combined length of all occurrences of RD-BASE-ID, RD-FILE-IND, RD-FILE-LEN and RD-FILE-NAME. RD-BASE-LEN is the combined length of all occurrences of RD-BASE-ID.

Id

Program name: **RD-ID**

The id of the repeat group. Automatically set in the panel editor when a new repeat group is defined. If RD-ID is set to zero before a SET-REPEAT-DEF call, a new repeat group will be created with an automatically assigned id. Otherwise, RD-ID should be set to the id of the repeat group to be accessed by GET-GROUP-DEF or SET-GROUP-DEF.

Row

Program name: **RD-ROW**

The row in cells within the panel where the top of the repeat group is located. May be set directly in the panel editor or indirectly by selecting and dragging the repeat group. The row does not allow for the border (if present) which is drawn outside the actual repeat group.

Column

Program name: **RD-COL**

The column in cells within the panel where the left of the repeat group is located. May be set directly in the panel editor or indirectly by selecting and dragging the repeat group. The column does not allow for the border (if present) which is drawn outside the actual repeat group.

Width

Program name: **RD-WIDTH**

The width of the repeat group in 1/10 cells. May be set directly in the panel editor or indirectly by selecting and then dragging the left or right hand border of the repeat group. The width does not allow for the border (if present) which is drawn outside the actual repeat group.

Height

Program name: **RD-HEIGHT**

The height of the repeat group in 1/10 cells. May be set directly in the panel editor or indirectly by selecting and then dragging the top or bottom border of the repeat group. The height does not allow for the border (if present) which is drawn outside the actual repeat group.

Vertical occurs

Program name: **RD-VERT-OCC**

The total number of vertical occurrences within the repeat group. Not all the occurrences need be visible.

Vertical visible

Program name: **RD-VERT-VIS**

The number of occurrences that are visible. Set automatically when a repeat group is created based on its size.

Vertical gap

Program name: **RD-VERT-GAP**

The vertical gap between the occurrences. The unit for this gap is normally the number of cells (vertically) occupied by the first field in the repeat group so that the row incrementor is calculated as (field Height / 10) * (Vertical gap + 1). This unit may be changed to single cells by setting the repeat group Miscellaneous property so that the row incrementor is calculated as (field Height / 10 + Vertical gap).

The default gap is zero (consecutive rows) but this may be adjusted automatically if the fields within the repeat group are positioned on different rows.

Vertical gap can be set to -1 which causes the borders of fields on each row to overlap each other. This feature can be used to create a simple grid – see field Border type for more details on this.

Vertical displacement

Program name: **RD-VERT-DISP**

The offset of the first visible occurrence in the repeat group, i.e. occurrence - 1.

Vertical shift

Program name: **RD-VERT-SHIFT**

The number of occurrences to scroll when the Scroll Up or Scroll Down keys are pressed. If this item is zero, the Vertical visible property will be used for this value.

Vertical scrollbar

Program name: **RD-VERT-BAR-ID**

The id of the vertical Scroll Bar included in the group. Automatically set in the panel editor when the repeat group is sized if there is a Scroll Bar within the area occupied by the repeat group.

Horizontal occurrences

Program name: **RD-HOR-OCC**

The total number of horizontal occurrences in the repeat group. Unlike vertical occurrences, all these horizontal occurrences must be visible. There is no check to see that all the occurrences fit within the bounds of the repeat group.

Horizontal gap

Program name: **RD-HOR-GAP**

The number of cells between horizontal occurrences in the repeat group. If this is set to zero, the occurrences will be adjacent to each other.

Horizontal gap can be set to -1 which causes the borders of fields in each column to overlap each other. This feature can be used to create a simple grid – see field Border type for more details on this.

Horizontal displacement

Program name: **RD-HOR-DISP**

The offset in characters of the first visible column in the repeat group.

Cursor movement

Program name: **RD-HOR-SHIFT**

Controls horizontal scrolling (only supported if repeat group contains one field):

0 -	no horizontal scrolling
-1 -	horizontal scrolling
-2 -	hold column when moving cursor up and down

Last occurs

Program name: **RD-LAST-OCCURS**

Used internally.

Program field length

Program name: **RD-PROG-LEN**

The total length of the fields in a single vertical occurrence based on the Program length property of each field. Automatically set when a repeat group is defined.

Field count

Program name: **RD-FLD-CNT**

The number of base fields included in the repeat group. Set automatically by the panel editor when the repeat group is sized based on the number of fields within the area occupied by the repeat group.

Border type

Program name: **RD-BOR-TYPE**

The border to be drawn around the repeat group:

low-value =	no border
l (lower-case L) =	default border
f =	flat border
3 =	3-dimensional border
t =	thin 3d border. This border is only one pixel wide and is therefore particularly useful if you are cramped for space.

Tabbing type

Program name: **RD-TAB-SW**

Controls tabbing within Repeat Groups:

low-value =	tab one row then exit repeat group
v =	tab through visible rows then exit
a =	tab through all rows (scrolling if necessary)

Miscellaneous

Program name: **RD-MISC-OPTIONS**

Controls repeat group behavior:

X'01' **Vertical gap in cells** -	see Vertical gap property
X'02' **Thread scrolling** -	enhanced program interaction if sp2thred.dll also used
X'04' **Click outside column**-	allow click outside column (space between columns if present) to select row in repeat group
X'08' **Disable repeat**-	disable whole repeat group
X'10' **Use window scrollbar**-	specifies that this repeat group should be controlled by the window scrollbar if Vertical scroll repeat is set

Anchor

Program name: **RD-ANCHOR**

Causes a repeat group to change size or position relative to the containing window size. The following values are combined to form the final value:

X'01' =	change size relative to right window border
X'02' =	change size relative to bottom window border
X'04' =	free left allows a repeat group anchored to the right (X'01') to move horizontally rather than change size
X'08' =	free top allows a repeat group anchored to the bottom (X'02') to move vertically rather than change size

Avoid setting the Bottom anchor for fields in a repeat group otherwise the field occurrences will most likely overlay each other.

Field ids

Program name: **RD-BASE-ID**

The ids of the fields included in the repeat group. Set automatically by the panel editor when the repeat group is sized based on the fields within the area occupied by the group.

File indicator

Program name: **RD-FILE-IND**

Switch to indicate that this repeat group is to display a text file (set switch to –1). Set automatically in the panel editor when you set the File name property.

On a SET-FIELD-DEF call, if RD-FILE-IND is not -1 but RD-VAR-LEN is greater than RD-BASE-LEN, the area after RD-BASE-ID's is assumed to contain initial values for the fields contained in the repeat group. The area is laid out as follows:

```
10  RD-INITIAL-COUNTS.
    15  FILLER PIC S9(4) COMP-5 VALUE count-base-id-1-vals.
    15  FILLER PIC S9(4) COMP-5.VALUE count-base-id-2-vals.
    ...
10  RD-INITIAL-LENS.
    15  FILLER PIC S9(4) COMP-5 VALUE len-base-id-1-val.
    15  FILLER PIC S9(4) COMP-5 VALUE len-base-id-2-val.
    ...
10  RD-INITIAL-VALS.
    15  FILLER PIC X(len-base-id-1-val) OCCURS count-base-id-1-vals.
    15  FILLER PIC X(len-base-id-2-val) OCCURS count-base-id-2-vals.
    ...
```

The length of RD-INITIAL-COUNTS and RD-INITIAL-LENS is RD-BASE-LEN.

File length

Program name: **RD-FILE-LEN**

The length of the name of the text file to be displayed. Set automatically in the panel editor when you set the File name property.

File name

Program name: **RD-FILE-NAME**

The name of the text file to be displayed.

CHAPTER E10 - Font Properties

Overview

These properties define the appearance of a font.

Font properties are set in the panel editor by displaying the font selection list for a panel, field or static field Font property and then clicking the right mouse button or pressing F2.

Font properties are set and accessed programmatically using data items in the FONT-DEF parameter before a SET-FONT-DEF call and after a GET-FONT-DEF call. To set a property after a font is already defined you would normally make a GET-FONT-DEF call followed by a SET-FONT-DEF call. The FONT-DEF parameter is contained in the copy file SP2.CPY.

The FONT-DEF parameter is also used on the QUERY-FONT function call.

Parameter layout

```
01  SP2-FONT-DEF.
      05  SP2-FO-RET-CODE         PIC S9(4) COMP-5.
      05  SP2-FO-LENS.
          10  SP2-FO-LEN-LEN      PIC S9(4) COMP-5 VALUE +8.
          10  SP2-FO-NUM-LEN      PIC S9(4) COMP-5 VALUE +16.
          10  SP2-FO-CHAR-LEN     PIC S9(4) COMP-5 VALUE +6.
          10  SP2-FO-VAR-LEN      PIC S9(4) COMP-5 VALUE +30.
      05  SP2-FO-DATA.
 ******** SP2-FO-NUM-DATA ********
          10  SP2-FO-ID           PIC S9(4) COMP-5.
          10  SP2-FO-WIDTH        PIC S9(4) COMP-5.
          10  SP2-FO-HEIGHT       PIC S9(4) COMP-5.
          10  SP2-FO-GUI-ID       PIC S9(4) COMP-5.
          10  SP2-FO-GUI-ID-2     PIC S9(4) COMP-5.
          10  SP2-FO-DECIPOINTS   PIC S9(4) COMP-5.
          10  FILLER              PIC S9(4) COMP-5.
          10  SP2-FO-ROTATION     PIC S9(4) COMP-5.
 ******** SP2-FO-CHAR-DATA *******
          10  SP2-FO-PITCH        PIC X.
          10  SP2-FO-WEIGHT       PIC X.
          10  SP2-FO-ITALIC       PIC X.
          10  SP2-FO-STRIKE-OUT   PIC X.
          10  SP2-FO-UNDERLINE    PIC X.
          10  SP2-FO-CHAR-SET     PIC X.
 ******** SP2-FO-VAR-DATA ********
          10  SP2-FO-NAME         PIC X(30).
```

FO-VAR-LEN is the length of FO-NAME.

Id

Program name: **FO-ID**

The id of the font. Set in the panel editor when a new font is defined (see Overview above). If FO-ID is set to zero before a SET-FONT-DEF call, a new font will be created with an automatically assigned id. Otherwise, FO-ID should be set to the id of the font to be accessed by GET-FONT-DEF or SET-FONT-DEF.

Width

Program name: **FO-WIDTH**

The average width of a character in the font in pixels. Set in the panel editor based on the point size of the font defined (see Overview above). If set to zero on a SET-FONT-DEF call, the default width for the font will be used.

Height

Program name: **FO-HEIGHT**

The height of a character in the font in pixels. Set in the panel editor based on the point size of the font defined (see Overview above).

GUI id

Program name: **FO-GUI-ID**

The first part of the handle used by the operating system to identify the field. If FO-GUI-ID and FO-GUI-ID-2 are set to zero on a SET-FONT-DEF call, the existing Windows font will be deleted. This must be done if you want to change a font that has already been created.

GUI id (2)

Program name: **FO-GUI-ID-2**

The second part of the handle used by the operating system to identify the field. If FO-GUI-ID and FO-GUI-ID-2 are set to zero on a SET-FONT-DEF call, the existing Windows font will be deleted. This must be done if you want to change a font that has already been created.

Decipoints

Program name: **FO-DECIPOINTS**

The height of the font selected in 1/720 inches. This property is read-only and only accessible after a QUERY-FONT call.

Rotation

Program name: **FO-ROTATION**

The number of degrees the font is to be rotated counter clockwise.

Pitch

Program name: **FO-PITCH**

The pitch of the font. Set in the panel editor when a new font is defined (see Overview above).

low-value =	variable pitch
f =	fixed pitch

Weight

Program name: **FO-WEIGHT**

The weight of the font. Set in the panel editor when a new font is defined (see Overview above).

low-value =	normal
b =	bold

Italic

Program name: **FO-ITALIC**

Specifies whether the font is italicized. Set in the panel editor when a new font is defined (see Overview above).

low-value =	normal
y =	italic

Strike out

Program name: **FO-STRIKE-OUT**

Specifies whether the font is struck out with a horizontal line. Set in the panel editor when a new font is defined (see Overview above).

low-value =	normal
y =	struck out

Underline

Program name: **FO-UNDERLINE**

Specifies whether the font is underlined. Set in the panel editor when a new font is defined (see Overview above).

low-value =	normal
y =	underlined

Character set

Program name: **FO-CHAR-SET**

Specifies the character set used by the font. Set in the panel editor when a new font is defined (see Overview above).

low-value = ANSI character set

Prior to a QUERY-FONT call, FO-CHAR-SET may be set to something other than low-value to cause non-ANSI fonts to be listed. On return, FO-CHAR-SET will be reset to low-value if an ANSI font was selected. Examples of non-ANSI fonts are the symbol fonts and also some international fonts. If a font is non-ANSI, it will be flagged on the font list dialog box in the "Ch" (non-ANSI Character set) column.

If you are using SP2IBM=1 (see appendix B), do not use characters outside the 32-127 range when using a non-ANSI font.

Name

Program name: **FO-NAME**

The name of the font type face such as "Helvetica", "Times Roman", or "Courier". Set in the panel editor when a new font is defined (see Overview above). If FO-NAME is blank on a SET-FONT-DEF call, the system font will be selected.

CHAPTER E11 - Color Properties

Overview

These properties define the appearance of a color.

Color properties are set in the panel editor by displaying the color selection list for a panel, field or static field Color property and then clicking the right mouse button or pressing F2.

Color properties are set and accessed programmatically using data items in the COLOR-DEF parameter before a SET-COLOR-DEF call and after a GET-COLOR-DEF call. To set a property after a font is already defined you would normally make a GET-COLOR-DEF call followed by a SET-COLOR-DEF call. The COLOR-DEF parameter is contained in the copy file SP2.CPY.

The COLOR-DEF parameter is also used on the QUERY-COLOR function call.

The first occurrence of SP2-CO-FG-BG defines the foreground color, the second occurrence defines the background color and the third occurrence (oprional) defines the ending gradient color.

Parameter layout

```
 01  SP2-COLOR-DEF.
     05  SP2-CO-RET-CODE         PIC S9(4) COMP-5.
     05  SP2-CO-LENS.
         10  SP2-CO-LEN-LEN      PIC S9(4) COMP-5 VALUE +8.
         10  SP2-CO-NUM-LEN      PIC S9(4) COMP-5 VALUE +4.
         10  SP2-CO-CHAR-LEN     PIC S9(4) COMP-5 VALUE +16.
         10  SP2-CO-VAR-LEN      PIC S9(4) COMP-5 VALUE +30.
     05  SP2-CO-DATA.
******** SP2-CO-NUM-DATA ********
         10  SP2-CO-ID           PIC S9(4) COMP-5.
         10  SP2-CO-FONT-ID      PIC S9(4) COMP-5.
******** SP2-CO-CHAR-DATA *******
         10  SP2-CO-FG-BG OCCURS 3.
             15  SP2-CO-NUM      PIC X.
             15  SP2-CO-TYPE     PIC X.
             15  SP2-CO-SYSTEM   PIC X.
             15  SP2-CO-TEXT     PIC X.
             15  SP2-CO-RED      PIC X.
             15  SP2-CO-GREEN    PIC X.
             15  SP2-CO-BLUE     PIC X.
             15  FILLER          PIC X.
******** SP2-CO-VAR-DATA ********
         10  SP2-CO-NAME         PIC X(30).
```

CO-VAR-LEN is the length of CO-NAME.

Id

Program name: **CO-ID**

The id of the color. Set in the panel editor when a new color is defined (see Overview above). If CO-ID is set to zero before a SET-COLOR-DEF call, a new color will be created with an automatically assigned id. Otherwise, CO-ID should be set to the id of the color to be accessed by GET-COLOR-DEF or SET-COLOR-DEF.

Font id

Program name: **CO-FONT-ID**

The id of a font. This allows a program Colors area item to be used to change the font of a field at run time. This property has no effect except used in this way. This might be used, for example, to set the font of a field in a row of a repeat group without effecting other rows.

Number

Program name: **CO-NUM**

Used internally. The first occurrence of color properties refers to the foreground color, the second occurrence refers to the background color and the third occurrence refers to the ending gradient color.

Type

Program name: **CO-TYPE**

The type of color being used. Set in the panel editor when a new color is defined (see Overview above).

low-value =	standard system color
t =	equivalent text-mode color
r =	defined using red, green and blue values
g =	for a background color (second occurrence), specifies that this is a starting gradient color
h =	for an ending gradient color (third occurrence), specifies that this is a horizontal gradient
v =	for an ending gradient color (third occurrence), specifies that this is a vertical gradient

System color

Program name: **CO-SYSTEM**

The system color being used, if the Type property is low-value. Set in the panel editor when a new color is defined (see Overview above).

Low-value =	default color
x'01' =	default text
x'02' =	default window
x'03' =	highlight text
x'04' =	highlight
x'05' =	menu text
x'06' =	menu
x'07' =	button text
x'08' =	button
x'09' =	button highlight
x'0A' =	button shadow
x'0B' =	grayed text
x'0C' =	transparent background

Text color

Program name: **CO-TEXT**

The text color being used, if the Type property is "t". Set in the panel editor when a new color is defined (see Overview above).

X'00' =	Black	X'08' =	dark grey
X'01' =	Blue	X'09' =	light blue
X'02' =	green	X'0a' =	light green
X'03' =	cyan	X'0b' =	light cyan
X'04' =	red	X'0c' =	pink
X'05' =	magenta	X'0d' =	light magenta
X'06' =	brown	X'0e' =	yellow
X'07' =	grey	X'0f' =	white

Red content

Program name: **CO-RED**

The red content of the color, if the Type property is "r", "g", "h" or "v". The value of the Red property can range from X'00' to X'ff'. Set in the panel editor when a new color is defined (see Overview above).

Green content

Program name: **CO-GREEN**

The green content of the color, if the Type property is "r", "g", "h" or "v". The value of the Green property can range from X'00' to X'ff'. Set in the panel editor when a new color is defined (see Overview above).

Blue content

Program name: **CO-BLUE**

The blue content of the color, if the Type property is "r", "g", "h" or "v". The value of the Blue property can range from X'00' to X'ff'. Set in the panel editor when a new color is defined (see Overview above).

Name

Program name: **CO-NAME**

The name of the color. This makes RGB colors easier to identify in the color selection list in the panel editor. Set in the panel editor when a new color is defined (see Overview above).

In Unix text mode, the name can be set to special values which indicate blinking and/or underlining if these features are supported by the system:

B-K	bkinking
U-L	underlining
B-U	blinking and underlining
U-B	blinking and underlining

CHAPTER E12 - Menu Properties

Overview

These properties define the appearance and behavior of a menu.

Menu properties are set in the panel editor by displaying the menu definition dialogbox. This can be done by clicking on the Menu icon or by selecting the panel Menu property to display the menu list and then clicking the right mouse button or pressing F2.

Menu properties are set and accessed programmatically using data items in the MENU-DEF parameter before a SET-MENU-DEF call and after a GET-MENU-DEF call. Individual menu option properties are set and accessed using data items in the MENU-OPTION parameter before a SET-MENU-OPTION call and after a GET-MENU-OPTION call. To set a menu option property after a menu is already defined you would normally make a GET-MENU-OPTION call followed by a SET-MENU-OPTION call. The MENU-DEF and MENU-OPTION parameters are contained in the copy file SP2.CPY.

Parameter layouts

```
 01  SP2-MENU-DEF.
     05  SP2-MD-RET-CODE         PIC S9(4) COMP-5.
     05  SP2-MD-LENS.
         10  SP2-MD-LEN-LEN      PIC S9(4) COMP-5 VALUE +18.
         10  SP2-MD-NUM-LEN      PIC S9(4) COMP-5 VALUE +2.
         10  SP2-MD-CHAR-LEN     PIC S9(4) COMP-5 VALUE +18.
         10  SP2-MD-VAR-LEN      PIC S9(4) COMP-5 VALUE +6800.
         10  SP2-MD-OPTN-LEN     PIC S9(4) COMP-5 VALUE +6.
         10  SP2-MD-OPTC-LEN     PIC S9(4) COMP-5 VALUE +2.
         10  SP2-MD-OPTV-LEN     PIC S9(4) COMP-5 VALUE +60.
         10  SP2-MD-NAME-LEN     PIC S9(4) COMP-5 VALUE +30.
         10  SP2-MD-TEXT-LEN     PIC S9(4) COMP-5 VALUE +30.
     05  SP2-MD-DATA.
******** SP2-MD-NUM-DATA ********
         10  SP2-MD-OPTION-CNT   PIC S9(4) COMP-5.
******** SP2-MD-CHAR-DATA *******
         10  SP2-MD-NAME         PIC X(8).
         10  SP2-MD-DRAW-SW      PIC X.
         10  FILLER              PIC X.
         10  FILLER              PIC X.
         10  FILLER              PIC X.
         10  FILLER              PIC X.
         10  FILLER              PIC X.
         10  FILLER              PIC X.
         10  FILLER              PIC X.
         10  FILLER              PIC X.
         10  FILLER              PIC X.
******** SP2-MD-VAR-DATA ********
         10  SP2-MD-OPTION OCCURS 100.
******** SP2-MD-OPTN-DATA *******
             15  SP2-MDO-ID      PIC S9(4) COMP-5.
             15  SP2-MDO-OWNR-ID PIC S9(4) COMP-5.
             15  SP2-MDO-ACC-KEY PIC S9(4) COMP-5.
******** SP2-MD-OPTC-DATA *******
             15  SP2-MDO-TYPE    PIC X.
             15  SP2-MDO-STATE   PIC X.
******** SP2-MD-NAME-DATA *******
             15  SP2-MDO-NAME    PIC X(30).
******** SP2-MD-TEXT-DATA *******
             15  SP2-MDO-TEXT    PIC X(30).
```

```
    01  SP2-MENU-OPTION.
        05  SP2-MO-RET-CODE         PIC S9(4) COMP-5.
        05  SP2-MO-LENS.
            10  SP2-MO-LEN-LEN      PIC S9(4) COMP-5 VALUE +12.
            10  SP2-MO-NUM-LEN      PIC S9(4) COMP-5 VALUE +2.
            10  SP2-MO-CHAR-LEN     PIC S9(4) COMP-5 VALUE +18.
            10  SP2-MO-VAR-LEN      PIC S9(4) COMP-5 VALUE +60.
            10  SP2-MO-NAME-LEN     PIC S9(4) COMP-5 VALUE +30.
            10  SP2-MO-TEXT-LEN     PIC S9(4) COMP-5 VALUE +30.
        05  SP2-MO-DATA.
   ******** SP2-MO-NUM-DATA ********
            10  SP2-MO-ID           PIC S9(4) COMP-5.
            10  SP2-MO-OWNR-ID      PIC S9(4) COMP-5.
            10  SP2-MO-ACC-KEY      PIC S9(4) COMP-5.
   ******** SP2-MO-CHAR-DATA *******
            10  SP2-MO-TYPE         PIC X.
            10  SP2-MO-STATE        PIC X.
   ******** SP2-MO-VAR-DATA ********
            10  SP2-MO-NAME         PIC X(30).
            10  SP2-MO-TEXT         PIC X(30).
```

MD-VAR-LEN is the combined lengths of all occurrences of MD-OPTION.
MD-OPTV-LEN is the combined lengths of MDO-NAME and MDO-TEXT in an occurrence of MD-OPTION.
MD-NAME-LEN is the length of MDO-NAME in an occurrence of MD-OPTION.
MD-TEXT-LEN is the length of MDO-TEXT in an occurrence of MD-OPTION.
MO-VAR-LEN is the combined lengths of MO-NAME and MO-TEXT.
MO-NAME-LEN is the length of MO-NAME.
MO-TEXT-LEN is the length of MO-TEXT.

Option count

Program name: **MD-OPTION-CNT**

The count of options in the menu. Set automatically by the panel editor after a menu has been defined.

Name

Program name: **MD-NAME**

The name of the menu. Generated automatically by the panel editor and then used to set the panel Menu property.

Draw switch

Used internally.

Id

Program name: **MDO-ID, MO-ID**

The id assigned to a menu option. Set in the panel editor using the menu definition dialogbox (see Overview above). The Converse panel Menu id property will be set to this if a menu option is selected at runtime.

Owner id

Program name: **MDO-OWNR-ID, MO-OWNR-ID**

The id of the submenu if this option is on a submenu. Set automatically in the panel editor when a submenu option is defined using the menu definition dialogbox (see Overview above).

Key

Program name: **MDO-ACC-KEY, MO-ACC-KEY**

The accelerator key that can be used by the user to select this option without going through the menu. Set in the panel editor using the menu definition dialogbox (see Overview above). When this key is pressed by the user, the Converse panel Menu id property will be set as if the menu option itself had been selected

Type

Program name: **MDO-TYPE, MO-TYPE**

The type of the menu option. Set automatically by the panel editor as options are defined using the menu definition dialogbox (see Overview above).

low-value =	regular option
s =	submenu option. Id of this option will be the Owner id of items on the submenu. In the menu definition dialogbox, key "=" to indicate that an option belongs to a submenu.
l (lower-case L) =	line separator. In the menu definition dialogbox, key "-" to indicate a line separator.
o =	edit override menu option. The user will be able to select this type of option even if edit errors exist in the current panel's data. This is useful for menu options that allow the user to exit without saving his or her work. In the menu definition dialogbox, click the "Override" button to indicate this type.

State

Program name: **MDO-STATE, MO-STATE**

The state of the menu option. Can only be set programmatically.

g =	greyed out
c =	checked
a =	on a SET-MENU-OPTION call, this causes a new menu option to be added even if an option with this idea already exists. This allows multiple options to return the same id when selected.

Option Name

Program name: **MDO-NAME, MO-NAME**

The name of the menu option. Set in the panel editor using the menu definition dialogbox. The Converse panel Menu option property will be set to this if a menu option is selected at runtime.

Text

Program name: **MDO-TEXT, MO-TEXT**

The text of the menu option. Set in the panel editor using the menu definition dialogbox (see Overview above). The "~" or "&" character is used to indicate that the character following is a mnemonic.

CHAPTER E13 - Message Properties

Overview

These properties define the appearance and behavior of a message box..

Message properties are only accessed programmatically, using data items in the MESSAGE-DATA parameter when a message box is displayed using the DISPLAY-MESSAGE function.. The MESSAGE-DATA parameter is contained in the copy file SP2.CPY.

Parameter layout

```
 01  SP2-MESSAGE-DATA.
     05  SP2-MS-RET-CODE         PIC S9(4) COMP-5.
     05  SP2-MS-LENS.
         10  SP2-MS-LEN-LEN      PIC S9(4) COMP-5 VALUE +12.
         10  SP2-MS-NUM-LEN      PIC S9(4) COMP-5 VALUE +2.
         10  SP2-MS-CHAR-LEN     PIC S9(4) COMP-5 VALUE +4.
         10  SP2-MS-VAR-LEN      PIC S9(4) COMP-5 VALUE +160.
         10  SP2-MS-TITLE-LEN    PIC S9(4) COMP-5 VALUE +80.
         10  SP2-MS-LINE-LEN     PIC S9(4) COMP-5 VALUE +80.
     05  SP2-MS-DATA.
******** SP2-MS-NUM-DATA ********
         10  SP2-MS-LINE-CNT     PIC S9(4) COMP-5.
******** SP2-MS-CHAR-DATA *******
         15  SP2-MS-ICON         PIC X.
         15  SP2-MS-BUTTON       PIC X.
         15  SP2-MS-CANCEL       PIC X.
         15  SP2-MS-REPLY        PIC X.
******** SP2-MS-VAR-DATA ********
         10  SP2-MS-TITLE        PIC X(80).
         10  SP2-MS-TEXT         PIC X(80).
```

MS-VAR-LEN is the sum of the lengths of MS-TITLE and MS-TEXT.
MS-TITLE-LEN is the length of MS-TITLE.

Line length

Program name: **MS-LINE-LEN**

The length in characters of each line of text. The text will be divided up into lines of this length. If this is zet to zero or the total length of the message text, the text will be automatically divided up into lines as necessary. If the length is set to -1, a beep will sound rather than a message displayed.

Line count

Program name: **MS-LINE-CNT**

The number of lines into which the message text is to be divided. Normally set to 1 in which case the text will be automatically divided up into lines as necessary .

Icon

Program name: **MS-ICON**

The type of icon to be displayed in the message box:

b = bang (exclamation mark)
s = stop sign
i = information
q = question

Button

Program name: **MS-BUTTON**

The type of Push Buttons to be displayed in the message box:
o = ok
y = yes/no
n = no/yes
r = retry/cancel

Cancel

Program name: **MS-CANCEL**

Specifies whether a cancel button should be displayed:

y = display cancel button

Reply

Program name: **MS-REPLY**

The reply to the message box from the user:

o = ok
y = yes
n = no
r = retry
c = cancel

The replies that can be received are dependent on the buttons displayed.

Title

Program name: **MS-TITLE**

The text to be displayed in the title bar of the message box.

Text

Program name: **MS-TEXT**

The text to be displayed in the message box, split into the number of lines specified by Line count, each line having a length as specified by Line length.

CHAPTER E14 - Extended Repeat Properties

Overview

These properties allow greater control over a repeat group and allow a repeat group to handle a greater number of occurrences than is specified by the repeat group Vertical occurs property.

Extended repeat properties are only accessed programmatically, using data items in the REPEAT-EXT parameter before a SET-REPEAT-EXT call and after (optionally) a GET-REPEAT-EXT call. The REPEAT-EXT parameter is contained in the copy file SP2.CPY.

Parameter layout

```
  01  SP2-REPEAT-EXT.
      05  SP2-RX-RET-CODE          PIC S9(4) COMP-5.
      05  SP2-RX-LENS.
          10  SP2-RX-LEN-LEN       PIC S9(4) COMP-5 VALUE +18.
          10  SP2-RX-NUM-LEN       PIC S9(4) COMP-5 VALUE +20.
          10  SP2-RX-CHAR-LEN      PIC S9(4) COMP-5 VALUE +0.
          10  SP2-RX-PTR-LEN       PIC S9(4) COMP-5 VALUE +12.
          10  SP2-RX-FIELD-LEN     PIC S9(4) COMP-5 VALUE +0.
          10  SP2-RX-COLR-LEN      PIC S9(4) COMP-5 VALUE +0.
          10  SP2-RX-TYPE-LEN      PIC S9(4) COMP-5 VALUE +0.
          10  SP2-RX-LONG-LEN      PIC S9(4) COMP-5 VALUE +16.
          10  FILLER               PIC S9(4) COMP-5 VALUE +0.
      05  SP2-RX-DATA.
******** SP2-RX-NUM-DATA ********
          10  SP2-RX-ID            PIC S9(4) COMP-5.
          10  SP2-RX-NEXT-OCC      PIC S9(4) COMP-5.
          10  SP2-RX-DISP-SW       PIC S9(4) COMP-5.
          10  SP2-RX-NEW-DISP      PIC S9(4) COMP-5.
          10  SP2-RX-BLOCK-SW      PIC S9(4) COMP-5.
          10  SP2-RX-BLOCK-DISP    PIC S9(4) COMP-5.
          10  SP2-RX-BLOCK-OCCS    PIC S9(4) COMP-5.
          10  SP2-RX-TOTAL-SW      PIC S9(4) COMP-5.
          10  SP2-RX-TOTAL-OCCS    PIC S9(4) COMP-5.
          10  FILLER               PIC S9(4) COMP-5.
 ******** SP2-RX-PTR-DATA ********
          10  SP2-RX-FIELD-PTR     POINTER.
          10  SP2-RX-COLR-PTR      POINTER.
          10  SP2-RX-TYPE-PTR      POINTER.
 ******** SP2-RX-FIELD-DATA ******
 ******** SP2-RX-COLR-DATA *******
 ******** SP2-RX-TYPE-DATA *******
 ******** SP2-RX-LONG-DATA *******
          10  SP2-RX-NEXT-OCC-L    PIC S9(8) COMP-5.
          10  SP2-RX-NEW-DISP-L    PIC S9(8) COMP-5.
          10  SP2-RX-BLOCK-DISP-L PIC S9(8) COMP-5.
          10  SP2-RX-TOTAL-OCCS-L PIC S9(8) COMP-5.
```

RX-PTR-LEN is the sum of the lengths of RX-FIELD-PTR,
RX-COLR-PTR and RX-TYPE-PTR.
RX-FIELD-LEN is the length of RX-FIELD-DATA.
RX-COLR-LEN is the length of RX-COLR-DATA.
RX-TYPE-LEN is the length of RX-TYPE-DATA.
RX-LONG-LEN is the length of RX-LONG-DATA.

Id

Program name: **RX-ID**

The id of the repeat group being accessed. It will normally be 1 if there is only one repeat group in the panel.

Next occurrence

Program name: **RX-NEXT-OCC**

The occurrence where the cursor is to be positioned. Overridden by the converse panel Next occurrence property unless the converse panel Next occurrence property is set to zero.

Displacement switch

Program name: **RX-DISP-SW**

Switch used to indicate that the first occurrence displayed in the repeat group is to be changed using the New displacement property (set switch to 1) or the New displacement long property (set switch to –1). This is the easiest way to data displayed in the repeat group. (The other way is to use the converse panel Displacement switch property.)

New displacement

Program name: **RX-NEW-DISP**

The offset in rows of the first occurrence to be displayed in the repeat group. This property is ignored unless Displacement switch is set to 1.

Block switch

Program name: **RX-BLOCK-SW**

Switch to activate extended repeat group processing. This type of processing allows you to pass blocks of data to a repeat group thus avoiding having to pass all the data in one go. The switch should be set to one of the following:

1 or -1= Field pointer (see below) points to the data.

2 or –2 = Field data (see below) contains the data.

3 or –3 = converse panel Fields area contains the data. This is usually the easiest way to handle the data because the area is already defined for you, but you must ensure that the repeat group Vertical occurs property is set high enough to contain the size of the block that you wish to process.

If this property is set to a negative value, the Block displacement long property defines the offset of the block of data rather than the Block displacement property.

Block displacement

Program name: **RX-BLOCK-DISP**

The displacement in rows of the block that is being passed to the repeat group. For example, if the first record in the block being passed was the 75th record in the file, Block displacement would be set to 74. The first block of data that is passed would normally have a displacement of 0 unless you wanted to start off showing data in the middle of a file.

Block occurrences

Program name: **RX-BLOCK-OCCS**

The number of occurrences in the block being passed. This should be at least the number of occurrences that are visible in the repeat group (repeat group Vertical visible property).

Total switch

Program name: **RX-TOTAL-SW**

Switch (set to 1) to indicate that the total number of occurrences being processed is to be set.

Total occurrences

Program name: **RX-TOTAL-OCCS**

The total number of occurrences that are to be processed. This can be an estimate if you are reading a file of unknown length - it should be reset when you reach the end of the file or the number of records exceeds your original estimate. This property is ignored unless Total switch is set to 1.

Field pointer

Program name: **RX-FIELD-PTR**

Pointer to the area that holds the block of data if Block switch is set to 1.

Color pointer

Program name: **RX-COLR-PTR**

Pointer to the area that holds color codes for each field if Block switch is set to 1. May be set to null. If set, the area pointed-to must be big enough to hold color codes for all occurrences, not just the current block.

Type pointer

Program name: **RX-TYPE-PTR**

Pointer to the area that holds type codes for each field if Block switch is set to 1. May be set to null. If set, the area pointed-to must be big enough to hold color codes for all occurrences, not just the current block.

Field data

Program name: **RX-FIELD-DATA**

The area to hold the block of data if Block switch is set to 2.

Color data

Program name: **RX-COLR-DATA**

The area to hold the color codes for the fields in the block of data if Block switch is set to 2.

Type data

Program name: **RX-TYPE-DATA**

The area to hold the type codes for the fields in the block of data if Block switch is set to 2.

Next occurrence long

Program name: **RX-NEXT-OCC-L**

The occurrence where the cursor is to be positioned. Used if you are processing a huge number of records. Ignored unless the Next occurrence property is 0.

New displacement long

Program name: **RX-NEW-DISP-L**

The offset in rows of the first occurrence to be displayed in the repeat group. Used if you are processing a huge number of records. This property is ignored unless Displacement switch is set to -1.

Block displacement long

Program name: **RX-BLOCK-DISP-L**

The displacement in rows of the block that is being passed to the repeat group. For example, if the first record in the block being passed was the 75th record in the file, Block displacement long would be set to 74. The first block of data that is passed would normally have a displacement of 0 unless you wanted to start off showing data in the middle of a file. This property is ignored unless Block switch is set to a negative value.

Total occurrences long

Program name: **RX-TOTAL-OCCS-L**

The total number of occurrences that are to processed. This can be an estimate if you are reading a file of unknown length - it should be reset when you reach the end of the file or the number of records exceeds your original estimate. This property is ignored unless Total switch is set to -1.

CHAPTER E15 - Clipboard Properties

Overview

These properties allow access to the system clipboard.

Clipboard properties are only accessed programmatically, using data items in the CLIPBOARD-DATA parameter before a SET-CLIPBOARD call and after a GET-CLIPBOARD call. The CLIPBOARD-DATA parameter is contained in the copy file SP2.CPY.

Parameter layout

```
01  SP2-CLIPBOARD-DATA.
      05  SP2-CB-RET-CODE         PIC S9(4) COMP-5.
      05  SP2-CB-LENS.
          10  SP2-CB-LEN-LEN      PIC S9(4) COMP-5 VALUE +10.
          10  SP2-CB-NUM-LEN      PIC S9(4) COMP-5 VALUE +2.
          10  SP2-CB-CHAR-LEN     PIC S9(4) COMP-5 VALUE +0.
          10  SP2-CB-VAR-LEN      PIC S9(4) COMP-5 VALUE +80.
          10  SP2-CB-TEXT-LEN     PIC S9(4) COMP-5 VALUE +80.
      05  SP2-CB-DATA.
 ******** SP2-CB-NUM-DATA ********
          10  FILLER              PIC S9(4) COMP-5.
 ******** SP2-CB-VAR-DATA *******
          10  SP2-CB-TEXT         PIC X(80).
```

CB-VAR-LEN is the length of CB-TEXT
CB-TEXT-LEN is the length of CB-TEXT

Text

Program name: **CB-TEXT**

The text to be copied to or from the system clipboard. It may contain embedded end-of-line separators.

CHAPTER E16 - Property Parameter

Overview

The PROPERTY parameter is used in conjunction with the GET-PROPERTY and SET-PROPERTY functions.

Parameter layout

```
01  SP2-PROPERTY.
    05  SP2-PR-RET-CODE         PIC S9(4) COMP-5.
    05  SP2-PR-LENS.
        10  SP2-PR-LEN-LEN      PIC S9(4) COMP-5 VALUE +18.
        10  SP2-PR-NUM-LEN      PIC S9(4) COMP-5 VALUE +6.
        10  SP2-PR-CHAR-LEN     PIC S9(4) COMP-5 VALUE +20.
        10  SP2-PR-VAR-LEN      PIC S9(4) COMP-5 VALUE +2000.
        10  FILLER              PIC S9(4) COMP-5 VALUE +0.
        10  SP2-PR-LENL-LEN     PIC S9(4) COMP-5 VALUE -1.
        10  SP2-PR-NUML-LEN     PIC S9(4) COMP-5 VALUE +0.
        10  SP2-PR-VAR-LEN-L    PIC S9(8) COMP-5 VALUE +0.
    05  SP2-PR-DATA.
******* SP2-PR-NUM-DATA ********
        10  SP2-PR-ID           PIC S9(4) COMP-5.
        10  SP2-PR-ROW          PIC S9(4) COMP-5.
        10  SP2-PR-COL          PIC S9(4) COMP-5.
******* SP2-PR-CHAR-DATA *******
        10  SP2-PR-KEY.
            15  SP2-PR-OBJECT-TYPE
                                PIC X.
                88  SP2-PR-WINDOW   VALUE "W".
                88  SP2-PR-PANEL    VALUE "P".
                88  SP2-PR-STATIC   VALUE "S".
                88  SP2-PR-FIELD    VALUE "F".
                88  SP2-PR-GROUP    VALUE "G".
                88  SP2-PR-REPEAT   VALUE "R".
            15  SP2-PR-TYPE     PIC X.
                88  SP2-PR-LEN-T    VALUE "L".
                88  SP2-PR-NUM-T    VALUE "N".
                88  SP2-PR-CHAR-T   VALUE "C".
                88  SP2-PR-VAR-T    VALUE "V".
            15  SP2-PR-VAR-TYPE PIC X.
                88  SP2-PR-VAR-1    VALUE "A".
                88  SP2-PR-VAR-2    VALUE "B".
                88  SP2-PR-VAR-3    VALUE "C".
                88  SP2-PR-VAR-4    VALUE "D".
                88  SP2-PR-VAR-5    VALUE "E".
                88  SP2-PR-VAR-6    VALUE "F".
                88  SP2-PR-VAR-7    VALUE "G".
                88  SP2-PR-VAR-8    VALUE "H".
                88  SP2-PR-VAR-9    VALUE "I".
                88  SP2-PR-VAR-10   VALUE "J".
            15  SP2-PR-OFFSET   PIC 9(5).
            15  SP2-PR-LEN      PIC 9(5).
            15  SP2-PR-FORMAT   PIC X.
                88  SP2-PR-NUMBER   VALUE "N".
                88  SP2-PR-BINARY   VALUE "B".
                88  SP2-PR-DECIMAL  VALUE "D".
                88  SP2-PR-LONG     VALUE "L".
            15  SP2-PR-ACTION   PIC X.
                88  SP2-PR-REDRAW   VALUE "R".
                88  SP2-PR-RECREATE VALUE "C".
```

```
              15  SP2-PR-VAR-ACT  PIC X.
                  88  SP2-PR-RESET-LEN
                                      VALUE "L".
          10  FILLER              PIC X(4).
   ******* SP2-PR-VAR-DATA ********
          10  SP2-PR-VALUE        PIC X(2000).
          10  SP2-PR-NUM-VALUE REDEFINES SP2-PR-VALUE
                                  PIC 9(5).
          10  SP2-PR-BIN-VALUE REDEFINES SP2-PR-VALUE.
              15  SP2-PR-BIT-VALUE OCCURS 8
                                  PIC X.
          10  SP2-PR-LONG-VALUE REDEFINES SP2-PR-VALUE
                                  PIC 9(8).
   ******* SP2-PR-VAR-DATA-L ******
   ******* 10  SP2-PR-VALUE-L *****
```

PR-ID

The id of the object to be accessed.

PR-ROW

The row of the static object to be accessed.

PR-COL

The row of the static object to be accessed.

PR-KEY

The key of the property to be accessed.

PR-OBJECT-TYPE

The type of the object to be accessed.

W =	window
P =	panel
S =	static
F =	field
G =	group
R =	repeat

PR-TYPE

The type of the property to be accessed.

L =	length
N =	numeric
C =	character
V =	variable

PR-VAR-TYPE

The type of variable property to be accessed.

A thru J depending on the position of the property in the variable portion of the record.

PR-OFFSET

The zero-based offset of the property within the area defined by PR-TYPE and PR-VAR-TYPE.

PR-LEN

The length of the property.

PR-FORMAT

The format of the property value passed or returned.

Blank =	character in PR-VALUE
N =	numeric in PR-NUM-VALUE
D =	one byte value converted to numeric in PR-NUM-VALUE
B =	one byte value converted to 8 on/off (1/0) values in PR-BIN-VALUE
L =	long numeric in PR-LONG-VALUE

PR-ACTION

Special action to be taken during function processing.

R =	redraw
C =	recreate
T =	retrieve from client (only referenced in thin client)
P =	retrieve from program. This currently only applies to the field Protection property which will be retrieved from the program Types area if this item is set.

PR-VAR-ACT

Special action to be taken if accessing a variable property.

L =	reset length of property value to PR-LEN
I =	insert new data into existing variable data at offset PR-OFFSET for a length of PR-LEN
D =	delete existing variable data at offset PR-OFFSET for a length of PR-LEN

PR-VALUE

The value of the property. Not used if PR-VAR-ACT = "D".

PR-VALUE-L

The value of the property if it exceeds 32k.

CHAPTER E17 - Miscellaneous Parameters

Overview

These parameters are used with various functions. The NAME-DEF parameter is used by functions needing to pass just the name of a panel or panel related object. The NULL-PARM parameter is used by functions not needing to pass any data. The BUFFER parameter is used by functions needing to pass a single block of data.

Parameter layouts

```
01  SP2-NAME-DEF.
      05  SP2-ND-RET-CODE         PIC S9(4) COMP-5.
      05  SP2-ND-NAME             PIC X(8).

01  SP2-NULL-PARM.
    05  SP2-NP-RET-CODE           PIC S9(4) COMP-5.

01  SP2-BUFFER.
    05  SP2-BF-RET-CODE           PIC S9(4) COMP-5.
    05  SP2-BF-LEN                PIC S9(4) COMP-5 VALUE +80.
    05  SP2-BF-DATA               PIC X(80).
```

ND-NAME

The name of the panel or object to be accessed.

BF-LEN

The length of the data to be passed.

BF-DATA

The data to be passed.

APPENDIX

APPENDIX A - Keyboard Decimal Values

Decimal Values for Keyboard Keys

The following is a list of keyboard keys and key combinations with their respective ASCII decimal value.

1=Ctrl-A	31=Ctrl-Minus	315=F1
2=Ctrl-B	32=Space Bar	316=F2
3=Ctrl-C	127=Ctrl-Backspace	317=F3
4=Ctrl-D	271=Backtab	318=F4
5=Ctrl-E	272=Alt-Q	319=F5
6=Ctrl-F	273=Alt-W	320=F6
7=Ctrl-G	274=Alt-E	321=F7
8=Backspace	275=Alt-R	322=F8
9=Tab	276=Alt-T	323=F9
10=Ctrl-Enter	277=Alt-Y	324=F10
11=Ctrl-K	278=Alt-U	327=Home
12=Ctrl-L	279=Alt-I	328=Up Arrow
13=Enter	280=Alt-O	329=PgUp
14=Ctrl-N	281=Alt-P	331=Left Arrow
15=Ctrl-O	286=Alt-A	333=Right Arrow
16=Ctrl-P	287=Alt-S	335=End
17=Ctrl-Q	288=Alt-D	336=Down Arrow
18=Ctrl-R	289=Alt-F	337=PgDn
19=Ctrl-S	290=Alt-G	338=Ins
20=Ctrl-T	291=Alt-H	339=Del
21=Ctrl-U	292=Alt-J	340=Shift-F1
22=Ctrl-V	293=Alt-K	341=Shift-F2
23=Ctrl-W	294=Alt-L	342=Shift-F3
24=Ctrl-X	300=Alt-Z	343=Shift-F4
25=Ctrl-Y	301=Alt-X	344=Shift-F5
26=Ctrl-Z	302=Alt-C	345=Shift-F6
27=Esc	303=Alt-V	346=Shift-F7
28=Ctrl-Backslash	304=Alt-B	347=Shift-F8
29=Ctrl-Square Bracket	305=Alt-N	348=Shift-F9
30=Esc	306=Alt-M	349=Shift-F10
350=Ctrl-F1	376=Alt-1	
351=Ctrl-F2	377=Alt-2	
352=Ctrl-F3	378=Alt-3	
353=Ctrl-F4	379=Alt-4	
354=Ctrl-F5	380=Alt-5	
355=Ctrl-F6	381=Alt-6	
356=Ctrl-F7	382=Alt-7	
357=Ctrl-F8	383=Alt-8	
358=Ctrl-F9	384=Alt-9	
359=Ctrl-F10	385=Alt-10	
360=Alt-F1	386=Alt-Minus Sign	
361=Alt-F2	387=Alt-Equal Sign	
362=Alt-F3	388=Ctrl-PgUp	
363=Alt-F4	389=F11	
364=Alt-F5	390=F12	
365=Alt-F6	391=Shift-F11	
366=Alt-F7	392=Shift-F12	
367=Alt-F8	393=Ctrl-F11	
368=Alt-F9	394=Ctrl-F12	
369=Alt-F10	395=Alt-F11	
370=Ctrl-PrtSc	396=Alt-F12	
371=Ctrl-Left Arrow	397=Ctrl-Up Arrow	
372=Ctrl-Right Arrow	401=Ctrl-Down	

373=Ctrl-End	Arrow
374=Ctrl-PgDn	402=Ctrl-Insert
375=Ctrl-Home	403=Ctrl-Delete
	404=Ctrl-Tab

The following values may also be returned as user input:

-1 =	Timeout
-2 =	Selection field selected using mouse or spacebar
-3 =	Switched to another window
-4 =	Control field exited
-5 =	Window closed using system menu (if window has a pushbutton with Help key set to 27, 27 will be returned instead)
-6 =	Menu option selected
-9 =	Window resized
-10 =	Mouse clicked
-11 =	Right mouse button click and converse panel Mouse switch is set to "r" or "o"
-12 =	Mouse double click and panel Miscellaneous property set to X'40'.
-13 =	Right mouse button double click and converse panel Mouse switch is set to "r" or "o"
-14 =	Mouse click outside current window and converse panel Capture switch is "x" and Mouse switch is "y"
-15 =	Toolbar input
-16 =	ActiveX event
-18 =	Click generated as a result of scrolling a repeat group with focus on a control field
-19 =	Window switch was denied because of internal error checking
-22 =	System shutdown (see SP2END in Appendix B)
-23 =	Parent window closed (see SP2END in Appendix B)
-26 =	Window opened
-27 =	Window closed
-28 =	Window entered
-29 =	Window exited
-30=	Notify icon clicked
-31 =	Field entered
-32 =	Splitter field moved
-33 =	Contents of field changed
-34 =	Mouse moved into or out of field

APPENDIX B - Configuring COBOL sp2

The COBOL sp2 runtime checks for a number of configuration variables in order to control various aspects of its behavior. These variables may be assigned to corresponding environment variables or may be placed in a configuration file. It is usually more convenient to place them in a file because this makes it easier to change the variables. The runtime looks for a configuration file by checking for the following (in the order listed):

1. A file called "SP2MYOWN.CFG".
2. The file referenced by the environment variable SP2CFG.
3. A file called "SP2.CFG".

To set up an environment variable, use the command that is appropriate for the system that you are running on. If you are using a configuration file, set up the variables in the following format:

SP2???=?

Please note that the configuration variable must always be:

1. in upper-case
2. one variable per line

The following is a list of the variables used:

SP2ALN	·	set to the id of a font and custom and system entry fields with their Height property set to 0 will be sized the same as the entry field portion of a combo box using this font.
SP2BRK (Unix)	·	set to the numeric value of the key that you wish to use to cancel your program.
SP2BSP (Unix)	·	set this to key code of the key to be used as the backspace key.
SP2CAS)	·	set to "u" or "l" to turn on global case conversin. The field Case property overrides this - set this property to "n" to turn off case conversion for a particular field.
SP2CDB	·	set to 1 to suppress use of common dialog box (useful for testing substitute code).
SP2CEN	·	If configuration variable SP2CEN is set, the century in 4 digit years will be generated automatically during input - only a 2-digit year need be keyed in. For example, if SP2CEN=30, any year between 30 and 99 will be interpreted as 1930-1999 and any year between 00 and 29 as 2000-2029. As soon as the first digit of the 4-digit year is keyed in, the remaining three digits will be blanked out. Additionally this will cause the century to be validated so that it begins with "19", "20" or" 21".
SP2CFG	·	the name of the configuration file If this variable is not defined, the runtime will check for a default configuration file called "SP2.CFG". To check for a default configuration file in the current directory first and then some other file, set this variable to something like ".;\sp2\mysp2.cfg". In order to prevent problems if multiple sp2 systems are being used and each system is relying on SP2CFG, a private configuration file called SP2MYOWN.CFG will be searched for in the current directory before anything else.

SP2CHK	·	set this to the number of times you want to check for a key during WAIT-SW=k processing.
SP2CUS	·	set to 1 to allow limited copy/paste for customfields using the right mouse button.
	·	set to 2 for behavior the ame as above except that no copy/paste wil be allowed if MOUSE-SW = "r" or "o".
SP2DBG	·	set to 0 to turn debug/trace mode off set to 1 to cause a message box to be displayed for critical errors e.g. panel not found set to 2 to cause a trace file to be generated named SP2DBG.nnn where nnn is a number between 000 and 999. set to 3 to cause a message box to be displayed for any error set to 4 to cause a trace file to be generated as for 2 but include details of files accessed set to 5 to cause a trace file to be generated as for 4 but also track memory errors, etc. and generate a dump file called "uib.dbg" if one occurs
SP2DBY	·	set to 1 to prevent "~" character from being treated as a mnemonic indicator if previous character >= 128.
SP2DEC	·	set to 1 for "decimal-point is comma".
	·	Set to 2 to suppress the "jump to decimal point" feature. Normally, initially keying in a decimal point into a custom input field will cause the cursor to jump to the digit after the decimal point. Setting SP2DEC to 2 will cause the initial decimal point to be treated as normal input.
	·	Set to 3 (1 + 2) if you want "decimal point is comma" and "no jump to decimal point".
	·	Set to 8 or 9 (1 + 8) if you want the "jump to decimal point" feature to apply even if the decimal point is not the first character keyed in.
	·	Add 32 to this value to cause the program Fields area to be updated for a field even if the data in this field has not changed.
	·	Add 32 to this value to cause the program Fields area to be updated for a field even if the data in this field has not changed.
	·	Add 64 to this value to allow a decimal point to be entered even if no decimal in the field Format.
	·	Add 128 to prevent the numeric keypad period being interpreted as a comma when SP2DEC=1 which is the default behavior.
SP2DIR	·	list of directories containing panel, font and icon files. Environment variables (expressed as %VAR%) may be included as long as they equate to single directories
SP2DTE	·	Set to 1 to cause the system date format to be used for all dates regardless of the field format. The field format is still used in the editor to derive the length of the field i.e. a length of 8 gives a 2 digit year, 10 gives a 4 digit year.
	·	Set to actual date format eg. MM/DD/YYYY (upper-case) to use that particular format.
SP2DUP	·	set to 1 and, if an open-window is attempted with a name that is already in use, the open will be allowed but a generated name will be used instead.

SP2EBC	·	set to 1 to indicate that EBCDIC signs are being used.
SP2EDT	·	controls handling of system entry fields (Control type = e).).
	·	Set to 1 or 5 and system edit fields will start off in insert mode (as normal) but go into over type mode if the insert key is pressed.
	·	Set to 2 or 6 and system edit fields will start off in overtype mode (as for custom fields) and go into insert mode if the insert key is pressed.
	·	Set to 1, 2, 5 or 6 and date fields and special format fields will always be in over type mode and characters will not be accepted if the cursor is over a separator character (the cursor will jump over separators during normal keying). Also characters will not be accepted past the maximum length of the field as specified by the COBOL format.
	·	Set to 4, 5 or 6 and ctrl-x, ctrl-c and ctrl-v can be used for Cut/Copy/Paste in system entry fields.
	·	Add 8 to this value to prevent system entry fields from appearing as grayed out when Protection is set to "y" (display-only) or "p" (protected).
	·	Add 16 to this value to cause system entry fields to highlight and select their contents when you click in them (as if you used tab to move to the field).
	·	Add 32 to this value to allow system entry fields to support the system highlight facility and Current color.·
	·	Add 64 to this value to cause additional character-by-character input checking (does not effect regular end-of-field checking) as follows: first, only allow a numeric character to be entered in a numeric field; second, truncate last character if field length exceeded in insert mode.
	·	Add 128 to this value to specify that no more characters may be inserted into the field once it is full.
	·	Add 256 to this value to allow double clicks to be captured (-12 will be returned).
		Add 512 to this value to cause the panel color to be used as the default color if field Color not set.
SP2END	·	set to 1 to suppress the display of the system shutdown message box (see SP2MSG) and terminate an application immediately if it's still running at system shutdown. This behavior is the default if sp2thred.dll is used instead of sp2.dll.
	·	set to 2 to suppress the message box and cause a key (-22) to be returned to your program. The next SP2 call will terminate the application. This option only works if sp2thred.dll is used instead of sp2.dll.
	·	set to 4 (0 + 4), 5 (1 + 4) or 6 (2 + 4) to 2 to allow the user to hit Close in a non-active window even if the Switch switch property is not set (normally this action would be ignored). In this case -23 will be returned in the Key property.
SP2ENK	·	set to 1 to allow the End key to be used as a control key in system fields.
SP2EUR	·	if you are using SP2IBM=1, set SP2EUR=xxx where xxx is the numeric representation (128-255) of the character in the IBM character set that you

		wish to use for the euro symbol (ANSI 128).
SP2F10	·	set this to 1 to allow F10 to be used as a control key (normally F10 is used to activate the window menu.)
SP2F11 (Unix)	·	set this to key code that is to be returned when function key 11 is pressed.
SP2FIL	·	set to the number of panel files which may be open (default=4).
SP2FON	·	set to 000000 for default behavior.
	·	reset first 0 to 1 for pre-4.0 font selection rules.
	·	reset second 0 to 1 to ensure that biggest possible font is used.
	·	reset sixth 0 to 1 to recheck font not too big after scaling.
SP2GRI	·	set to 1 to suppress visible graying-out of icons.
SP2HID	·	set to 1 to cause all windows to be created invisible and only made visible after all fields have been created - this may lead to more snappier displays depending on how your system works.
SP2HIN	·	specify milliseconds before icon hint is displayed (default = 500).
SP2IBM	·	set to 1 to indicate that your program is handling data in ASCII (IBM-PC) rather than ANSI format (only required for users using the extended PC character set).
	·	Set to 2 to indicate that your program is handling data using the Roman 8 (HP) character set.
	·	Set to 32 to indicate that your program is handling data using the UTF-8 (multi-byte) character set.
SP2IGI	·	set to 1 and the key related to a grayed-out icon will still function as a valid control key.
SP2KBF	·	set to 1 and keys pressed before the next field is ready to accept input will be held in a buffer so they are not lost. This only works with custom fields.
	·	set to 2 (Unix only) and duplicate character flushing will be disabled).
	·	set to 2 (Non-Unix) to enable keyboard buffering for system fields
SP2LTR	·	same as SP2RTL. See below.
SP2MSG	·	set this to the name of a file containing error message text to cause the default error text to be overridden. The file should contain one line of text for each message. If a line is left blank, the default text will be used. The default messages are: Input must be numeric Invalid value entered Input is out of range Input is required Input must be alphabetic Input exceeds field size Input must be a valid date Input must not have imbedded blanks Please exit active application End session (title for above) If the first line of the file is preceded by a dash "-", this line will be taken as the new title for the error message box.
SP2NCS	·	set this to 1 to stop the cursor moving during

		vertical scrolling.
SP2NLK	·	set this to 1 to override fi-share=y at runtime and prevent file locking actions (i.e. equivalent to setting fi-share=low-value on open-file call).
	·	Set to 2 to allow read-only access to panel file even if another process (e.g. editor) has file open for update. This allows updating of file during 24/7 runtime use with only slight performance penalty.
SP2NUM	·	set this to 1 to cause the operating system to be referenced for the character to be used for the decimal point and thousands separator and the position of the negative sign. This overrides the SP2DEC setting.
SP2OBO	·	set this to 1 to cause borders to be adjusted during scaling.
SP2OHT	·	set this to the height (in pixels) of the screen on which your panels were designed to cause the panels to be scaled up or down for other screen sizes.
SP2OCW	·	set this to adjust horizontal scaling so that a scaled fixed pitch font will still be in sync with the scaled cell size allowing columns to be aligned correctly. For example, if SP2OCW=2 scaling will be done in increments of 1/2 so that SP2OWD=640 would scale to 960 on a 1024 monitor rather than 1024.
SP2OAR	·	set this to 1 to maintain aspect ratio of windows during scaling with SP2OWD and SP2OHT.
SP2OIC	·	set to 1 to cause the bitmap images in icon fields to be scaled as for SP2OWD and SP2OHT.
	·	set to 2 to cause better quality scaling at the expense of longer processing time.
SP2OWD	·	set this to the width (in pixels) of the screen on which your panels were designed to cause the panels to be scaled up or down for other screen sizes.
SP2OWN	·	set to 1 to create all windows as non-owned windows.
	·	set to 2 to cause the first window opened to be moved to the foreground.
		set to 4 to cause all new windows to be moved to the foreground.
		set to 8 to cause a window that is made visible to be moved to the foreground.
SP2PFK	·	set to 1 to treat the top 4 keys on the numeric keypad as F1-F4 like vt keyboards
SP2RDR (Unix)	·	set to a file name to get input from that file rather than /dev/tty
SP2RFR (Unix)	·	set to the numeric value of the key that you wish to use to initiate a screen refresh. This key can be used to restore screens damaged by line noise and also during debugging. See Appendix A for key values.
SP2RGB	·	set to 1, static text will be displayed using a dithered RGB color if necessary
	·	Set to 2 to display all fields using a dithered color if necessary.
SP2RSM	·	set to # of 4k blocks to reserve for dynamic panel expansion.
SP2RTL	·	display menus in right-to-left sequence.
SP2SAV	·	see the release notes for your compiler to check if

	use of this variable is necessary. · Set to 1 if your compiler moves working-storage areas in the event that memory becomes short. · Set to 2 if your compiler passes copies of parameters on call statements rather than the parameters themselves. Set to 4 to suppress the resetting of system entry field values during repaints. This option applies to all compilers.
SP2SGN	· set to 1 and all signed fields will be assumed to have a separate byte for the sign. If you wish to use this option and have existing panels with signed fields, you must run each panel through the generator in order for FD-PROG-LEN to be recalculated (incremented by 1).
SP2TAB	· set to 1 and you can give focus to the tabs by back-tabbing out of the first field on the panel or tabbing out of the last field. · set to 2 and the tabs will be displayed on multiple rows if there is not enough room to display them on one row. · set to 3 for both of the above.
SP2TBF	· set to 1 to prevent the program Fields, Types and Colors areas being accessed while displaying the toolbar.
SP2TCP (Unix)	· set to 1 to use termcap instead of terminfo · set to 3 to use a_termcap · set to 4 to use terminfo and a_termcap
SP2TMO	· set to the number of minutes to wait before timing out on a converse-panel or get-input due to lack of user activity. KEY-TIMEOUT is returned to the program
SP2TMX	set to 1 to cause the application to be terminated on an SP2TMO timeout rather than KEY-TIMEOUT being returned
SP2TTY (Unix)	· set to 0 to indicate a 7-bit terminal (the default). · Set to 1 to indicate an 8-bit terminal that accepts the ISO-Latin multinational character set. · Set to 2 to indicate a PC-compatible terminal · Set to 3 to indicate a terminal using the Roman 8 (HP) character set.
SP2U3D	· set to 1 to use 3D effects on a global scale.
SP2VST	· set to 1 to activate visual styles without the need for a manifest file. Normally a manifest file of the form "manifest.txt" and named "yourpog.exe.manifest" must be present to activate visual styles.
SP2VTS	· set to number of pixels to scroll a window vertically during tabbing or scrolling.

APPENDIX C - Panel File Organization

It is intended that panel files should contain as large a number of panels as you wish. If you wish to have multiple panel files containing logical groups of panels, this is quite permissible. You can have up to three panel files open at any one time. You can also use the OPEN-FILE function to point to a currently open file so that all the open files do not have to be searched to find a particular object. It is NOT recommended that you should have a panel file for each panel.

Panels can be transferred from one panel file to another by using the File/Save menu option in the panel editor. Open the first panel file, then open the panel to be transferred. Then open the second panel file and save the panel to transfer it.

Free space within a panel file that results from updating panel definitions may not always be reused. This may cause your panel file to grow much larger than is necessary. A utility program is included to compress panel files. Invoke this program as follows:

```
sp2check PANEL-FILE-NAME
```

If PANEL-FILE-NAME is not specified, a dialog box will be displayed allowing you to choose the panel fiel to be compressed.

sp2check will create a new file called "checked.pan" which can be copied back to the original file and then deleted.

sp2check can also be used to update a panel file as follows:

```
sp2check SOURCE-PANEL-FILE-NAME TARGET-PANEL-FILE-NAME PANEL-NAME
```

In this case, TARGET-PANEL-FILE will be created if it doesn't exist.

A fourth parameter may also be passed to sp2check. In this case the panel being moved will also be renamed to the name specified as the fourth parameter.

If a panel being transferred from one file to the other has a menu attached to it, the name of that menu will be displayed. If this menu does not exist in the target file, it should be transferred also.

Additional options may be specified on the command line as:

```
sp2check -clm PANEL-FILE-NAME
```

or by responding to the message boxes displayed after the file dialog box.

-c	compress fields
-l	generate a log file called checked.log
-m	discard unused menus

APPENDIX D - Return Codes

The following error codes are returned by the various programming functions:

1	object not found. This may be a file, panel or field.
2	file exists but cannot be opened. May be due to too many files already opened or file is locked.
3	insufficient memory to load object.

In addition, errors may occur because a panel in the process of being updated will no longer fit in memory, or will exceed the 64k limit on an object's size. In these cases, a message box will be displayed indicating the error and your program will abort.

APPENDIX E - The Code Generator

The code generator is normally invoked by selecting the File/Generate menu option. Code will be generated for the panel being edited. As explained in Chapter C1, code is generated based on a template file. The editor prompts for the name of the template file to be used. The last used template file is the default.

The generator can also be invoked directly by running the program SP2GEN. If no parameters are supplied, the program will prompt for panel file, panel and template file. Otherwise, the first parameter is the name of a panel file, the second the name of a panel and the third the name of a template file. If the second parameter is not supplied, code will be generated for all the panels in the panel file. If the second parameter is set to "container", code will be generated for the first container panel in the file. If the third parameter is not supplied, the default template file (SP2.CBX) will be used.

This default template file can be modified to produce code that may be more suited to your own needs. Within the file, there are a number of occurrences of the "$" character followed by 3 numeric digits or an alphanumeric string. These are generator tokens and have special meaning when the generator runs. The digits and strings are interchangeable but the strings make the template code easier to understand. You can use the sp2cbxcv program to convert digit tokens into strings. The meaning of each token is as follows:

$000	$new-file	generate a new file
$001	$panel-name	panel name
$002	$panel-prog-len	length of program Fields area
$003	$panel-prog-cnt	length of program Colors or Types area
$004		used internally
$005		used internally
$006		used internally
$007	$perform for fields with prog-len	for all fields with Program length property not zero, do the following...
$008	$end-perform	end for (like end-perform)
$009	$field-name	field naame derived from Name property
$010	$field-fmt	field format derived from Format property
$011	$if repeat	if field is member of a repeat group
$012	$else	else (if not member of repeat group)
$013	$repeat-vert-occ	repeat group Vertical occurrences
$014	$perform for fields in repeat	for all fields in this repeat group, do the following...
$015	$end-if	end repeat
$016	$end-perform	end of $011 processing
$017	$field-id	value of field Id property
$018	$perform for fields with prog-num	for all fields with Program number property not zero, do the following...
$019	$perform for base fields	for all fields except repeat fields with Occurrence property > 0, do the following...
$020		display number of lines generated
$021	$if first-time	only generate the first time thru this for loop
$022	$file-name	fi-name (panel file name)
$023		if this field, color or type does not immediately follow previous item, do the following... (REDUNDANT)
$024	$field-filler	difference in bytes between start of this item and end of last item
$025		end of $023 processing (REDUNDANT)
$026	$field-	field property, expressed as $026x or $field-y where x or y is the property id (see below)
$027	$if	simple conditional, expressed as:
	$if x = y	$027x=(y) - if x equal y
	$if x != y	$027x!=(y) - if x not equal y
	$if x > y	$027x>(y) - if x greater than y
	$if x < y	$027x<(y) - if x less than y
	$if x>= y	$027x>=(y) - if x greater than or equal y
	$if x <= y	$027x<=(y) - if x less than or equal y
		x and/or y may be constants or dollar variables

$028	$end-if	end of condition
$029	$new-file-ask	generate a new file but issue warning if it already exists
$030	$group-	group property, expressed as $030x or $group-y where x or y is the property id (see below)
$031	$else	else (use with $027 and $028)
$032	$panel-	panel property, expressed as $032x or $panel-y where x or y is the property id
$033	$perform for statics	for all statics, do the following...
$034	$static-	value of static field property, expressed as $034x or $static-y where x or y is the property id (see below)
$035		express dimension in decicells as whole cells with 1 decimal - must be immediately followed by a $026, $03$0, $032, $034 or $054
$036	$dollar	dollar sign
$037		position in cells incremented by 1 (same format as $035)
$038		width in 1/10 cells in terms of pixels (same format as $035)
$039		height in 1/10 cells in terms of pixels (same format as $035)
$040	$var-0 thru $var-9	(thru $049) user variables (may be numeric or non-numeric)
$050	$set	math functions (x must be a user variable):
	$set $var-x = y	$050x=(y) means move y to x (y may be numeric or non-numeric)
	$set $var-x + y	$050x+(y) means compute x = x + y (x, y must be numeric)
	$set $var-x – y	$050x-(y) means compute x = x - y (x, y must be numeric)
	$set $var-x * y	$050x*(y) means compute x = x * y (x, y must be numeric)
	$set $var-x / y	$050x/(y) means compute x = x / y (x, y must be numeric)
	$set $var-x % y	$050x%(y) means divide x by y remainder x (x, y must be numeric)
	$set $var-x # y	$050x#(y) means compute x = y as a numeric (eg #(A) is 65)
	$set $var-x & y	$050x&(y) means compute x = x anded with y
	$set $var-x @ y	$050x@(y) - set x to a character within x at offset y
$051	$perform until exit	loop continuously
$052	$exit perform	break out of loop
$053	$end-perform	end loop
$054	$menu-	menu property, expressed as $054x or $menu-y where x or y is the property id (see below)
$055		text with all ~ replaced by & (same format as $035)
$056	$perform for fields with tab	for all base tab fields, do the following...
$057	$perform for groups	for all groups, do the following...
$058		foreground color of panel, field or static depending on context
$059		background color of panel, field or static depending on context
$060	$toolbar	switch to toolbar panel
$061	$end-toolbar	end toolbar
$070	$var-10 thru $var-19	(thru $079) more user variables (may be numeric or non-numeric)
$080		special color code (same format as $035)
$081		special color code for fg (same format as $035)
$082		special color code for bg (same format as $035)
$083		row in cells expressed in logical cells with 2 decimals
$084		column in cells expressed in logical cells with 2 decimals
$085		width in 1/10 cells expressed in logical cells with 2 decimals
$086		height in 1/10 cells expressed in logical cells with 2 decimals
$087	$decimal	decimal point symbol
$088		width in cells expressed in logical cells with 2 decimals
$089		height in cells expressed in logical cells with 2 decimals
$090		initial value in display format
$091	$perform for fields and statics	for all fields and statics in sequence, do the following…
$092		used by Web Client
$092		used by Web Client
$093		used by Web Client
$094		used by Web Client
$095		used by Web Client
$096		used by Web Client

$097		text with all ~ and leading spaces removed (same format as $035)
$098	$group-name	group name derived from Name property
$099		used by Web Client
$100	$no-sp-lv	generate space and low-values in hexadecimal
$101		font property (same format as $026)
$102		right parenthesis
$103		used by Web Client
$104		suppress end-of-line
$105		used by Web Client
$106	$perform for repeats	for all repeat groups, do the following...
$107	$repeat-	repeat group property, expressed as $107x or repeat-y where x or y is the property id
$108		foreground color of current element in html rgb format
$109		background color or current element in html rgb format
$110		protection type or current field
$111	$perform for panel and subpanels	for panel and all sub panels, do the following…
$112		color property (same format as $026)
$113	$or	or condition (same format as $027)
$114		name of sub panel linked to key code in $070
$115		type of sub panel (child or popup window)
$116		current window property (same format as $026)
$117	$perform for fields	for all fields (including repeated fields), do the following...
$118		value of item in converse-data area (same format as $026)
$119		value of this configuration variable eg. $119DIR
$120	$var-20 thru $var-29	(thru $139) more user variables (may be numeric or non-numeric)
$141		parent of current subpanel
$143	$quote	quote character

A number of the tokens require a property id to be specified. These tokens are as follows:

$026 -	field property (uses field property id)
$030 -	group property (uses group property id)
$032 -	panel property (uses panel property id)
$034 -	static property (uses static property id)
$054 -	menu property (uses menu property id)
$107 -	repeat group property (uses repeat property id)

The property ids for use with numeric tokens are listed in Appendix F. They are structured as follows:

TypeOffset(Length)

where Type is "L" (length), "N" (numeric), "C" (character), "VA" (first variable), "VB" (second variable), etc. For example, $032N4(2) is panel row (numeric value at offset 4 for a length of 2).

The corresponding ids for use with the new character tokens are structured as follows:

Type (Start : Length)

where Type is "len", "num", "char", "var-A", "var-B", etc. For example, $panel-num (5 : 2) is panel row (numeric value starting at position 5 for a length of 2).

APPENDIX F - Property Ids and Keys

Panel Properties

Id(Len)	Key	Name
L8(2)	PL-0000800002N	Length of Description
L10(2)	PL-0001000002N	Length of Title
L12(2)	PL-0001200002N	Length of Cursor keys
L14(2)	PL-0001400002N	Length of Control keys
L16(2)	PL-0001600002N	Length of Message text
L18(2)	PL-0001800002N	Length of User data
L20(2)	PL-0002000002N	Length of Help keyword
N0(2)	PN-0000000002N	Width
N2(2)	PN-0000200002N	Height
N4(2)	PN-0000400002N	Row
N6(2)	PN-0000600002N	Column
N8(2)	PN-0000800002N	Field count
N10(2)	PN-0001000002N	Program field count
N12(2)	PN-0001200002N	Program field length
N14(2)	PN-0001400002N	Help key
N16(2)	PN-0001600002N	Edit override key
N18(2)	PN-0001800002N	Message refresh switch
N20(2)	PN-0002000002N	Title rows
N22(2)	PN-0002200002N	Default pushbutton
N26(2)	PN-0002600002N	Menu rows
N28(2)	PN-0002800002N	Total width
N30(2)	PN-0003000002N	Total height
N32(2)	PN-0003200002N	Message length
N34(2)	PN-0003400002N	Cell width
N36(2)	PN-0003600002N	Cell height
N38(2)	PN-0003800002N	Font id
C0(8)	PC-0000000008	Name
C8(8)	PC-0000800008	Menu name
C17(1)	PC-0001700001D	Current field color
C18(1)	PC-0001800001	Cursor skip
C19(1)	PC-0001900001	Cursor show
C20(1)	PC-0002000001	Cursor movement
C21(1)	PC-0002100001	Shift numerics
C22(1)	PC-0002200001	Blank numerics
C23(1)	PC-0002300001	Assume decimals
C24(1)	PC-0002400001	Format numerics
C25(1)	PC-0002500001	Cursor wrap
C26(1)	PC-0002600001	Initialize numerics
C27(1)	PC-0002700001	Btab overrides reqd.
C28(1)	PC-0002800001	Cell size
C30(1)	PC-0003000001B	Miscellaneous
C31(1)	PC-0003100001D	Cell width division
C32(1)	PC-0003200001D	Cell height division
C38(1)	PC-0003800001D	Color
C39(1)	PC-0003900001B	Program date format
C40(8)	PC-0004000008-	Help panel
C48(1)	PC-0004800001B	Text options
C57(8)	PC-0005700008-	Toolbar
C65(1)	PC-0006500001D	Message line color
C66(1)	PC-0006600001B	More options
C67(1)	PC-0006700001B	Options-3
C74(1)	PC-0007400001	Window border type
C75(1)	PC-0007500001	Initial switch
VA0(*)	PVA00000*****--L	Description

VB0(*)	PVB00000*****--L	Title
VC?(N)	PVC?????00002N	Cursor keys
VD?(N)	PVD?????00002N	Control keys
VF0(*)	PVF00000*****--L	Message text
VG0(*)	PVG00000*****--L	User data
VH0(*)	PVH00000*****--L	Help keyword

* the length of the property value depends on the corresponding length property
? the offset of the property value depends on which item is to be accessed

Field Properties

Id(Len)	Key	Name
L8(2)	FL-0000800002N	Length of Format
L10(2)	FL-0001000002N	Length of Caption
L12(2)	FL-0001200002N	Length of Value
L16(2)	FL-0001600002N	Length of Range
L18(2)	FL-0001800002N	Length of Discrete values
L20(2)	FL-0002000002N	Length of Message text
L22(2)	FL-0002200002N	Length of User data
L24(2)	FL-0002400002N	Length of Help keyword
N0(2)	FN-0000000002N	Id
N2(2)	FN-0000200002N	Gui id
N4(2)	FN-0000400002N	Gui id 2
N6(2)	FN-0000600002N	Occurrence
N8(2)	FN-0000800002N	Base id
N10(2)	FN-0001000002NC	Row
N12(2)	FN-0001200002NC	Column
N14(2)	FN-0001400002N	Program offset
N16(2)	FN-0001600002N	Field number
N18(2)	FN-0001800002N	Tab number
N20(2)	FN-0002000002N	Program number
N22(2)	FN-0002200002NC	Width
N24(2)	FN-0002400002NC	Height
N26(2)	FN-0002600002N	Maximum length
N28(2)	FN-0002800002N	Program length
N30(2)	FN-0003000002N	Item length
N34(2)	FN-0003400002N	Help key
N38(2)	FN-0003800002N	Group id
N40(2)	FN-0004000002N	Repeat id
N42(2)	FN-0004200002NR	Font id
C0(30)	FC-0000000030	Name
C30(1)	FC-0003000001-C	Type
C31(1)	FC-0003100001	Protection
C32(1)	FC-0003200001D	Decimals
C36(1)	FC-0003600001	Initialize if numeric
C37(1)	FC-0003700001B	Miscellaneous
C40(8)	FC-0004000008	Help
C56(1)	FC-0005600001	Required
C57(1)	FC-0005700001	Program control
C58(1)	FC-0005800001	Justify
C59(1)	FC-0005900001	Fill character
C60(1)	FC-0006000001	Assume decimals
C61(1)	FC-0006100001	Special format
C62(1)	FC-0006200001	Case
C63(1)	FC-0006300001	Imbedded blanks
C64(1)	FC-0006400001D	Current color
C65(1)	FC-0006500001	Usage option
C66(1)	FC-0006600001	Cursor show

C67(1)	FC-0006700001	Blank first
C68(1)	FC-0006800001	Blank if zero
C69(1)	FC-0006900001-C	Control type
C70(1)	FC-0007000001DR	Color
C71(1)	FC-0007100001	Mnemonic
C72(1)	FC-0007200001-R	Border type
C73(1)	FC-0007300001B	Program data
VA0(*)	FVA00000*****--L	Format
VB0(*)	FVB00000*****-RL	Caption
VC0(*)	FVC00000*****-RL	Value
VD0(*)	FVD00000*****-L	Class
VE0(*)	FVE00000*****--L	Range
VF0(*)	FVF00000*****--L	Discrete values
VG0(*)	FVG00000*****--L	Message text
VH0(*)	FVH00000*****--L	User data
VI0(*)	FVI00000*****--L	Help keyword

* - the length of the property value depends on the corresponding length property

Static Properties

Id(Len)	Key	Name
L8(2)	SL-0000800002N	Length of Text
N0(2)	SN-0000000002N	Id
N2(2)	SN-0000200002NC	Row
N4(2)	SN-0000400002NC	Column
N6(2)	SN-0000600002NC	Width
N8(2)	SN-0000800002NC	Height
N10(2)	SN-0001000002NR	Font id
C0(1)	SC-0000000001DR	Color
C1(1)	SC-0000100001-C	Type
C2(1)	SC-0000200001	Justify
C3(1)	SC-0000300001B	Miscellaneous
VA0(*)	SVA00000*****-RL	Text

* - the length of the property value depends on the corresponding length property

Group Properties

Id(Len)	Key	Name
L6(2)	RL-0000000002N	Length of variable data
N0(2)	GN-0000000002N	Id
N2(2)	GN-0000200002NC	Row
N4(2)	GN-0000400002NC	Column
N6(2)	GN-0000600002NC	Width
N8(2)	GN-0000800002NC	Height
N10(2)	GN-0001000002N	Field count
C0(30)	GC-0000000030	Name
C30(1)	GC-0003000001	Tab within
C31(1)	GC-0003100001D	Current color
C33(1)	GC-0003300001	Select type
C44(1)	GC-0004400001DR	Color
C46(1)	GC-0004600001-R	Border
VA0(*)	GVA-00000*****-CL	Field ids

* - the length of the property value depends on the corresponding length property

Repeat Group Properties

Id(Len)	Key	Name
L8(2)	RL-0000000002N	Length of Field ids
N0(2)	RN-0000000002N	Id
N2(2)	RN-0000200002NC	Row
N4(2)	RN-0000400002NC	Column
N6(2)	RN-0000600002NC	Width
N8(2)	RN-0000800002NC	Height
N10(2)	RN-0001000002N	Vertical occurs
N12(2)	RN-0001200002NC	Vertical visible
N14(2)	RN-0001400002NC	Vertical gap
N22(2)	RN-0002200002N	Horizontal occurs
N26(2)	RN-0002600002NC	Horizontal gap
N30(2)	RN-0003000030N	Cursor movement
N40(2)	RN-0004000001N	Program field length
N42(2)	RN-0004200001N	Field count
C0(1)	RC-0000000001-R	Border type
C1(1)	RC-0000100001	Tabbing type
C2(1)	RC-0000200001B	Miscellaneous
VA0(*)	RVA-00000*****-CL	Field ids

* - the length of the property value depends on the corresponding length property

Menu Properties

Id(Len)	Key	Name
L8(2)	%	Length of Numeric Option Data
L10(2)	%	Length of Character Option Data
L12(2)	%	Length of Variable Option Data
L16(2)	%	Length of Option Text
N0(2)	%	Option Count
C0(8)	%	Name
C8(1)	%	Draw Switch
VA?(N)	%	Option Id
VA?(N)	%	Option Owner Id
VA?(N)	%	Option Accelerator Key
VA?(1)	%	Option Type
VA?(1)	%	Option State
VA?(*)	%	Option Text

? - the offset of the property value depends on which item is to be accessed
* - the length of the property value depends on the corresponding length property
% - use the get/set-menu-option functions rather than get/set-property

INDEX

INDEX