JAM Author's Guide

Contents

# 1 Introduction

## 1.1 Overview of Authoring

The Introduction to this manual described screens, links, and programs from JAM's point of view; it explained how JAM works and what it does with those things. This chapter is concerned with authoring, or actually building an application, and the point of view is different: for example, we speak about links to screens and links to programs as different things, despite the fact that they both are control links.

To create a JAM application, you build screens, link them together, make a data dictionary, and write programs. Authoring includes all but the last of those tasks, which is covered in depth in the next chapter. The authoring utility is three programs in one, corresponding to the three tasks: its heart is a navigation system that runs your application, enabling you to follow the links from screen to screen, enter data, and see what works and what doesn't. Within this environment you can call up a screen editor and a data dictionary editor; whatever changes you make with them will take effect as soon as you return to the navigation system.

### The Navigation System

The navigation system is an event-driven program similar to the JAM run-time environment: it displays your screens, reads the keyboard, and executes the links found in your application. When you make a menu selection, it finds the screen linked with that selection and brings it up; fills in fields that appear in the local data block, or LDB; and goes back to the keyboard for data entry. When you enter data, it stores it in the LDB. When you press a function key, the navigation system executes that key's link.

Navigating your application is just like running it, with a couple of exceptions. Most important is that you can alter your screens and data dictionary on the fly; special function keys are supplied for invoking the screen and data dictionary editors on the currently displayed screen. You can also have access to your application code in the navigation system, if you link it with the authoring utility. Naturally, if you have application code written in JPL, no linking is necessary.

### Editing Screens

With the screen editor, you can create, change, move, and delete screen fields, as well as the constant display data that is there to explain them. When you change the data entry characteristics of fields, you can immediately try them out. Graphics, screen borders, highlighting, and colors (to the extent supported by your display) are all at your disposal. You can attach help windows and prompts to fields, or to entire screens. Functions attached to fields are also named in the screen editor.

### Editing Control Links

Links are actually parts of the screen, although they are normally hidden from view; you create them with the screen editor, too. To create a menu, for instance, you press a function key to create menu selection and control fields automatically; type in the menu selection text; and finally type the name of the screen or routine for each selection into its control field. To change a menu link, you replace the contents of its control field with the new link.

Screens are normally linked together with menus, as just described; but data entry screens frequently have links to other screens as well. For instance, to link a window for a sub-transaction to its main screen, you would create a control string tied to a function key (using a special screen editor function), and enter the name of the window there. Under navigation, pressing the function key in that screen would cause the window to appear.

The screen editor also interacts with the data dictionary to create data links. There are special function keys that copy a field from the data dictionary to the screen, or vice versa, and look up a field's characteristics in the data dictionary. Very often these capabilities relieve you of the need to get into the data dictionary editor at all.

### Links to Programs

Programming, for the most part, lies outside the authoring process; an author must ensure only that the appropriate links to programs are provided in the screens. But it is important to understand that application routines can be linked in at two sorts of places: to data entry fields, as attached functions, and to control strings, as invoked functions. Attached functions are typically used for validating data entered into a field, or for altering other fields conditionally upon the entry. Invoked functions, on the other hand, are usually attached to function keys (such as TRANSMIT), and process the whole screen.

### Editing the Data Dictionary

The data dictionary editor is most useful when you are in the first stages of creating an application, or for browsing; in a developed application the screen editor's data linking facilities are often more convenient. But if you invent names for your shared data items before creating screens, you can enter them all through the data dictionary editor, and then simply copy those you need into the screens as you create them.

## 1.2 Authoring Examples

### Creating a New Application...

How you go about creating a new application depends on many things; one of the most important is how it is specified. More often than not, one aspect of a new system is much better defined than the others. It makes sense to start from the aspect of the application you know the most about, or have the most stringent requirements on, and let the others evolve through the authoring process. Here are a few scenarios.

### ...from the Top Down

Sometimes the overall organization of an application clarifies itself before the individual transactions do. You can create a menu structure and a first cut at some data entry screens; then you have a running prototype, which you can use as a "straw man" to stimulate concrete thinking and discussion.

### ...from the Bottom Up

Sometimes, on the other hand, the specific functions a system is to perform have been thought out, but the most convenient way to group them together has not. In this case you can go ahead and create data entry screens for all the transactions, then experiment with different ways of linking them: different sets of menus, with corresponding groups of transactions; accessing transactions directly from one another via function keys; or perhaps a mixture of the two.

### Modifying an Existing Application

Bear in mind that navigating with the authoring utility is like running your application; so you simply run through it until you find something that doesn't work, or hasn't been put in yet, and then you get right into the screen editor and fix it. This ability is particularly useful for prototyping: if, while you are demonstrating a prototype, someone asks for some changes, you can put them in and demonstrate them right on the spot.

### Note on function key names

JAM is available on many different computers with a variety of terminals. Because the package is terminal-independent, this manual refers to function keys by generic names (TRANSMIT, EXIT, PF2, etc.). The actual keys you use will depend on your keyboard; see the Introduction and Section 3.4.

## 1.3 On-line Help

The JAM authoring utility features on-line help. Every pop-up window and menu has a help screen, and where appropriate each field or selection within the window will have its own. Windows with more than one screenful of information may be scrollable, or they may contain a menu of subtopics in more windows.

Pressing the HELP key will get you the most specific help available: the help window for the field under the cursor, if it has one, or for the whole pop-up. Pressing the FORM HELP key will always bring up the pop-up's help window.

Finally, in the screen editor's draw mode, pressing either HELP or FORM HELP will bring up a long introduction to the screen editor. Pressing either key with the cursor in the first column of the data dictionary editor screen will get you a summary of its functions.

## 1.4 Entering and Exiting the JAM Authoring Utility

You invoke the JAM authoring utility by entering

```
jxform top-level-screen
```

The optional argument top-level-screen is the name of a screen. Depending on your environment, JAM may automatically append an extension such as .jam to it. The screen may or may not exist; if it does, it will be used as the top-level screen by the JAM run-time system.

The display will clear. jxform will then check for the presence of the data dictionary and LDB initialization files, and issue some diagnostic messages; hit the space bar to dismiss the messages. You will now be presented with the screen named in the command line, if any, with the following legend at the bottom of the display:

```
SPF1: TOP  SPF2: SHELL  SPF3: GOTO  SPF5: FMKR SPF6: DD
```

The navigation functions are now available. To enter the screen editor, press SPF5; to enter the data dictionary editor, press SPF6.

To exit the authoring utility, press EXIT. You will be prompted for confirmation before it exits to the operating system.

# 2 Navigation

## 2.1 Testing Control Links

Control links define the flow of a JAM application. The Introduction explains them, and Section 7 tells how to create them. Here is a brief list for reference:

| Lead Character | | Link Action |
|---|---|---|
| caret | ^ | call a function exclamation point |
| | ! | run a program ampersand |
| | & | display a window other |
| | display a form | |

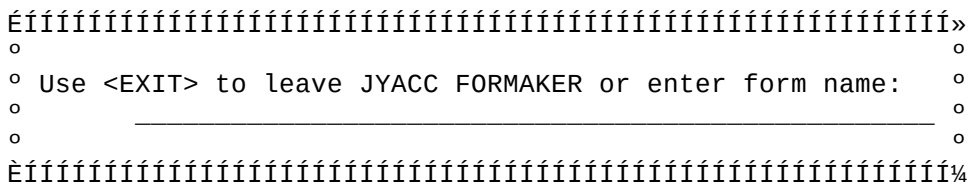


Figure 1: JAM Entry Screen

When navigating in the authoring utility, all link types except the first are active. (In order for function calls to be effective, your application code must be compiled and linked with the authoring system.) Thus, when you make a menu selection or press a function key, JAM will bring up the screen or run the program specified in the associated control link.

## 2.2 Special Navigation Control Keys

Three function keys have special meanings within the navigation mode of the authoring utility, as well as while your application is running.

| | |
|---|---|
| SPF1 (Top Level) | Returns you to the screen named on the command line when jxform was invoked. This has the side effect of removing all screens subsequently entered from the control path. |
| SPF2 (Shell Escape) | Prompts you for an operating system command, then executes it. Before execution the display is cleared and set to the normal operating system mode; after execution it is reset to the JAM mode, you are prompted to hit the space bar, and the screen is redrawn. |
| SPF3 (Go To Screen) | Prompts you for the name of a screen, then brings it up. This is a useful way to short-circuit menu trees, if you know the name of the screen you want. |

# 3 Entering and Exiting the Screen Editor

## 3.1 Entering the Screen Editor

You enter the screen editor from the navigation screen by pressing SPF5. If you invoked jxform with the name of a screen to edit, it will respond:

```
JYACC FORMAKER Rel 4.0 Copyright (C) JYACC, Inc. 1988
Editing the form "formname".
Hit the <EXIT> key to abort editing this file,
or any other key to continue.
```

If you hit a key other than the EXIT key, JAM brings up the screen for editing, in test mode. If no screen was displayed for navigation, or you hit the EXIT key at the first prompt, the window shown in Figure 1 will appear. Enter the name of the screen you want to edit and hit TRANSMIT.

## 3.2 Exiting the Screen Editor

You invoke the screen editor's exit functions by hitting the EXIT key when no pop-up windows are displayed. The exit options menu (Figure 2) will appear, with save shown in reverse video. To make a selection, either:

1. Use the tab, backtab, space bar, backspace, or arrow keys, to position the reverse video area to the desired option, then hit TRANSMIT, or
2. Hit the initial letter of the desired option (e.g. c for continue).

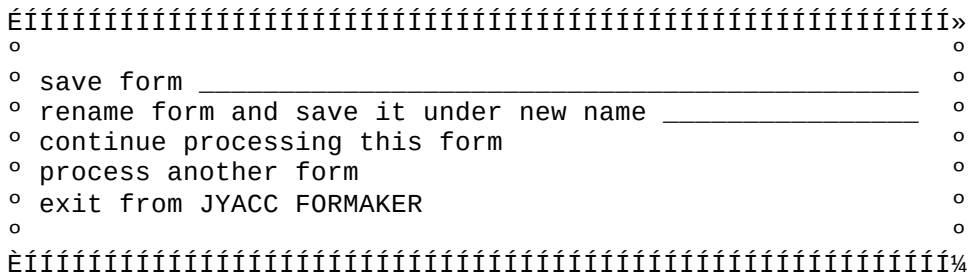


Figure 2: Exit Options Menu

Note that choosing exit or hitting the EXIT key will exit the screen editor without saving the screen. To rename the screen:

1. Choose rename. The cursor will be positioned to a field on the right of the option line, where the name is to be entered.
2. Enter the new name.
3. Hit TRANSMIT to save the screen under its new name. To abort or choose another option, hit the EXIT key.

To process another screen, choose process. The screen name/exit window will appear.

## 3.3 Screen Editor Processing Levels and Modes

The JAM screen editor operates on its top level at all times except

- while one or more pop-up windows are displayed;
- while you are moving or copying a field or display area;
- during exit functions.

When the screen editor is on its top level, all its function keys are available. At other times, none are available (except PF7 and PF8, which are used for terminating the move and copy functions, respectively). On the top level, the TRANSMIT key compiles the screen, and the EXIT key invokes exit functions. When the pop-up windows are active, the TRANSMIT key is used to effect changes, and the EXIT key is used to abort the changes, or to terminate processing on that level.

On the top level, the screen editor has two modes of operation, draw mode and test mode. You create and modify screens primarily in draw mode; in test mode, you can quickly check whether they behave as you intend. The display's status line always indicates which mode you are in, so you can check at a glance.

- In draw mode, you can enter data anywhere on the screen, using arrows and the RETURN key to position the cursor. The display shows an exact image of the screen as it is developed. Constant data (such as borders, field labels, and initial field contents) have their true colors and display attributes, as supported by the display hardware. Fields you have defines for data entry appear in draw mode as underlined areas.
- Test mode enables you to try out the screen, to see whether it works as planned and is comfortable to use. In test mode, data may be entered only in previously defined fields, and all data restrictions and field editing rules apply.

Certain function keys are used directly for toggling the mode, and for deleting, moving, and copying areas of the screen. Other function keys bring up menus and windows for entering detailed information about screen size, display attributes, data restrictions, and editing rules. The next section summarizes those keys.

## 3.4 Data Entry

In the JAM run-time system, data entry is permitted only in fields. When you type a normal data character, it is copied into the field under the cursor, subject to certain restrictions and rules. There are also specially defined keys that move the cursor, clear areas, scroll, invoke help, etc. In this section we explain rules that apply generally to all fields. Data entry restrictions that can be applied to specific fields are explained later on, in Section 6.

### Menus

Data entry rules for menu screens are completely different and much simpler than for data-entry screens, to which the rest of this section applies. In a menu there is a reverse-video cursor, often referred to as a "bounce bar," that occupies all of the current menu item. The TAB, right and down arrows, space, and RETURN keys all move the bounce bar to the next eligible menu item; the BACKTAB, left and up arrows, and BACKSPACE key all move it to the previous item. Pressing the TRANSMIT key causes the item under the bounce bar to be selected and returned to the function processing the menu. Typing the first character (or characters) of a menu item may also cause it to be selected. All these behaviors are subject to modification by library functions.

### Insert and Overstrike Modes

At any given time, either insert or overstrike mode is active. In overstrike mode, characters you type simply replace any previously existing data in a field. In insert mode, the old field contents are shifted, left or right according to the field's justification, to make room for the new character. The INSERT key toggles this mode.

### Character Edits

Fields can be marked to permit the entry of only certain characters; for instance, a field that is to contain a number will accept only digits, and beep if you attempt to enter anything else. See Section 6.5 for full details.

### Special Data Editing Keys

The following table describes the behavior, in test mode and applications, of the data editing keys defined by JAM. Where their behavior in draw mode is different, the differences are noted. Where there are run-time options that modify the behavior, they are also noted.

| Short name | Long name | Description |
|---|---|---|
| ABORT | ABORT | Causes keyboard input functions to return to their callers as quickly as possible, and sets a global flag. |
| BACK | BACKTAB | Move the cursor to the unprotected field with the next lower number. Validation is inhibited by default. BKSP |
| | BACKSPACE | Delete the character to the left of the cursor, and shift all characters to the right (left) of the cursor left (right) 1 position, according as the field is left (right) justified. |
| CLR | CLEAR ALL | Clear all unprotected fields in the current window, both onscreen and off. System date and time fields are updated. |
| DARR | DOWN | Move to the entry point of the next field below the current line, unless inhibited by sm_ok_options. If in |

| | | |
|---|---|---|
| | | the last line of a scrolling array, scroll up this and any parallel arrays.<br>In draw mode, move the cursor down one line. |
| DELE | DELETE CHAR | Delete the character under the cursor and shift the field contents on its right (or left) one position to the left (or right), according as the field is left (or right) justified.<br>In draw mode, everything is treated as left-justified. |
| DELL | DELETE LINE | Move all array occurrences below the current one in this and any parallel arrays up one line, overwriting the one under the cursor.<br>In draw mode, delete the current line and scroll up those below. |
| EMOH | LAST FIELD | (HOME backwards.) Move the cursor to the beginning of the last unprotected field. |
| EXIT | EXIT | Cease the current operation and usually close the current window without making the changes that have been specified since the window opened. |
| FERA | ERASE | In a left-justified field, clears from the cursor to the end of the field; in a right-justified field, always erases the whole field. In a system date or time field, updates the value. |
| FHLP | FORM HELP | Display the help window for the current screen, regardless of the field in which the cursor is placed. |
| HELP | HELP | Display the help screen for the field in which the cursor stands, or for the screen if the field has none. |
| HOME | HOME | Move the cursor to the entry point of the first unprotected field. If no field is unprotected, move to the top left-hand corner of the screen. |
| INS | INSERT CHAR | Toggle the insert/overstrike mode of data entry. |
| INSL | INSERT LINE | In an array and any parallel arrays, move the current item and all below it down by one, and clear the current item. Fails if the last item in the array or one parallel is already filled.<br>In draw mode, move the current line and all below it down by one, and blank the current line. |
| LARR | LEFT | Move the cursor one position to the left within a field; from the first position, move into the previous field, unless inhibited by ok_options.<br>In draw mode, move the cursor 1 character to the left. |
| LP | LOCAL PRINT | Send what is in the screen buffer to a file for printing. |
| NL | RETURN | Move the cursor to the first unprotected field below the current line, wrapping to the top if there are none below. If in the last line of a scrolling array, scroll up this and any parallel arrays.<br>In draw mode, move the cursor to the beginning of the next line, wrapping at the bottom. |
| RARR | RIGHT | Move the cursor one position to the right. If at the right end of a shifting field, shift in offscreen data. At the very end of the field, move to the beginning of the next unprotected field. May be inhibited, equated to TAB, or otherwise altered by ok_options.<br>In draw mode, move the cursor right by one position. |
| REFR | RESCREEN | Refresh the screen if it gets scrambled by line noise or output from other programs. |
| SPGD | PAGE DOWN | In a scrolling array, scroll it and any parallel arrays down by several lines. |
| SPGU | PAGE UP | In a scrolling array, scroll it and any parallel arrays up by several lines. |
| TAB | TAB | Move the cursor to the beginning of the next unprotected field. If in the last such field, move to the first such field. Field entry and exit routines are performed as |

| | | |
|---|---|---|
| | | appropriate; if field validation fails, the cursor remains in the current field. |
| UARR | UP | Move the cursor to the entry point of the previous field above the current line, unless inhibited by ok_options. If in the first line of a scrolling array, then scroll down this and parallel arrays, if any, if any previous items exist.<br>In draw mode, move the cursor up one line. |
| XMIT | TRANSMIT | Make effective any choices made in the current window, frequently closing the window as a side effect. Causes validation of the entire screen.<br>In draw mode, convert all underscored strings to fields. |
| ZOOM | ZOOM | Expand the scrolling and/or shifting field under the cursor into a pop-up window, where more of the field will be visible. You may enter data into this window just as into the field itself. |

Special Function Keys

The table below describes the actions of program function keys in the navigation system, screen editor, and data dictionary editor, which are respectively denoted by N, S, and D in the second column.

| Key | Context | Description |
|---|---|---|
| PF2 | S | Toggles the utility between draw mode and test mode. |
| | D | Adds an item at the cursor position. |
| PF3 | S | Brings up the form characteristics window. |
| | D | Modifies the item under the cursor. |
| PF4 | S | Brings up the field characteristics window, if the cursor is positioned within a field, or the display attributes window if the cursor is positioned within a display area. |
| | D | Same as in the screen editor, unless the entry has scope r, in which case the record editing window appears. |
| PF5 | S | Brings up the field summary window. |
| | D | Deletes the item under the cursor. |
| PF6 | S | Deletes the display area, field, or array of fields within which the cursor is positioned. |
| | D | Undeletes the last item deleted. |
| PF7 | S | Moves the display area, field, or array under the cursor. |
| | D | Searches for an item by name or comment field, with wildcard matching. |
| PF8 | S | Copies the display area, field, or array under the cursor. |
| | D | Finds the next item that matches the previously entered search string. |
| PF9 | S | Repeats the last move, copy, delete, graphics, or change field characteristics sequence. See below for details. |
| | D | Moves the cursor to a particular item in the dictionary, by number. |
| PF10 | S | Brings up a menu with options corresponding to the shifted function keys. |
| | D | Brings up the data dictionary field defaults window. |
| SPF1 | S | Brings up the JAM control strings window. |
| | N | Brings up the application's top-level screen. |
| SPF2 | S | Brings up the JAM Menu window. |
| | N | Prompts for an operating system command and executes it. |
| SPF3 | S | Causes field names to be displayed temporarily, in place of their contents. |
| | N | Prompts for the name of a JAM screen, and displays it as a base form. |
| SPF4 | S | Creates a data dictionary entry from a named field. |

| | | |
|---|---|---|
| SPF5 | S | Brings up the graphics character selection window. |
| | N | Invokes the screen editor on the currently displayed screen. |
| SPF6 | S | Starts a data dictionary search. |
| | N | Invokes the data dictionary editor. |
| SPF7 | S | Creates a jam_name field at the current cursor position. |

The PF9 key in the screen editor repeats, internally, the last sequence of keystrokes:

1. starting with PF4 and ending with the closing of the field characteristics (or display attributes) window, or
2. consisting of PF6 (only), or
3. beginning and ending with PF7 (move sequence), or
4. beginning and ending with PF8 (copy sequence), or
5. starting with SPF5, the graphics key, and ending with TRANSMIT.

The screen display is changed only to reflect the end result of the sequence. You must be careful to position the cursor before hitting PF9 so that the repeated sequence makes sense. In particular, a sequence that changes field characteristics should be repeated only when the cursor is within a field; a sequence that changes the attributes of a display area should be repeated only in another display area.

## 4 The Screen as a Whole

Screens created by JAM are, by default, as big as your display. Smaller screens are convenient for use as windows, which temporarily overlay part of a larger screen. Screens can be outlined by a reverse-video or graphics border. On color displays, you can select a border color; on displays that support background color, you can select a background color as well.

For quicker and more consistent creation of screens, you can define several draw-field symbols, and an initial display attribute for display data. These defaults, as well as display text and any fields common among screens in your application, can be stored in a template and used as a basis for the creation of many screens.

Finally, you may define a help window to provide basic information to a user of your screen, and screen-entry and screen-exit functions to perform initialization and wrap-up on the screen. jxform provides access to all these screen characteristics through the screen characteristics window (Figure 3), which you can call up using PF3.

### 4.1 Screen Size

To change the screen size:

1. Hit the PF3 key to bring up the screen characteristics window (Figure 3). The window will show the current size.
2. Enter the new number of lines (or TAB, if no change).
3. Enter the new number of columns (if changed).
4. Hit TRANSMIT to effect the change (or EXIT to abort).

jxform will not allow a screen to be made larger than the display, nor smaller than its contents. To reduce the size of a screen, it may therefore be necessary to delete or rearrange some of it first.

### 4.2 Borders

To add or change a border:

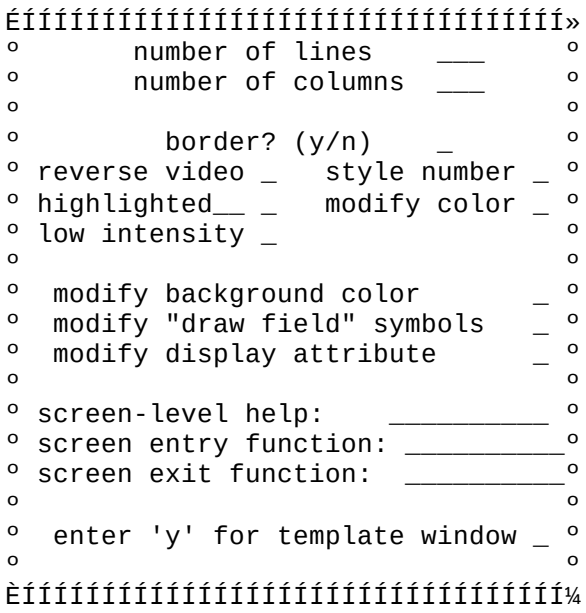


Figure 3: Screen Characteristics

1. Hit PF3 to bring up the screen characteristics window (Figure 3). It will show the current border characteristics; if there is no border, several of the fields shown in the figure will not be visible.
2. After border? enter y if a border is desired, or n for no border.
3. If a border is chosen, the window will display some additional yes/no prompts:

    a. reverse video (defaults to y).
    b. highlighted (defaults to y). On displays that support low intensity but not highlighting, the prompt will be low intensity instead.
    c. style number (defaults to 0). Enter a digit from 0 through 9. The chosen border will immediately appear around the screen characteristics window, and the cursor will remain in this field so you can try more selections. (The actual styles available depend on your video file.)
    d. modify color. See the next section.

4. Hit TRANSMIT to effect the change (or EXIT to abort).

### 4.2.1 Border Colors

On color displays, the screen characteristics window has an option for border color selection (Figure 3). To change the color of a border:

1. Hit PF3 to bring up the screen characteristics window.
2. Enter y after border.
3. Tab to modify color and enter y. The color selection menu (Figure 4) will appear, with the current color shown in reverse video. (The default color is white.) To change the color, either:

    a. Using TAB, BACKTAB, space, BACKSPACE, or the arrow keys, position the reverse video area to the desired color, and hit TRANSMIT, or
    b. Hit the initial letter of the desired color (r for red, etc.). Choosing exit or hitting the EXIT key will close the window without changing the border color.


Figure 4: Color Selection Menu

At this point, the screen characteristics window will still be displayed, and you can make any other desired changes. When finished, hit TRANSMIT to save the new screen characteristics, or EXIT to abort. The screen's border, if any, will now have the color that was selected on the color menu.

4.3 Background Color

If background color is available, the screen characteristics window will display an option for background color selection (Figure 3). To change the background color:

1. Hit PF3 to bring up the screen characteristics window.
2. After modify background color enter y. The color selection menu (Figure 4) will appear, with the current color shown in reverse video. The default is no color (black). To change the color, either:

    a. Using TAB, BACKTAB, space, BACKSPACE, or the arrow keys, position the reverse video area to the desired color, and hit TRANSMIT, or
    b. Hit the initial letter of the desired color (r for red, etc.). Either choosing exit or hitting the EXIT key will close the window without changing the background color.

At this point, the screen characteristics window will still be displayed, and you can make further changes if you like. When you are done, hit TRANSMIT to save the new screen characteristics, or EXIT to abort. The screen's background will now be the color that was selected on the color menu.

4.4 Draw-field Symbols and Field Defaults

The underscore character is used by default for drawing fields. The resulting fields, when first compiled, are underlined and highlighted (if highlighting is available), and have no character or field edits. You can change those field characteristics, and define more draw-field characters for creating fields with different initial characteristics. To specify draw field symbols or initial field characteristics:

1. Hit PF3 to bring up the screen characteristics window (Figure 3).
2. Enter y after modify "draw field" symbols. The draw-field symbol window (Figure 5) will be displayed. Initially the window will show only one draw-field symbol, the underscore; you can define up to 9 symbols.
3. Change or add draw-field symbols by entering them in the window.

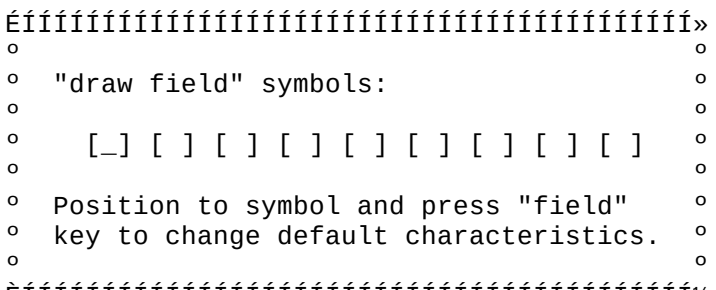


Figure 5: Draw-field Symbol Window

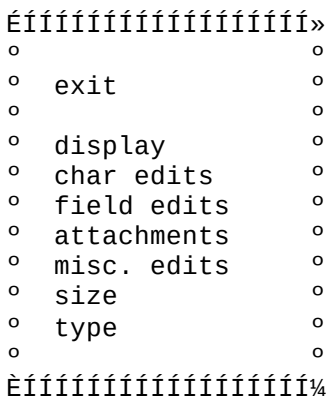


Figure 6: Default Field Characteristics Menu

To change the initial field characteristics, position the cursor to the corresponding draw-field symbol, and hit the PF4 key to bring up the default field characteristics menu (Figure 6). This menu is just like the regular field characteristics menu, Figure 9; refer to Section 6 for its use, as well as explanations of the many different characteristics you can give your fields.

Note that different symbols can be used to create fields with different initial characteristics. The initial characteristics as specified above will apply, however, only to fields that are created afterwards; previously created fields will not change. If you want to use the same draw-field symbols for several different screens, you may find it convenient to define the symbols, as above, for one screen, and then use that screen as a template for others; see Section 4.8.

## 4.5 Default Display Attributes

It is also possible to change the attributes with which display data is first written to the screen, thus:

1. Hit PF3 to bring up the screen characteristics window (Figure 3).
2. Enter y after modify display attribute. The display attributes window (Figure 8) will appear, with the current attributes marked with a y.
3. To turn on an attribute enter y; to turn it off, enter n or space. More than one attribute may be selected.
4. If you have a color display, the initial color can be changed by entering y after modify color, and proceeding as in Section 5.2.1.
5. Hit TRANSMIT to effect the change, or EXIT to abort. Note that a color selection, if any, will take effect even if the other attribute changes are aborted.

Display data created after this point will have the display attributes specified above; display data already on the screen will retain their old attributes.

## 4.6 Help Windows on a Screen Basis

Note: the remainder of Section 4 uses some features of JAM that have not yet been presented. If you are reading this manual for the first time, you may want to skip to the beginning of Section 5.

A help window may be specified on a screen basis. The screen so designated will be displayed as a window whenever the FORM HELP key is pressed. It will also be displayed when the HELP key is struck and the cursor is not in a field that has its own help or item selection window.

To specify a help window for a screen:

1. Hit PF3 to bring up the screen characteristics window (Figure 3). If a help window has already been specified for this screen, its name will be displayed.
2. After screen-level help enter the name of the screen that is to serve as a help window for the current screen. You may optionally follow the window name with the line and column at which it is to appear, as helpwin(5,25).
3. Hit TRANSMIT to effect the change, or EXIT to abort.

The creation of help windows is described fully in Section 10.6.

## 4.7 Screen-entry and Exit Functions

These functions are analogous to the field-entry and exit functions described later. They are called by library functions such as r_window, d_form, or close_window at the very beginning and end of a screen's active life. They may be used to initialize the screen, to allocate or release resources, or to do whatever the application needs at that point. Refer to the Programmer's Guide for details. To specify such a function, follow the instructions below. For simplicity, they refer only to screen-entry functions, but the procedure for screen-exit functions is exactly analogous.

1. Press PF3 to bring up the screen characteristics window (Figure 3).
2. Tab to the field labeled screen entry function and type in the function name.
3. Hit TRANSMIT to accept the screen-entry function, or EXIT to discard the change.

Functions written in a standard programming language must be registered with the JAM library through a call to install; see the Programmer's Guide. You can also use a procedure written in JPL, the JYACC Procedural Language, and stored in a file by entering jpl filename in the screen-entry or exit function field.

## 4.8 Using Another Screen as a Template

When designing a screen, a copy of an existing screen can be used as a template. Screen templates can be particularly useful for creating several screens with similar characteristics, including initial display attributes (Section 4.5) and draw-field symbols (Section 4.4). To make use of this feature:

1. Enter the screen editor with the name of the new screen (Section 3).
2. After the screen is blanked, hit PF3 to bring up the screen characteristics window (Figure 3).
3. Tab to the last field and enter y. The window shown in figure 7 will pop up.
4. Enter the name of the existing screen and hit TRANSMIT, or EXIT to abort. The screen is now initialized to a copy of the existing screen.

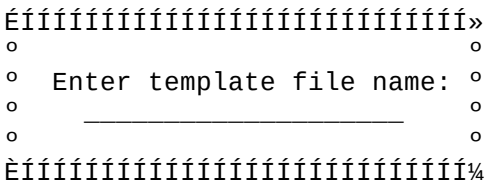


Figure 7: Template Window

5. You will be prompted to hit TRANSMIT to accept the template, or EXIT to reject it.

Loading a template destroys whatever fields or data you had previously created for the new screen.

## 5 Display Data

Display data include form headings, field headings, and all constant data that are displayed as part of the screen. Borders and messages to be displayed on the status line are not considered display data. To enter display data:

1. Make sure you are in draw mode. Hit the PF2 key if necessary.
2. Use the RETURN and arrow keys to position the cursor.
3. Enter the display data as you want them to appear on the screen. All displayable characters can be used, except those reserved for drawing fields. If you have not defined any draw-field symbols yourself (Section 4.4), the only reserved character is the underscore.

The remainder of this section will sometimes treat an area of display data as a unit. Such a display area consists of display data that are on one line; not interrupted by the start or end of a field; and not separated by more than one blank.

Entry of display data follows the rules defined in Section 3.4. When a key pays attention to the extent of a field, a display area is treated like a field.

### 5.1 The Graphics Selection Window

If you need to enter special graphics characters, hit the SPF5 key to bring up the graphics selection window. JAM will display all the graphics characters defined for your terminal in a window. Use arrow keys to position the cursor over the character you want, hit TRANSMIT, and it will be placed on your screen at the current cursor position. To insert a character repeatedly, use the PF9 (repeat) key instead of bringing up the window again.

### 5.2 Display Attributes

You can change the appearance of a display area by changing its display attributes, thus:

1. Position the cursor anywhere within the display area.
2. Hit the PF4 key to bring up the display attributes window (Figure 8). It will show which attributes are currently in effect. The list of available attributes will depend on the video file associated with the terminal, but should always include non-display and underlined. (On displays that cannot underline, JAM will simulate underlining by placing an underscore character wherever the actual character is a blank.)
3. To set an attribute, enter y; to reset one, enter n or space. Attributes may be combined.
4. Hit TRANSMIT to effect the change, or EXIT to abort.

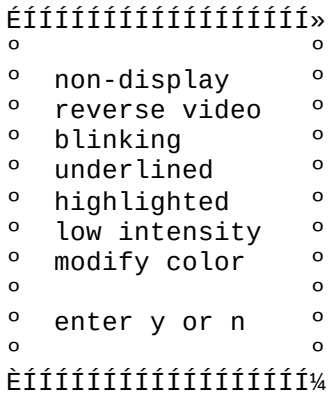


Figure 8: Display Attributes

To change the display attributes of a field, see Section 6.4.

The above procedure will change the attributes of a display area after it has been created. To change the default attributes of display data, see Section 4.5.

### 5.2.1 Color

On screens that support color, the display attributes window has an additional option for color selection (Figure 8). To change the color of a display area:

1. Position the cursor anywhere within the display area.
2. Hit the PF4 key to bring up the display attributes window.
3. Enter y after modify color. The color selection menu (Figure 4) will be brought up on the screen, with the current color shown in reverse video.
4. To change the color, either:

    a. Using the TAB, BACKTAB, space bar, BACKSPACE, or arrow keys, position the reverse video area to the desired color, and hit TRANSMIT, or
    b. Hit the initial letter of the desired color (r for red, etc.). Either choosing exit or hitting the EXIT key will close the window without changing the color.

At this point, the display attributes window will still be displayed; you may continue to change attributes as desired. When finished, hit TRANSMIT to set the display attributes (or EXIT to abort). The color will remain as selected on the color menu.

## 5.3 Deleting, Moving, and Copying Display Data

Individual display characters can be deleted, in draw mode, with the DELETE CHAR key (Section 3.4). Similarly, an entire display area or the end of a display area can be deleted by using the ERASE key in draw mode. An entire display area can also be erased by positioning the cursor anywhere within it and hitting the PF6 key.

You can reposition or copy a display area anywhere within a screen. If the new location overlays a field, the display area becomes initial data for that field. It is possible to overlay (and lose) other display data. The procedures are identical, except that you use PF7 to move and PF8 to copy:

1. Position the cursor anywhere within the chosen display area.
2. Hit the PF7 (PF8) key. The extent of the data to be moved will be shown by brackets. If possible, all fields will appear underlined, and non-display fields will be made visible.

3. Position the brackets to the display data's new location on the screen, using the arrow, TAB, BACKTAB, PAGE UP and PAGE DOWN keys. (The arrow keys move the data one line or column at a time. The TAB and BACKTAB keys move the data ten columns at a time, wrapping past the sides of the screen. The PAGE UP and PAGE DOWN keys move the data five lines at a time, wrapping past the top and bottom of the screen.)
4. Hit the PF7 (PF8) key again to deposit the display area in its new location, or hit TRANSMIT. Hit the EXIT key to abort the procedure.

# 6 Fields

Fields are areas you define for data entry when the screen is used. Each field must be contained within a single line of the screen. Using pop-up menus, you can set display attributes separately for each field, restrict what may be entered into fields, and specify many other field characteristics.

## 6.1 Creating Fields and Compiling the Screen

To create a field:

1. Make sure you are in draw mode. Hit the PF2 key if necessary.
2. Use the RETURN and arrow keys to position the cursor.
3. Type underscores (or any key defined as a draw-field symbol, Section 4.4) to define the extent of the field.

At this point, the field is only display data, and all rules for modifying, deleting, moving, and copying display data apply. To change areas defined by the underscore (or other symbol) into actual fields, hit the TRANSMIT key to compile the screen. When a screen is compiled, any areas containing underscores or other draw-field symbols and not within existing fields or borders are converted into fields. Fields are then renumbered, from top to bottom and from left to right within each line. Compilation is fast, and may not be noticeable. A screen is automatically compiled:

1. When the PF2 key is used to toggle from draw mode to test mode.
2. Whenever the TRANSMIT key is struck while the screen editor is at its top level (Section 3.3).
3. Before jxform's exit functions are performed (when the EXIT key is struck at the top level).
4. After a field is deleted, moved, or copied.
5. After a field is made shiftable or scrollable, or reset to non-shiftable or non-scrollable.
6. After a field is made an array, the number of elements in a array is changed, or an array is converted back to a single field.
7. After a field attachment or other special edit is altered.
8. When a help window for the screen as a whole is added, deleted, or changed.
9. When a field data type is changed to or from its default value.

Only the first three actions in the list will cause new fields to be created.

Initial data may be entered into a field in test mode, in which case all field restrictions apply, or in draw mode, in which case anything may be entered. Initial data may also be moved or copied onto a field from a display area; see Section 5.3. When a screen is saved, data contained in a field are saved with the screen, and will be displayed as the field's initial value when the screen is displayed at run-time. There is one exception: contents of date and time fields that default to system values are not saved (see Sections 6.8.2 and 6.8.2).

To modify field characteristics, including the size of a compiled field, press the PF4 key with the cursor located anywhere within the field. (If the field is protected against tabbing, you will not be able to move the cursor there in test mode; use the PF2 key to get into draw mode, in which all fields are

accessible.) The field characteristics menu, shown in Figure 9, will pop up near your field. Each option on the menu is described in great detail in subsequent sections.

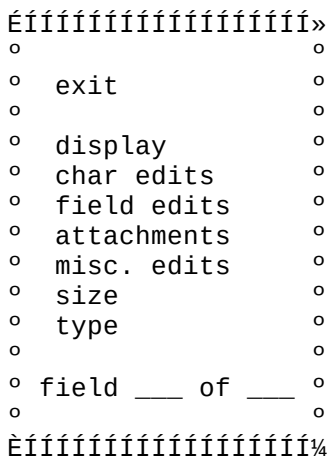


Figure 9: Field Characteristics Menu

## 6.2 Deleting, Moving, and Copying Fields

The PF6, PF7, and PF8 keys will delete, move, and copy fields as well as display data (see Section 5.3). A field can be moved or copied anywhere within a screen, as long as the new location doesn't overlap a side border or another field. Note that it is possible to overlay, and thus lose, display data.

When a field is deleted, moved, or copied, the field's current contents go with it. When a field is moved, it retains its display attributes and restrictions on data. When a field is copied, its display attributes and restrictions on data are copied also. The sole exception to this rule is that field names (Section 6.7.1) are not copied. An array of fields (Section 6.3.2) is always deleted, moved, or copied as a unit.

Since the data modification keys (Section 3.4) are applied to data entered within a field, they cannot be used to change the extent of the field itself. Changing the length of a field is described in Section 6.3.

## 6.3 Field Size

The simplest sort of field you can define with JAM is just part of a single line on the screen. Every field takes up at least one space on the screen; but fields can be made much larger, larger even than the physical display. You can extend a field horizontally by making it shifting; then only a portion of the field's contents will be visible on the display. You can extend a field either vertically or horizontally by making it an array, and you can expand an array offscreen by making it scrolling. A field's size, then, has four parameters:

1. its visible, onscreen length;
2. its total length, which is greater if it's shifting;
3. its number of onscreen array elements;
4. its total number of data items, which is larger if the array is scrollable.

Here are definitions of a few more terms that occur frequently in the following pages.

Each of the fields that comprise an array is referred to as an element of the array. Each element occupies a fixed position on the screen.

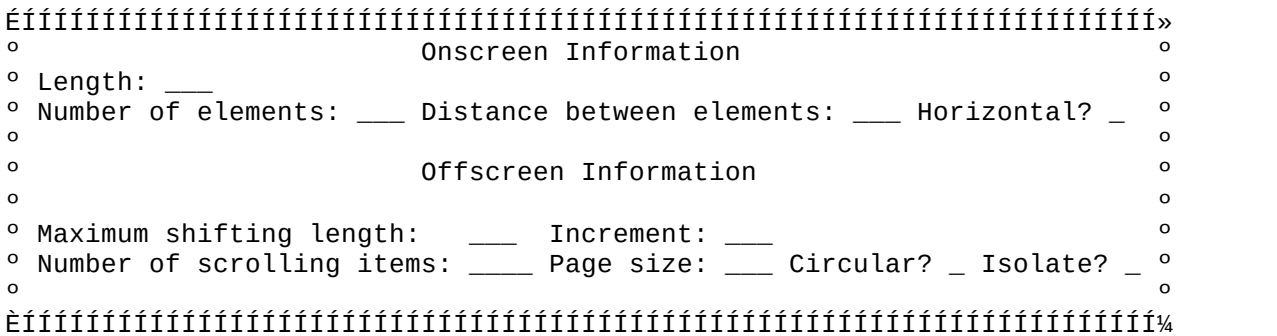


Figure 10: Field Size Window

If a field, or an array of fields, is scrollable, each individual entry is referred to as an item of data. Suppose an array of 3 elements is made scrollable, with a maximum of 10 items. When you begin entering data, item 1 is written to element 1, item 2 is written to element 2, and item 3 is written to element 3. After the third item is written, scrolling moves the first item to an offscreen buffer, moves item 2 into element 1, and moves item 3 into element 2; you then proceed to enter item 4 into element 3.

Although the appearance on the screen is quite different, from a program's point of view it doesn't matter whether a data item is entered into a non-scrollable array of 5 elements, or a scrollable array of 2 elements that allows a maximum of 5 items. To make those differences transparent to programs, the concept of an occurrence was introduced. In a scrollable field or array, each occurrence is an item, and an occurrence can be either on the screen or offscreen. In a non-scrollable array, each occurrence is an element, and it is always onscreen. A non-scrollable field that is not part of an array consists of a single occurrence.

To modify a field's size:

1. Position the cursor anywhere within the field.
2. Hit the PF4 key to bring up the field characteristics menu (Figure 9).
3. Choose size to bring up the field size window (Figure 10. It contains the field's current size parameters.

Here is how to use the field size window. Following sections contain more detailed explanations of arrays, shifting, and scrolling.

| | |
|---|---|
| Length | Enter the field's visible, onscreen length. If lengthening the field causes it to overlap another field, you will get an error message. |
| Number of elements | To make a field an array, enter a number greater than 1; the array will have this many onscreen elements. To reduce an array to a single field, enter 0 or 1. See Section 6.3.2. |
| Distance between elements | Enter the number of spaces between successive array elements, a number of lines (for vertical arrays) or columns (for horizontal arrays). The default value of distance is 1. See Section 6.3.2. |
| Horizontal? | For a vertical array, enter n (or leave blank); the new elements will appear below the field you are modifying. For a horizontal array, enter y, and the new elements will appear to its right. See Section 6.3.2. |
| Maximum shifting length | Enter the maximum possible length of data to be put into the field. Must be greater than the onscreen length. See Section 6.3.1. |

| | |
|---|---|
| Increment | Enter the number of characters by which the field's contents should be shifted when you arrow "beyond" its edge. See Section 6.3.1. |
| Number of scrolling items | Enter the greatest number of data items the field or array can possibly contain; must be greater than the number of onscreen elements. See Section 6.3.3. |
| Page size | Enter the number of items by which the PAGE UP and PAGE DOWN keys should scroll the field, or leave blank for default. This entry must be less than or equal to the number of onscreen elements; the default is one less, or one for a single scrolling field. See Section 6.3.3. |
| Circular? | Enter y if the scroll should wrap from bottom to top on down arrow, and from top to bottom on up arrow. If you enter n, those arrow keys will leave the field when the end of the scroll is reached. See Section 6.3.3. |
| Isolate? | Enter y if the field should not scroll when a parallel field scrolls, or n to include it in parallel scrolling. See Section 6.3.3.1. |

### 6.3.1 Shiftable Fields

A shiftable field is one that can hold data wider than the field's onscreen length. When a character is keyed into the last position of the field, the visible contents of the field are shifted left by the field's shifting increment, a user-defined constant, and the cursor is positioned immediately to the right of the last character entered. Data can be entered up to the length specified as maximum shifting length. When the maximum size is reached, the cursor tabs to the next field (unless tabbing has been inhibited by the no-autotab option or the return entry option).

When the cursor is at the first position of the field, and the beginning of the data is not visible, the left arrow key can be used to shift the field's contents right by one shifting increment. Similarly, when the cursor is at the last position of the field, the right arrow can be used to shift the field's contents to the left. Tabbing out of the field leaves its visible contents unchanged; tabbing or backtabbing to the beginning of the field resets its contents to the beginning of the data. If the cursor is in a protected shifting field, only the data (not the cursor) will move when you press the arrow keys.

If the screen positions immediately adjacent to a shifting field contain neither display data nor other fields, shifting indicators will appear on the screen whenever there are additional data offscreen to either the left or the right of the field's visible contents.

Fields comprising a shiftable array all shift together. Tabbing into one field of such an array, therefore, will reset the contents of all the fields (since the field that has been tabbed into must be reset).

### 6.3.2 Arrays

An array is a set of fields that can be treated as a unit. The fields that make up an array have the same length, common field characteristics, and starting locations separated by a constant vertical or horizontal distance. If you want a horizontal array of 15-character fields with five spaces between them, the distance must be 5; however, for a vertical array with one blank line between elements, the distance must be 2.

Any change to the specifications of a field within an array changes the whole array. The PF6, PF7 and PF8 keys delete, move, and copy an array as a unit. Shifting and scrolling also apply to an array as a whole. Finally, only one

field name (Section 6.7.1) can be assigned to an array; individual array elements can be referenced by field name and element number, or by field number.

Arrays can be either horizontal or vertical, not both. You can achieve a spreadsheet effect by creating a group of parallel vertical arrays (see Section 6.3.3.1).

### 6.3.3 Scrollable Fields

Scrollable fields are usually defined as arrays (Section 6.3.2), but individual fields can be scrolled as well. A scrollable field or array displays the visible portion of a larger set of data items. A field can be both shiftable (Section 6.3.1) and scrollable.

The first data entered go into the field or array of fields visible on the screen. When the visible fields become full, the contents of the first disappear from the screen; if it is an array, the contents of each succeeding element move up into the previous element, and the last element is again available for data entry.

If you don't want to fill each field, the same scrolling effect can be achieved by hitting the RETURN or down arrow key. The field or array will scroll as long as there is more data to display. Scrolling in the opposite direction is achieved by hitting the up arrow key, with the cursor in the first field. The PAGE UP and PAGE DOWN keys will scroll the current scrollable array (or the scrollable array closest to the current cursor position) by the number of data items given in page size in the field size window. The TAB and BACKTAB keys normally have no effect on scrolling, and can cause the cursor to leave the scrolling area; see the section on next field edits for an example of how they can cause scrolling.

When a field or an array scrolls, every field or array parallel to it will also scroll; see Section 6.3.3.1.

You may define a scrolling field as circular. When you press down-arrow with the cursor on the last item of such a scroll, it will scroll to the first item instead of exiting the field. The RETURN and up-arrow keys wrap around in the same way. To insert a new item into a circular field or array, you must use the INSERT LINE key.

#### 6.3.3.1 Parallel Arrays

When a field or array is scrolled, any fields or arrays parallel to it will scroll too. See the preceding sections for a discussion of scrolling. Vertical (horizontal) scrolling arrays are considered parallel if

1. they start at the same line (column) of the screen;
2. the offsets between elements are the same;
3. they contain the same number of onscreen elements;
4. they have the same maximum number of scrollable items.

Single fields are considered parallel if they meet the first and last criteria (the others don't apply). Non-scrolling arrays are never considered parallel. Scrolling and shifting fields that are parallel do not shift together.

You can exclude a field that meets the above requirements from parallel scrolling by placing y in the isolate field of the field size window.

## 6.4 Field Display Attributes

Fields are displayed, by default, underlined and highlighted (if those attributes are available). To change the display attributes of a field:

1. Position the cursor anywhere within the field.

2. Hit the PF4 key to bring up the field characteristics window (Figure 9).
3. Choose display to bring up the display attributes window (Figure 8). The window will list the attributes supported by your display, and show which are currently in effect.
4. To turn on an attribute enter y; to turn it off, enter n. More than one may be active.
5. On screens that have color, the display attributes window has an additional option for color. To change the color of a field, enter y after modify color, and follow the procedure in Section 5.2.1.
6. Hit TRANSMIT to effect the change (or EXIT to abort).

At this point, the field characteristics window will still be displayed. You may choose exit or hit the EXIT key to close the window, or modify other field characteristics by choosing another option.

Note that in draw mode all fields appear underlined, and the contents of non-display fields are displayed. In test mode, fields are displayed with their real attributes.

The above procedure will change the attributes of a field after it has been compiled. To change the default attributes with which fields are initially created, see Section 4.4.

## 6.5 Character Edits

Character edits, or filters, are restrictions on what may be keyed into a field. For example, if a field is restricted to digits only, an attempt to enter a letter into the field will make the bell ring, and the letter will be discarded (unless you are in draw mode). The default edit is unfiltered, or all characters permitted. Defining a field as a currency field (Section 6.8.4) will not automatically make it numeric. To put character restrictions on a field:

1. Position the cursor anywhere within the field.
2. Hit the PF4 key to bring up the field characteristics window (Figure 9).
3. Choose char edits to bring up the character edits window (Figure 11). The current option will be shown in reverse video.
4. To change the option, either:

    a. Position the reverse video area to the desired option using the TAB, BACKTAB, space, BACKSPACE, or arrow keys, then hit TRANSMIT, or
    b. Hit the initial letter of the desired option (such as d for digits only).

Either choosing exit or hitting the EXIT key will close the window without changing the option. At this point, the field characteristics window will still be displayed; you may choose exit or hit the EXIT key to close the window, or set other field characteristics by choosing another option.

The above procedure will set the character edits of a field after it has been compiled. To set default character edits, see Section 4.4. The following character edits are available:

| | |
|---|---|
| unfiltered | allows entry of all characters, without restriction. |
| digits only | allows entry of the digits 0-9 only. In a normal (left-justified) digits-only field, no spaces may remain blank to the left of any digit entered. In a right-justified digits-only field, no spaces may remain blank to the right of any digit entered. A special feature of a digits-only field is that punctuation (any non-digit characters) within the field is skipped during normal data entry. The punctuation |

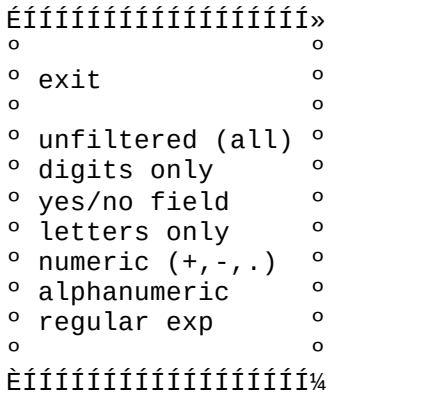


Figure 11: Character Edits Menu

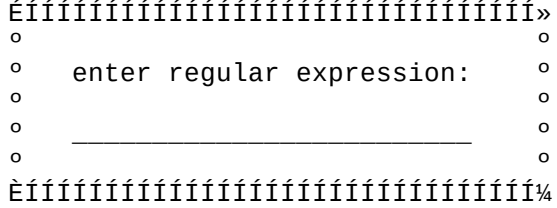


Figure 12: Regular Expression Window

characters themselves must be entered in draw mode, or before the field is set to digits-only; they cannot be deleted in test mode. This enables you, for instance, to define a digits-only social security number field, with the initial value - - . During data entry, the dashes are permanent.

yes/no — allows entry of only the initial letters of yes and no, normally y, Y, n, N, or space (which is converted to n). This edit actually uses the initial letters of the SM_YES and SM_NO entries in the message file. The displayable characters can be further limited by setting the field edits to upper or lower case (Section 6.6).

letters only — allows entry of the letters a-z, A-Z, and the space character.

numeric — allows entry of the digits 0-9, the plus sign, minus sign, and at most one decimal point. A plus or minus sign, if present, must be leftmost in the field. In a left-justified numeric field, no blanks are allowed to the left of any character entered; in a right-justified field, no blanks may occur to the right of any character entered.

alphanumeric — allows entry of the digits 0-9, the letters a-z and A-Z, and the space character.

regular exp — allows entry according to a flexibly defined template, which you enter in the window of Figure 12. A description follows in Section 6.5.1.

### 6.5.1 Regular Expressions

A regular expression is a pattern or template made up of characters. It divides ordinary character strings into two kinds: those that match the pattern, and those that don't.

JAM supports regular expressions in the style of the UNIX editors, and uses them to check that the contents of a field conform to a pattern. You can define the pattern in a way that is extremely flexible. Other JAM character edits, such as numeric, force every character entered in a field to belong to the same type; with regular expressions, you can restrict different parts of the field to different character types, or classes.

When JAM checks a field against a regular expression, it steps through the field data and the regular expression together. It matches as many field characters as it can against the first subexpression before going on to the next, and quits at the first mismatch.

Here is an example of a regular expression. This one defines a sort of ID number that is three digits, followed by a dash, followed by at least three letters or numbers, up to the length of the field:

```
[0-9]\{3\}-[a-zA-Z0-9]\{3,\}
```

If you didn't understand that, read on.

#### 6.5.1.1 Character and Field Regular Expressions

You may install a regular expression as a character edit on a field; then each character typed into that field causes the whole field to be checked against the regular expression. If the check fails, JAM beeps and rejects the character, as with simpler character edits.

If, on the other hand, you install a regular expression as a field edit, then it is not checked until you tab from the field. If that check fails, JAM displays an error message and positions the cursor to the first character that failed to match.

You may combine a regular expression field edit with any character edit, including another (compatible!) regular expression. For instance, to prevent the entry of numbers with leading zeroes into a field, you could make the field digits-only and give it a regular expression field edit of

```
[1-9][0-9]*
```

The non-regular character edits are, of course, more efficient at run-time.

#### 6.5.1.2 Forming Regular Expressions

There are two kinds of rules for constructing a regular expression. One kind tells you how to form a simple expression, and the other tells you how to combine expressions into a more complex expression. The basics of regular expressions are quite simple; however, by combining them, you can quickly arrive at expressions that are quite complex. The following discussion, therefore, proceeds from simple rules for forming simple expressions to somewhat more complicated rules for combining and repeating expressions.

##### Simple expressions

The simplest regular expression is a single character, which matches itself: the regular expression z matches the string z. There are only a few characters that are special and do not match themselves; they are explained below. Blanks are not special, but they are not ignored either; a blank in a regular expression matches a blank in a field. (This includes leading blanks.)

A dot (.) is a special character; it matches any single character at all, including (but not limited to) itself.

The backslash (\) is also special. It is a quote character: it forces the following character to match itself, like an ordinary character, even if that

character is special. For instance, the sequence \. matches a dot, and only a dot; the sequence \\ matches a single backslash. The sequence \z matches a single z; here the backslash changes nothing.

Character classes

A group of characters between brackets ([]) matches a single occurrence of any of the characters; [13579] matches any odd digit, and [aA] matches an a of either case. The group of characters is called a character class. The order of characters in a class is not significant.

Long lists of consecutive characters can be abbreviated using a hyphen (-). For instance, [a-z] matches any lowercase letter, and [A-Za-z] matches any letter at all. (Owing to the nature of the ASCII collating sequence, [A-z] matches all letters plus some punctuation characters that fall between Z and a.) You may use any number and combination of characters and ranges within one set of brackets.

You can also negate a character class, that is, cause it to match any character except those between the brackets. Do this by placing a caret (^) immediately after the left bracket. The expression [^0-9+.-] matches any non-numeric character.

Note from the previous example that special characters other than ^-] are not special in a character class, i.e. between brackets. You do not need to quote dot with a backslash to include it in a character class.

All the non-regular JAM character edits can be simulated with character classes:

| | |
|---|---|
| unfiltered | . |
| digits only | [0-9] |
| yes/no | [YyNn] |
| letters only | [A-Za-z] |
| numeric | [+-][0-9.] |
| alphanumeric | [A-Za-z0-9] |

Concatenating subexpressions

The simplest way of combining two or more expressions is to put one after another. They then match whatever matches the first, followed by whatever matches the next, and so on. The expression JYACC matches the string JYACC; the expression a[0-9] matches a followed by a digit.

Repeating subexpressions

The star (*) causes the preceding subexpression to match zero or more characters that match the subexpression, instead of only one. The expression [0-9]* matches any number or none at all; [0-9][0-9]*, however, matches any number with at least one digit.

You can also give a more definite repeat count for an expression, enclosing it in quoted curly braces \{ and \}. The repeat count follows the subexpression, and has three possible forms:

| | |
|---|---|
| \{n\} | exactly n repetitions |
| \{n,\} | n or more repetitions |
| \{n,m\} | at least n repetitions, but no more than m |

For example, [0-9]\{5\} gives you a five-digit number, or an old-style zip code.

Repeat counts and the star are restricted to the kinds of expressions we have met so far; they may not be applied to grouped expressions, which are explained next.

Re-matching subexpressions

To re-match an expression or sequence of expressions, use quoted parentheses \( and \) around them. If you place a quoted number later in your expression, say \1, it will match whatever the first subexpression surrounded by \( \) matched. \2 will rematch the second grouped subexpression, and so on.

Note that \n does not reproduce subexpression n, but the actual character sequence that it matched earlier in the field data. The expression \([0-9]*\)\.\1 will match 123.123, or any other real number where the integer and fractional parts are the same; it will not match 123.45.

It is a confusing aspect of the backslash that it makes special characters ordinary (for purposes of matching), but also makes certain ordinary characters special (for purposes of grouping). C'est la guerre.

Some more examples

[iI][cC][Ee] matches ice, icE, iCe, iCE, Ice, IcE, ICe, or ICE.

212-[0-9][0-9][0-9]-[0-9]\{4\} matches a telephone number in Manhattan or the Bronx.

[0-9]\{3\}-[0-9]\{2\}-[0-9]\{4\} matches a Social Security number.

[a-zA-Z_][0-9a-zA-Z_]* matches an identifier in the C language.

[+-]\{0,1\}[0-9]*\.[0-9]* matches a floating point number, and [dDeE][+-]\{0,1\}[0-9]\{1,\} an exponent, in the FORTRAN language.

6.5.1.3 Summary of Special Characters in Regular Expressions

| | | |
|---|---|---|
| \ | backslash | makes any special character, including itself, ordinary; makes the ordinary characters {}() and numbers special, in certain contexts. . |
| | dot | matches any single character. [] |
| | brackets | surround a character-class subexpression. ^ |
| | caret | immediately following [, negates the character class - |
| | hyphen | within brackets, denotes a character range (unless last). * |
| | star | causes the preceding subexpression to match zero or more occurrences. \(\) |
| | quoted parens | surround an arbitrary subexpression. \[0-9] |
| | quoted numbers | rematch a previous subexpression enclosed by \(\) \{\} |
| | quoted curlies | surround a repeat count for the preceding subexpression. |

The caret (^) and dollar sign ($), which represent beginning and end of line respectively in the UNIX editors, do not have that meaning in JAM regular expressions.

6.6 Field Edits

Field edits generally control the processing of data that have already been keyed into a field. Thus, right justified and upper case modify the appearance of data on the screen, while data required and must fill call validation routines at field exit. To modify field edits:

1. Position the cursor anywhere within the field.
2. Hit the PF4 key to bring up the field characteristics window (Figure 9).
3. Choose field edits to bring up the field edits window (Figure 13). The window will show any field edits that are currently set.
4. To turn on a field edit enter y; to turn it off, enter n or space. More than one field edit may be active.

5. Hit TRANSMIT to effect the change (or EXIT to abort).

At this point, the field characteristics window will still be displayed. You may then choose exit or hit the EXIT key to close the window, or set other field characteristics by choosing another option.

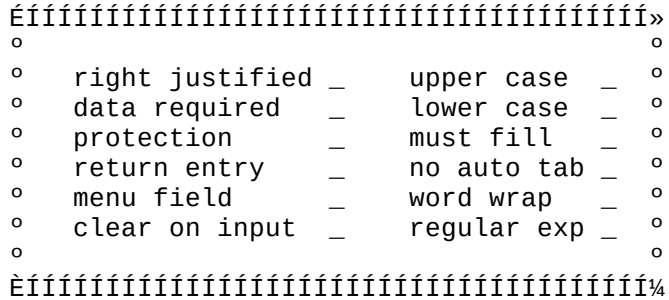


Figure 13: Field Edits

The following field edits are available. Unless otherwise stated, the actions described are performed by the library routine openkeybd.

### 6.6.1 Right-justified Fields

In a right justified field, characters are entered starting at the rightmost position. As each additional character arrives, previously entered text shifts one position to the left. When you tab into a right-justified field, the cursor goes to the right-hand edge.

If you activate insert mode using the INSERT key, characters entered in the middle of the field cause text between the cursor and the start of the field to be shifted left. The DELETE CHAR key shifts all characters to the left of the cursor one position to the right. The FIELD ERASE key clears the entire field, regardless of cursor position.

### 6.6.2 Required Fields

A required field is checked at field exit for the presence of at least one non-blank character. If the field is digits-only, it must contain at least one digit; punctuation doesn't count. This validation is performed when the last position of the field is filled, or when you attempt to leave the field using the TAB, NEW LINE, or arrow keys. If the field is blank, JAM displays an error message and resets the cursor to the beginning of the field (according to its justification).

### 6.6.3 Protected Fields

You can protect fields against the actions listed in figure 14, which pops up when you type y after protected in the field edits window. Protections may be combined arbitrarily; just type y after those you want to turn on. Use n or space to turn a protection off.

If a field is protected from data entry, all data characters typed there will be rejected, with a beep. Often it makes sense to protect such a field from tabbing into as well. One case where it does not is a circular scrolling field containing a list of choices, from which you select by scrolling to the appropriate one. Another is a shifting field, which a user must tab into in order to view the offscreen data.

If a field is protected from tabbing into, you will not normally be able to move the cursor there. The TAB, BACKTAB, arrow, and RETURN keys all cause the cursor to skip over tab-protected fields. Under certain circumstances, the arrow and

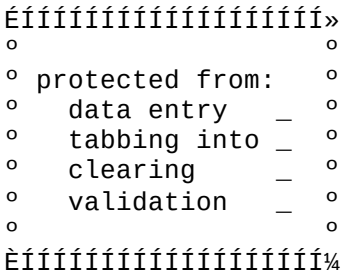


Figure 14: Field Protection Window

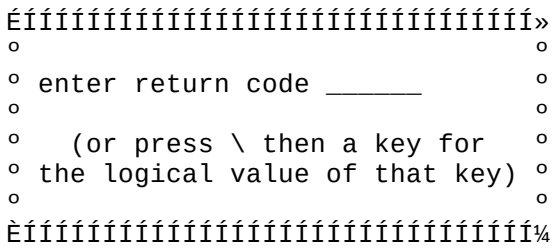


Figure 15: Return Code Window

HOME keys may place the cursor in a protected field; refer to the library functions ok_options and home, respectively.

If a field is protected against clearing, then the CLEAR ALL, ERASE, and DELETE CHAR keys, plus the clear-on-input field edit and certain library functions, will not erase its contents.

Finally, if a field is protected against validation, none of the normal field-exit checks will be performed, either when the field is tabbed from or when the entire screen is validated. Character edits, and certain field edits such as case and justification, will still be enforced.

In draw mode, all field protections are suspended.

### 6.6.4 Return-entry Fields

Note: return-entry fields are no longer used in menus. See the following section.

Normally, the library function openkeybd processes an entire screen, returning control to its caller only when the TRANSMIT key or another "returnable" key is struck. If your program needs to receive control at a certain field, however, you may designate that field as return entry. When such a field is filled or tabbed from, openkeybd validates the field, leaves the cursor there, and returns immediately to its caller.

When you type y after return entry on the field edits menu, the window in Figure 15 will pop up. In this window you may enter an integer code for openkeybd to return from the field; this makes it easy for your program to tell what field to process. You may enter the return code in any of several formats:

- a decimal number: 50
- an octal number, with leading 0: 062
- a hexadecimal number, with leading 0x: 0x32

- an ASCII character, with surrounding apostrophes: '2'
- a JAM key mnemonic, as an alphanumeric string. You may also press the \ key, then the desired function key.

If you do not specify a return code in this window, openkeybd applies a set of default rules to determine its return value. When a field is return-entry, the application program must provide for positioning the cursor to the next field (perhaps by calling tab). If a field is return entry and scrollable, the application program must provide for scrolling, by calling rscroll. Refer to the library function openkeybd for more details.

### 6.6.5 Menu Fields

Note: menu fields perform the functions provided under Release 3 by return-entry fields in menus.

In menu screens processed by the library functions menu_proc and choice, and in item selection windows, only fields tagged with the menu field edit are eligible for selection. You can additionally choose the value to be returned by menu_proc, in the window that pops up when you enter y after menu field in the field edits window:

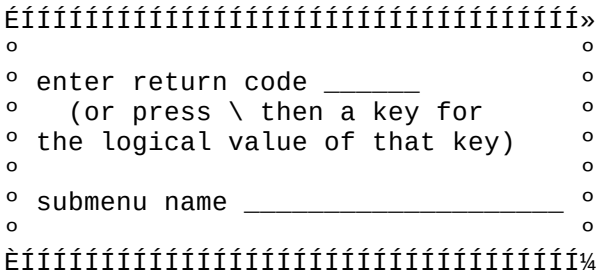


Figure 16: Menu Field Window

The rules for entering the return code are exactly as in Section 6.6.4. You may also enter the name of a sub-menu window, after submenu name, to achieve one level of pull-down menus; in this case, specifying a return code is useless, since the return value will come from the sub-menu.

Sub-menus work as follows. When the cursor enters a menu field with a sub-menu, the sub-menu window automatically pops up. All the cursor-motion keys operate as normally in the sub-menu, except the left and right arrows; they move to the next or previous selection in the main menu, and bring up a new sub-menu if it has one. Selecting an item from a sub-menu causes its value to be returned to the caller of menu_proc. Consequently, the main menu items themselves are never returned; your program only needs to worry about the return values of sub-menu items.

### 6.6.6 Clear-on-input Fields

For some kinds of fields, such as right-justified and currency fields, it is easier to type the field's contents afresh when a change is desired than to overwrite part or all of the field. Such fields may be designated clear-on-input; then, the first data character typed after the cursor enters the field causes it to clear, and that character stands alone in the field.

Right-justified and currency fields are not made clear-on-input by default.

### 6.6.7 Upper- and Lower-case Fields

These edits convert any alphabetic characters entered into the field to upper or lower case. This is valuable for consistency of appearance, and simplifies such tasks as database lookups. A field may be set to either upper case or lower case, but not both.

### 6.6.8 Must-fill Fields

A must-fill field is valid if it is empty, or if it contains no blanks whatsoever; leading, trailing, or embedded blanks with data characters are invalid.

### 6.6.9 No-autotab Fields

Normally, when you fill the last position of a field, the cursor will jump to the beginning of the next one. In a no-autotab field, however, the cursor will remain at the last position of the current field. Further input will either overwrite the last position or, if the OK_ENDCHAR option is set, be rejected with a beep (see ok_options). You must use the TAB, RETURN, or other cursor motion key to get to exit the field.

### 6.6.10 Word Wrap Fields

Word wrapping automatically transfers text from occurrence to occurrence of an array, to prevent "words" from being broken across lines. A "word" is simply a string of non-blank characters. Word wrap may be used only on fields that have been made scrollable (Section 6.3.3), or arrays (Section 6.3.2), or both. The fields may or may not be shiftable; they must be unfiltered (Section 6.5), and may not have a field regular expression. Within an array or field that uses word wrap, the following rules apply:

- Spaces separate words. You type text in freely; when a non-blank character occurs at the last position of a line, the entire word of which it is a part is removed from that line and inserted at the beginning of the next. If that line has insufficient room, the end of that line is also wrapped; wrapping propagates, if necessary, to the end of the array or scroll. If there is insufficient room on the last line, nothing moves, and the terminal beeps.
- Blank lines separate paragraphs. Whenever word wrap moves text into a blank line that is followed by additional lines of text, the blank line is preserved, and all following lines of text move down by one.
- If the DELETE CHAR key is hit and there is no text to the right of the cursor, as much text from the following line as will fit is moved to the cursor's current position; subsequent text moves up appropriately. Hitting DELETE CHAR at the beginning of an empty line deletes that entire line. DELETE LINE will delete a full line of text.
- The RETURN key positions the cursor to the next line of text within the current array or scroll, regardless of other fields on the screen, until the last (maximum) line is reached. It then positions the cursor to the first field past the current line.
- In insert mode only, the RETURN key opens up a line. Any text following the cursor on the same line is moved to the next line, and any additional lines of text are moved down by one. The INSERT LINE key does the same thing, whether insert mode is active or not.

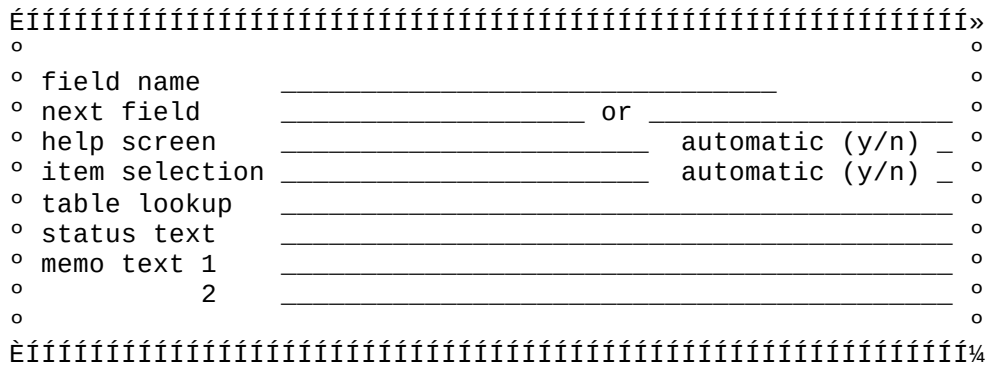


Figure 17: Field Attachments Window

### 6.6.11 Field Regular Expressions

You may attach a regular expression to a field, thus requiring that its contents match the regular expression when you exit the field. When you enter y after regular expression in the field edits window, a small window (Figure 12) pops up, prompting you to type in the expression there.

See Section 6.5.1 for a full explanation of regular expressions.

## 6.7 Field Attachments

Field attachments are chiefly names of other objects associated with a field, such as its help screen, although the field's name is itself an attachment. You can alter a field's attachments by bringing up the window in Figure 17, thus:

1. Bring up the field characteristics menu by hitting PF4.
2. Choose attachments. The field attachments window will be brought up, with current attachments (if any) displayed; see Figure 17.
3. Enter or erase the desired data, as decribed in subsequent sections.
4. Hit TRANSMIT to save the field attachments, or EXIT to abort.

### 6.7.1 Field Name

A field name entry assigns the field a name, to which an application program can then refer. The first character of the name must be alphabetic, and the rest must be either alphanumeric or underscores; no blanks are allowed. Upper and lower case letters are considered distinct, so that "field1" and "Field1" are different names.

Although field names are optional, they are strongly recommended. An application program can also refer to fields by number, which may be more convenient at times. However, names can be permanently assigned to fields, while field numbers change whenever a field is moved, inserted, or deleted from a screen (Section 6.1). Fields must be named in order to take advantage of local data block processing. Names are also more convenient if you plan to generate programming language data structures from your screens.

Only one field name can be assigned to an array (Section 6.3.2); individual array elements can be referenced by field name and element number.

When a field is copied (Section 6.2), the new field retains all the characteristics of the original except its name.

### 6.7.2 Next Field

A next field entry designates the field to be tabbed to when this field is exited. Normal tabbing is from left to right and top to bottom (the same order as field numbering), except that tab-protected fields are ignored. A next field entry is ignored if the target field is nonexistent or tab-protected; in such cases, the next-field designation is said to fail. The field attachments window provides for two next-field designations; if the first fails, the second is tried.

The next-field designation does not affect the BACKTAB or RETURN keys. It may affect the right-arrow key, if horizontal arrow behavior is set to OK_TAB using ok_options; other arrow keys are unaffected.

Next fields can be designated by name, or by absolute or relative number. Field numbers are assigned during screen compilation (Section 6.1); precede them by a # sign (#1, #14) for absolute, or a plus or minus (+1, -5) for relative. Use either +0 or -0 to designate the current field. Names are assigned using the field attachments window (preceding section), and should be entered without adornment. If you specify a next field by number, and later insert or delete a field, the resulting tab operation may be quite different from what you intended.

You can designate a particular array element or scrolling item by attaching a subscript, the occurrence number, to a field designation. The number may again be either absolute (no sign) or relative (plus or minus sign). It may be attached in two ways: enclosed in brackets (alpha[5], beta[+1], #5[1]), or preceded by a colon (alpha:5, beta:+1, #5:1). The colon form is obsolescent. Occurrences are explained in Section 6.3.

If a next-field edit belongs to an array or scrollable field, then whenever the operator tabs from an occurrence of the array or field, the same next-field designation is used. In this case, the current occurrence number is saved; if the designated next field is also either scrollable or part of an array, but the next-field edit contains no occurrence, the saved occurrence is tried. If it is greater than the number of occurrences in the destination field or array, the next-field option fails.

As an example of next-field use, suppose you have two parallel scrolling arrays named array1 and array2, and you wish the cursor to tab through them column-wise (the default is row-wise). You would designate the next fields as follows:

```
                    array1         array2

     primary        array1[+1]     array2[+1] alternate
                    array2[1]      array1[1]
```

These designations would move the cursor to the next occurrence of each array, until the last was reached; then the primary designation would fail, and the secondary would take the cursor to the first occurrence of the other array. If the arrays are scrolling, this will also cause the TAB key to scroll them.

### 6.7.3 Help Windows

The help screen option enables you to name a window to be displayed whenever the HELP key is hit while the cursor is within the field. If automatic help is specified, the help window will be displayed as soon as the field is tabbed into, if the field's contents have not been validated (i.e. its VALIDED bit is not set). The creation of help windows is described in Section 10.6. There are several types, including display-only text, menus for more detailed help, and help windows allowing data entry.

A help window can also be specified on a screen basis; see Section 4.6. You may specify both a help window and an item selection screen (Section 6.7.4) for a field, but they will conflict, with unpredictable effects.

The name of the help window may optionally be preceded by the location on the screen where it should appear. You write the line and column where the window's upper left-hand corner should go, enclosed in parentheses. The following example specifies a window named customer.hlp, to be placed at line 5 and column 10 of the screen:

```
help screen: (5,10)customer.hlp_____________
```

If you do not supply a location for the window, JAM will automatically bring it up where it does not hide the field it is attached to.

### 6.7.4 Item Selection

The item selection option enables you to name an item selection window which will be displayed whenever the HELP key is hit while the cursor is within the field. An item selection window contains a list of valid field entries, from which the operator selects one; the selected item is then copied to the underlying field, and the window closed. Selection operates according to the rules defined at choice in the Programmer's Guide; basically menu rules. The creation of item selection screens is described in Section 10.7.

If automatic item selection is specified, the item selection window will be displayed whenever the field is tabbed into and the field has not been validated since it was last changed. Although you may specify both a help window (Section 6.7.3) and an item selection screen for a field, they will conflict and the effects will be unpredictable.

Item selection, even if automatic, does not prevent an operator from entering data into the field ad lib. You may want to use a table lookup screen (see below) to restrict a field to some list of entries; in fact, you can use a single screen for both item selection and table lookup.

The name of the item selection window may optionally be preceded by the location on the screen where it should appear. You write the line and column where the window's upper left-hand corner should go, enclosed in parentheses. The following example specifies a window named areacodes, to be placed at line 12 and column 55 of the screen:

```
item selection: (12,55)areacodes_______________
```

If you do not supply a location for the window, JAM will automatically bring it up where it does not hide the field it is attached to.

### 6.7.5 Table Lookup

A table lookup window is very similar to an item selection screen, except that the list of items it contains is used to validate a field entry. The window is never actually displayed; the contents of the field being validated are compared with the items it contains, and the field fails validation if none matches. Windows used for table lookup validation are ordinarily also used for item selection.

### 6.7.6 Status Text

Status text is a message to be displayed on the screen's status line whenever the cursor is within its field. The message can be used as a prompt to the operator, explaining what entries are required or appropriate to that field. You can embed function key names and change display attributes within such a message; refer to the library function d_msg_line for details.

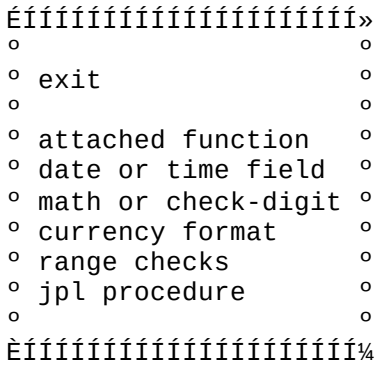


Figure 18: Miscellaneous Special Edits Menu

### 6.7.7 Memo Text

The memo text option enables you to mark fields for special processing by an application program. Nine lines of memo text are available, although only two are visible in the field attachments window at one time. You enter memo edits into a scrollable and shiftable array; you can get to the rest of the data with the arrows or the ZOOM key.

Any or all of these edits can store additional information about a field. Each edit is saved as a string, which can be retrieved using edit_ptr and examined by the application program. See the JAM Programmer's Guide for examples.

Memo edits are not validated in any way.

## 6.8 Miscellaneous Special Edits

There are several more field characteristics that require relatively detailed information from you. To get at them,

1. Position the cursor anywhere within the field.
2. Hit the PF4 key to bring up the field characteristics window (Figure 9).
3. Choose misc. edits to bring up the menu in Figure 18. Initially, exit is displayed in reverse video. To select an edit, either:

    a. Using the TAB, BACKTAB, space, BACKSPACE, or arrow keys, position the reverse video area to the desired edit, and hit TRANSMIT, or
    b. Hit the initial letter of the desired edit.

4. In either case, a new window will appear, depending on the entry. You can exit from the menu by following the above procedure, or by hitting the EXIT key.

The field characteristics window will still be displayed; you may choose exit or hit the EXIT key to close the window, or set other field characteristics by choosing another option.

### 6.8.1 Attached Functions

An attached function names an application function associated with a field, to be called by the JAM run-time system. A field may have two attached functions, which you enter in the window of Figure 19.

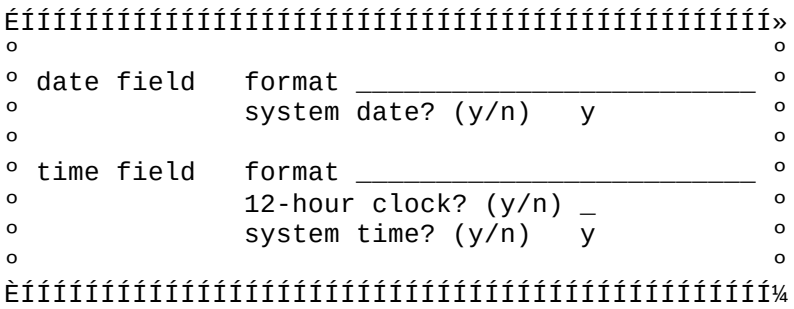


Figure 20: Date and Time Field Window

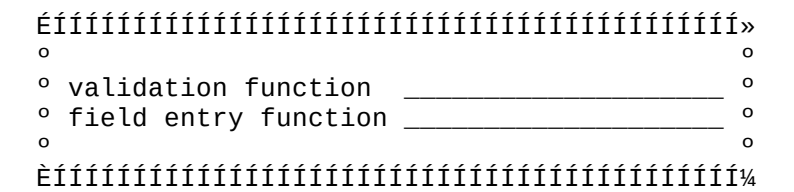


Figure 19: Attached Function Window

The validation function is called after field exit, late in the field validation process. It may be used to do any checking not supported by the other field edits available in JAM, or to set up other fields that are conditional on the present one. The field entry function is called when the cursor enters the field. With care, it is possible to use the same function for both field entry and exit.

Attached functions written in a standard programming language must be registered in a function list in order to be called at run-time; refer to the section on attached functions in the JAM Programmer's Guide. You can attach a procedure written in the JYACC Procedural Language (JPL) and stored in a file by typing jpl filename in the window; JPL procedure names need not be put into the function list.

### 6.8.2 Date and Time Fields

To alter or abolish date or time fields:

1. Bring up the miscellaneous edits menu (Section 6.8).
2. Choose date or time field. The date and time field window will be brought up, with the current date or time edit (if any) displayed. See Figure 20.
3. Enter or erase the desired data.
4. Hit TRANSMIT to save the date or time data, or EXIT to abort.

A date field displays a date in a user supplied format, such as mm-dd-yyyy or DD/MM/YY. The format is not validated. It is used to format a date, by making the following substitutions. (Except where otherwise noted, either upper or lower case may be entered without affecting the result.)

- MM is replaced by a two-digit numeric month; ZM is replaced by a month of one or two digits, with leading zero suppressed.
- DD is replaced by a two-digit numeric day of the month; ZD is replaced by a day of one or two digits, with leading zero suppressed.
- DDD is replaced by a three-digit numeric day of the year, e.g. 33 for Groundhog Day.

- YY is replaced by the last two digits of the year; YYYY is replaced by the entire year.
- MMM is replaced by a 3-character alphabetic abbreviation of the month name. The case of each M determines the case of the corresponding letter in the name. In July, for example, MMM would be replaced by JUL, and Mmm would be replaced by Jul.
- DOW is replaced by a 3-character alphabetic abbreviation of the day of the week. The case of each letter in the format string determines the case of the corresponding letter in the day's name. On Wednesday, for example, dow would be replaced by wed, and dOw would be replaced by wEd.

All other characters are put literally into the formatted date. The above items may be supplied in any order, and any of them may be omitted or repeated within the field.

If system date is specified, the date initially displayed is the current date obtained from the operating system when the screen is brought up on the display. If the field is not protected from clearing, you can refresh the date by hitting the ERASE or CLEAR ALL key. Alternatively, you may enter a date which is then validated by the utility as to both content and format. If no system date is specified, the field is not initialized with the current date, but any date you enter will be validated as to both content and format.

A time field displays a time of day in a user-supplied format, such as hh.mm.ss or HH:MM a.m. The format string is not validated. It is used to format the time, by making the following substitutions. (Either upper or lower case may be entered without affecting the result.)

- HH is replaced by a two-digit hour; ZH is replaced by the hour, with leading zero suppressed.
- MM is replaced by a two-digit number of minutes; ZM is replaced by the minutes, with leading zero stripped off.
- SS is replaced by a two-digit number of seconds; ZS is replaced by the seconds, with leading zero suppressed.
- If the time is after noon, AM or A.M. is replaced by PM or P.M. If the time is before noon, PM or P.M. is replaced by AM or A.M.

All other characters are put literally into the formatted time. The above items may be supplied in any order, and any of them may be omitted or repeated within the field.

The time is given according to a 24 hour clock; if the format includes AM, A.M., PM, or P.M., they are treated as constant text and displayed as is.

If system time is specified, the time initially displayed is the current time obtained from the operating system when the screen is brought up on the screen. If the time field is not protected from clearing, you can refresh the time by hitting either the ERASE or the CLEAR ALL key. Alternatively, you may enter a time which is then validated by the utility as to both content and format. If no system time is specified, the field is not initialized with the current time, but any time you enter will be validated as to both content and format.

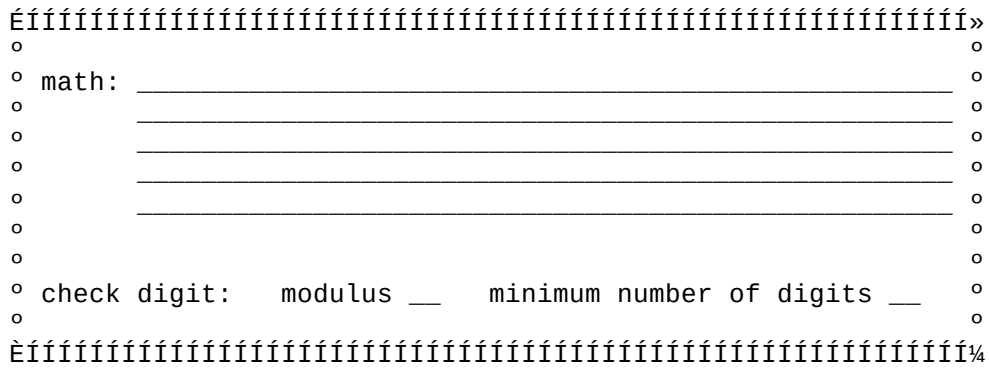


Figure 21: Math and Check-Digit Window

### 6.8.3 Math and Check Digit

To create a math or check digit edit:

1. Bring up the miscellaneous edits menu (Section 6.8).
2. Choose math or check-digit. The math and check-digit window will be brought up, with the currently set math or check digit edit, if any, displayed; see Figure 21.
3. Enter or erase the desired data.
4. Hit TRANSMIT to save the edit (or EXIT to abort).

The check digit option causes the field to be validated according to a standard check digit algorithm. Two algorithms, mod-10 and mod-11, are automatically supported, but others can be added. (The check digit routine, ckdigit, is included in the JAM library. A detailed description can be found in the source code, which is included with the package.) Non-numeric characters in the field are ignored.

The math option causes JAM to evaluate one or more user-defined calculations and place the results in fields. The calculations are performed when you fill or tab out of the field to which the expressions are attached, or hit the TRANSMIT key upon completing the screen. The calculated values are stored in whichever field you designate or in the LDB; the fields to which the expressions are attached need not appear in the expression. Multiple expressions must be separated by semicolons, even if they are on separate lines in the window.

A math expression starts with an optional floating point size specification for the destination field. This specification has the form %m.n where m specifies the total number of characters in the output and n the number of digits after the decimal point. If no size is supplied, the total length defaults to the length of the destination field. The number of decimal places defaults to that given in a float or double data type edit attached to the destination field (see Section 6.9); if there is none, to the number of decimal places in an amount edit attached to the destination field (see Section 6.8.4); or to 2 if there is neither.

The optional size specification is followed by the destination field designation, an equal sign, and the body of the expression. The expression body can contain numeric constants, field designations, parentheses, and the arithmetic operations + - * / and ^ (raise to a power).

Fields and occurrences in math expressions may be designated by name or by absolute or relative number (preceded by the sign #), with an optional occurrence number. If the calculation is attached to an array or scrolling field, it is performed every time you fill or tab out of an occurrence of the array or field. In this case, the current occurrence number is saved as a default. If any field specified in the math expression is either scrollable or

part of an array but no associated occurrence number is supplied, the default number is used, if possible. If the default number is greater than the number of occurrences in the specified field or array, an error results.

Typical math expressions look like this:

```
%8.0 #3 = #1 * 12 + #2
fielda:2 = (fielda:1 - 6.235) / fieldb:1
```

In math expressions, the designation for the "current field" is #+0; the designations for the fields preceding and following the "current field" are, respectively, #-n (= n fields before the current field) and #+n (= n fields after the current field). As an example of this notation, consider the following two math expressions:

```
#-3 = #+3 * #+0

#-3 = #+0+6
```

In the first math expression, the field that occurs three fields before the current field is set to the value obtained by multiplying the current field's value by the value of the third field after the current field. Thus, if #+0 = 10 and #+3 = 5, then #-3 = 50. In the second math expression above, the value of the field three fields before the current field is equal to the sum of the current field's value and 6 (e.g., #-3 = 16, if #+0 = 10).

There are three special functions: @sum, @date, and @abort. @sum yields the sum of all occurrences in an array or scroll:

```
@sum array1
@sum #2
```

@abort followed by a number in parentheses passes the number to the library function isabort, q.v. That function causes the JAM library to return control to the application as quickly as possible:

```
@abort(1)
```

@date yields the number of days elapsed between 1/1/1753 and the date given in the fon fields after the current field item. The fon fields after the current field item may be a field name or number, or a literal date enclosed in parentheses. Literal dates must have the format MM/DD/YYYY. For instance:

```
@date quarterday
@date #-1
@date (3/31/1985)
```

An error results if the field designated by @date is not a date field, or does not contain month, day and year somewhere in its format. If the destination field is a date field, the number resulting from the calculation is interpreted as a number of days elapsed since 1/1/1753, and the resulting date is displayed according to the date field format. If field1 and field2 are both date fields,

```
field2 = @date field1 + 30
```

will set the date in field2 to 30 days past the date in field1.

### 6.8.4 Currency Formatting

A currency or amount field is formatted specially after you tab out. To create or change a currency format:

1. Bring up the miscellaneous edits menu (Section 6.8).

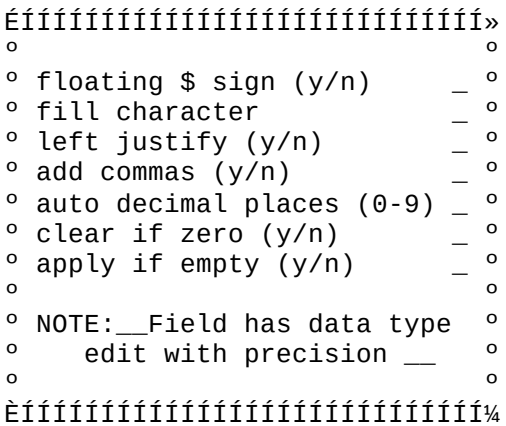


Figure 22: Amount Edits Window

2. Choose currency format. The window of Figure 22 will be brought up, with the current amount edit, if any, displayed.
3. Enter or erase the desired data. To get rid of a currency format altogether, use the CLEAR ALL key, or erase each field individually.
4. Hit TRANSMIT to save the edits, or EXIT to abort.

The following fields after the current field format parameters are supported:

| | |
|---|---|
| floating $ sign | Enter y to insert a floating dollar sign, n or space to omit it. (For a fixed dollar sign, omit this option; enter a $ as display data on the screen, in the last character position preceding the field.) |
| fill character | Enter a fill character, if desired, or leave blank. The character specified will replace any blanks in the field. |
| left justify | Enter y for left justification, n or blank for right justification. |

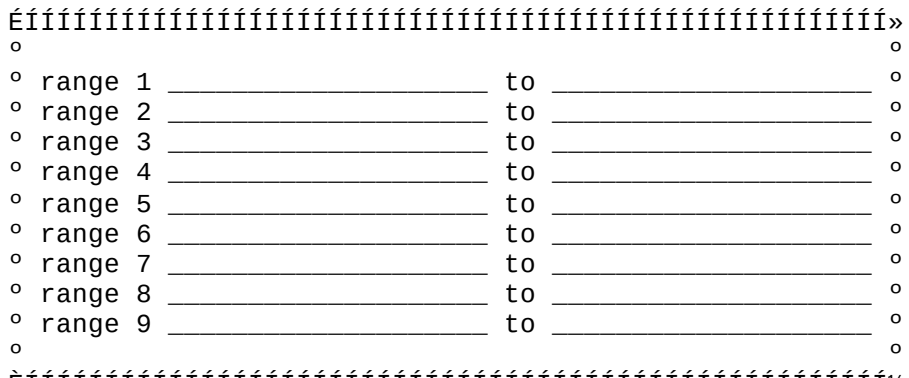


Figure 23: Range Window

| | |
|---|---|
| add commas | Enter y to insert commas, n or blank to omit them. |
| auto decimal places | Enter the minimum number of decimal places to be displayed. More decimal places will be displayed if keyed in during data entry. |
| clear if zero | Enter y if the field should be cleared to blanks when the amount is zero, n or blank for a formatted zero amount, such as *******0 or 0.00. |
| apply if empty | Enter y if the amount edit should be applied to empty fields, n or blank if the edit should be skipped. |

A 15-character-wide field with the fon fields after the current field currency format parameters:

```
floating $
fill with "*"
right-justified
add commas
two decimal places
```

will produce the fon fields after the current field transformations:

```
Plain     Formatted

123456789 $123,456,789.00
12.345    *********$12.35
```

### 6.8.5 Range Checks

To create or change range edits:

1. Bring up the miscellaneous edits menu (Section 6.8).
2. Choose range checks. The range window will be brought up with the current ranges, if any, displayed (Figure 23).
3. Enter or erase the desired data.
4. Hit TRANSMIT to save the ranges, or EXIT to abort.

The range option enables you to enforce minimum and maximum values for a field. If the field has been defined as digits-only, the values entered will be treated as integers. If the field is numeric, the values will be interpreted as floating point numbers. In all other cases, the values will be compared as character strings.

Range endpoints are restricted, both in length and character filter, to valid field entries. A field may have up to nine permissible ranges; you may omit

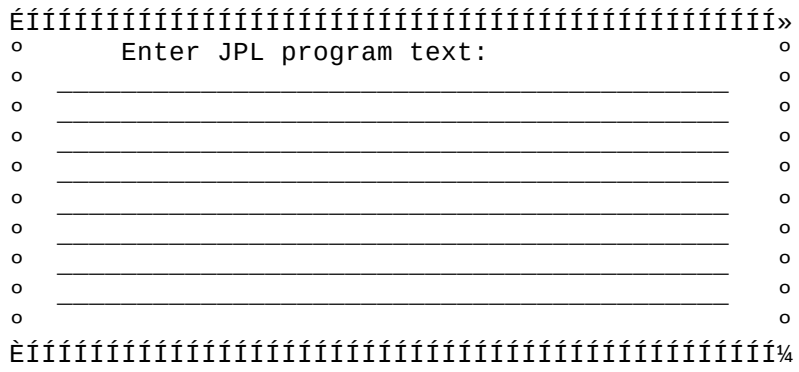


Figure 24: Attached JPL Procedure Window

either the lower or the upper limit for any of them. An empty field is considered in range; if data entered are out of range, the run-time system displays an error message on the status line.

### 6.8.6 JPL Procedure

The JYACC Procedural Language, or JPL, is an interpreted programming language designed especially for use in JAM screens. You can attach JPL statements directly to a field, using the window in Figure 24

To create or alter an attached JPL procedure:

1. Bring up the miscellaneous edits menu (Section 6.8).
2. Choose jpl procedure. The window of Figure 24 will be brought up, with the current JPL code, if any, displayed.
3. Enter or erase the desired data. The window contains a scrolling and shifting array, in which you type your JPL program.
4. Hit TRANSMIT to save the changes, or EXIT to abort.

The attached JPL code will be executed during field exit processing, directly after the field validation function. Refer to the JPL Programmer's Guide for how to write JPL programs. As a special case, you can execute a JPL procedure stored in a file by typing jpl filename in this window.

## 6.9 Data Types

JAM includes a utility, f2struct, that can generate a prgramming language data structure corresponding to the fields of a screen; see the Configuration Guide for usage. You can use the screen editor to assign data types to fields which that utility understands. To supply a field with a data type, or to exclude it from the structure:

1. Position the cursor anywhere within the field.
2. Hit the PF4 key to bring up the field characteristics window (Figure 9).
3. Choose type to bring up the data type window (Figure 25). Note that the labels in this window are taken from the message file, where you may adapt them to suit your primary programming language; the figure uses C language labels, as distributed by JYACC.
4. To set or change the option, either:

    a. Using the TAB, BACKTAB, space, BACKSPACE, or arrow keys, position the reverse video area to the desired option, and hit TRANSMIT, or
    b. Hit the initial letter of the desired option (o for omit, etc.). Either choosing exit or hitting the EXIT key will close the window without changing the type.

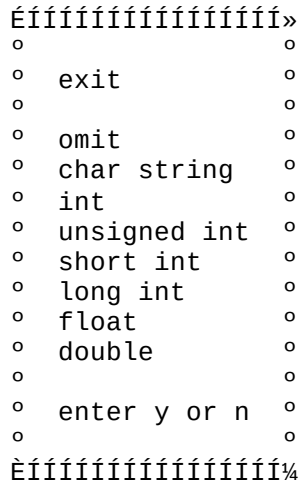


Figure 25: Data Type Menu

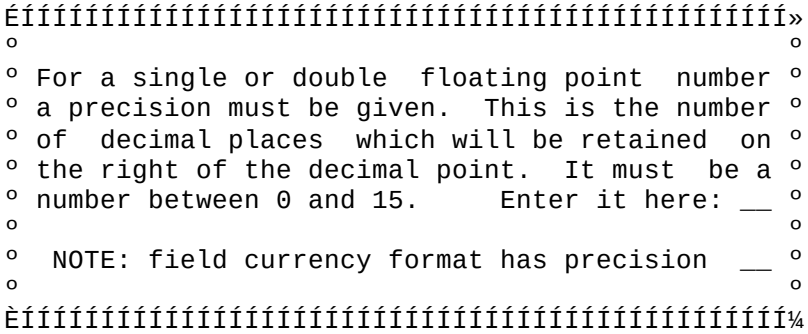


Figure 26: Precision Window

If you select a real data type (float or double in the figure), the precision window (Figure 26) will pop up. The default precision is the number of decimal places in the currency edit, if any, or 2 otherwise. To change the precision, enter the desired number and hit TRANSMIT. Hitting the EXIT key will leave the old value in effect.

## 6.10 Field Summary Window

jxform provides a field summary window, a convenient way to inspect and change certain frequently modified field characteristics without wading through the PF4 menus. The window appears in Figure 6.10; to use it, you

1. Position the cursor anywhere in the field.
2. Press the PF5 key.
3. Enter any changes in the window. Only items in the first two lines may be modified; the rest are display-only, and are formatted similarly to a screen listing.
4. Hit the TRANSMIT key to commit your changes, or EXIT to discard them.

While the field summary window is open, you may press the PF4 key at any time, and alter field characteristics from its menu in the usual way. However, if you subsequently hit the EXIT key in the summary window, those changes will be lost.

In the remainder of this section we describe quickly how to use the window. The field characteristics themselves are fully described elsewhere in this chapter, and a reference is provided for each.

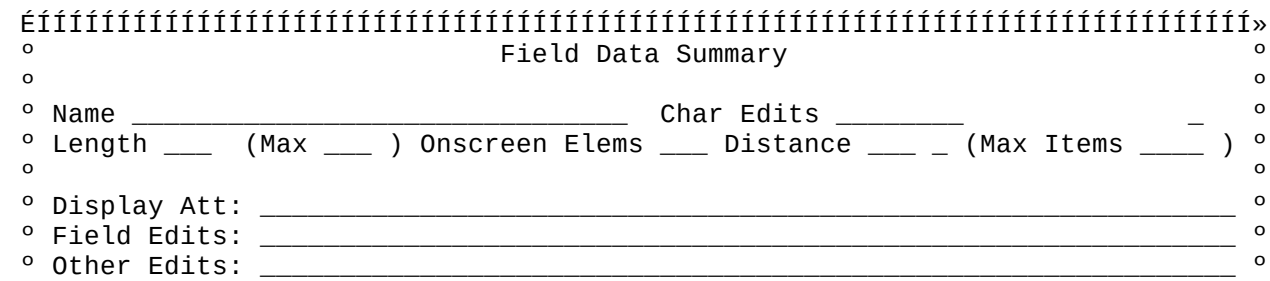


Figure 27: Field Summary Window

| | |
|---|---|
| Name | The field's name, if any, is displayed here, and you may simply type in changes. (Section 6.7.1) |
| Char Edits | This is a circularly scrolling field (Section 6.3.3) displaying the field's character filter; use the up- and down-arrow keys to select the one you want. The one that is visible when you hit TRANSMIT on the window will be used. (Section 6.5) |
| Length | The field's onscreen length; type in any changes. (Section 6.3) |
| (Max) | The field's maximum shifting length. Modify directly. (Section 6.3) |
| Onscreen Elems | The number of onscreen array elements. Modify directly. (Section 6.3) |
| Distance | The distance from one array element to the next. Modify directly. (Section 6.3) |
| (Max Items) | The maximum number of scrolling items. Modify directly. (Section 6.3) |
| Display Att | All the field's display attributes. Not modifiable; refer to the lstform utility if you need an explanation of the attribute mnemonics. (Section 6.4) |
| Field Edits | Mnemonics for the field edits. Not modifiable; see lstform. (Section 6.6) |
| Other Edits | Mnemonics for the field's attachments and other special edits. Not modifiable, and the values of the edits are not given. (Sections 6.7, 6.8) |

# 7 JAM Control Links

## 7.1 Control Strings and Control Fields

The JAM run-time environment determines how to respond to events based on information contained in JAM control strings on the current screen. An event is generally either a menu selection or a function key stroke, although there are a few automatically generated events. For each event of interest, you define a specially placed control string:

- For a menu selection, the control string goes in an onscreen control field immediately fon fields after the current field the field containing the selection.
- The control string for a function key goes into a section of the screen that is not displayed; it is named after its function key.

A detailed explanation of how to create control strings appears a little further on in Section 7.3; we turn first to the contents of control strings.

## 7.2 Link Actions

The first character of a JAM control string designates a kind of action for the run-time system to take; the remainder can be thought of as parameters to the action. There are four possible kinds of action:

| Lead Character | | Link Action |
|---|---|---|
| caret | ^ | call a function exclamation point |
| | ! | run a program ampersand |
| | & | display a window other |
| | display a form | |

### 7.2.1 Display a Form

Any control string that begins with a character other than ampersand, exclamation point, or caret is treated as a display-form command. The entire control string is assumed to be the name of a screen created with JAM. The name may be converted to upper case letters, and may have an extension appended to it, according to parameters in the setup files. If a screen with the given name exists, JAM displays it, replacing the previous screen; otherwise, an error message appears.

JAM searches memory, active screen libraries, and directories on disk for all screens named in control fields, according to the rules of the library function r_window.

JAM maintains an ordered list of screens processed for the purpose of backtracking with the EXIT key. When a display form command is executed, the new screen is added to the list. If a display screen is executed from a window, that window and all other open windows are closed before the new screen comes up. Therefore, if screen1 is a non-window and it invokes screen2 via the display screen command, pressing EXIT on screen2 will return control back to screen1. However, if screen1 was a window, it and all other open windows would have been closed before screen2 appeared, and EXIT from screen2 would return to the last form displayed prior to screen1.

Control strings are normally case-sensitive: their contents give the names of screens or routines, and JAM pays attention to case when searching for them. You can make the search insensitive to case by calling fcase, or by defining the SMFCASE setup variable.

### 7.2.2 Display a Window

If you want a screen to appear as a window overlaying part or all of the current screen, rather than replacing it, begin the control string with an ampersand, followed by the name of the screen to be displayed. The conventions for appending a file extension and for searching directories are the same as in the display form command (see the previous section).

The position of the window may optionally be specified, by including the row and column of the upper left corner of the window after the name of the screen. The row and column are separated from the screen name and each other by spaces. (You may have to lengthen the control field to specify a position.) If the position would cause the window to extend past the end of the screen, JAM will adjust the position (at runtime) so that the entire window will be shown.

For example, the control string in example 1 below would display SCR1 as a window, at the current cursor position. If you wanted the window to display at line 3, column 20, you would add coordinates as in example 2. The line and column can optionally be enclosed in parentheses and separated by a comma (example 3); such coordinates may also be placed before the screen name (but after the ampersand).

1. &SCR1
2. &SCR1 3 20
3. &SCR1(3,20)
4. &(3,20)SCR1

You must use the syntax of example 4 to position argument windows in system and function call control strings. Screens to be displayed as windows have some special requirements; see section 10.4.

### 7.2.3 Execute a Program

If you want an event to cause a program to execute, the associated control string should begin with an exclamation point. The remainder of the field is then interpreted by JAM as an operating system command, and passed to the command interpreter.

If the field contains a percent ("%") character followed by a screen name, the screen is displayed as an argument window prior to invoking the command. The fields in the window are concatenated and merged into the control field, replacing the "%", the name of the prompt screen, and trailing spaces; the whole is then used to invoke the command. For example, either of the fon fields after the current field strings would invoke the MS-DOS COPY command, with the verify (/V) option and arguments from a window named cp:

```
!COPY %CP /V
!COPY %(10,35)CP /V
```

The second example specifies the position of an argument window, by including the row and column between the percent sign and the window name. That is the only syntax for specifying the position of an argument window, since anything fon fields after the current field the window name will be passed to the operating system as part of the command line. More than one argument window may be included in the control field. The rules for appending a file extension, and for searching directories for screens, are the same as in the display screen commands. Creating argument windows is described in section 10.5.

### 7.2.4 Call a Function

If you want a function to be executed when TRANSMIT or a function key is pressed or when a menu item is selected, the associated control string should contain a caret (^) in the first position. Characters between the caret and the first blank are then interpreted as the name of a function, to be invoked through a function list linked to JAM (see the Programmer's Guide).

If, after the function name, the control string contains a screen name preceded by a percent (%) character, the specified screen is displayed as a window prior to invoking the function. Its fields are then concatenated and merged with the contents of the control field, replacing the %, the name of the prompt screen, and trailing spaces, and the function is invoked with the whole as an argument. For example, the fon fields after the current field field value would invoke the function uprint using a window named pr.

```
^uprint %pr
```

Creating the windows for function arguments is described in section 10.5. The conventions for appending a file extension and for searching directories for screens are the same as in the display screen command.

There are several built-in functions for special tasks, which need not appear in the application's function list. They are documented in the Programmer's Guide.

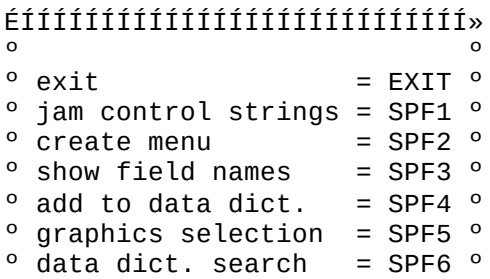


Figure 28: JAM functions menu

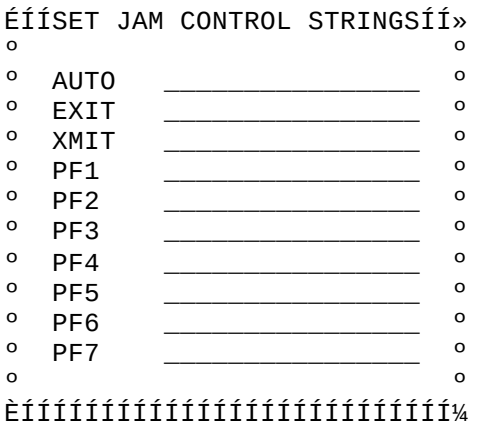


Figure 29: JAM Control Strings Window

## 7.3 Creating JAM Control Strings and Fields

jxform provides a number of function keys, SPF1 through SPF7, for creating JAM control fields. The same functions are also available through a menu window, invoked by the PF10 key (see Figure 28).

### 7.3.1 Function Key Control Strings (SPF1)

You create JAM control strings using the SPF1 key. It brings up a window containing a scrollable list of key-associated JAM control strings; Figure 29 shows the beginning of the list. It includes a first activity, AUTO; the TRANSMIT and EXIT keys, XMIT and EXIT; and function keys PF1 through PF24, SPF1 through SPF24, and APP1 through APP24. You simply type in your control string next to its key; you may find the ZOOM key useful if the control string is long.

When you enter a control string in the SPF1 window it is placed in a special section of the screen, and takes up no onscreen area.

The JAM first activity field (AUTO) is not associated with a key. It is executed automatically after a screen is brought up, but before the keyboard is opened for data entry. See Section 4.7 for a complete list of screen initialization steps.

### 7.3.2 Menu Control Fields (SPF2)

You can create a menu quickly by pressing the SPF2 key. A window similar to Figure 30 will pop up, and you may define the size of your menu and its control fields there.

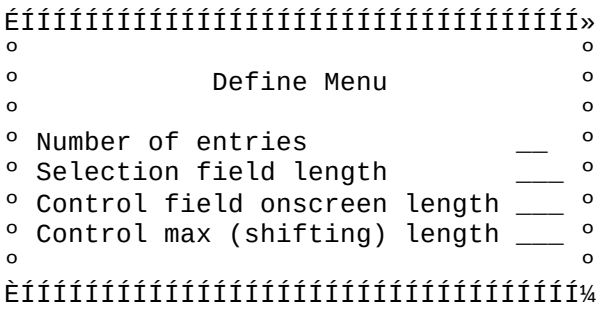


Figure 30: JAM Menu Control Fields Window

When you press TRANSMIT in the window, several menu selection field pairs (Number of entries) are created, beginning at the current cursor position. Each pair consists of an unprotected, menu field of Selection field length, and a protected, non-display field beginning one position to its right. The control field's onscreen and offscreen lengths are given by the last two entries in the window. The name of the menu field or array created by JAM is jam_menu; but if you add more menu selections that are not part of the array, they need not be named.

You should be sure to place the cursor so that the new fields will not overlap existing fields, or reach beyond the edge of the screen. If that occurs, no new fields will be created. You are, of course, not restricted to a vertical array of fields. Other arrangements, such as horizontal menus, or menus that are actually display text headings may be created as separate fields; see Section 10.1.1 for examples. You can use SPF2 to define a menu of one entry, then copy the fields thus created to other locations on the screen with the PF8 key.

Any screen with a field named jam_menu will be treated as a menu by the JAM run-time system. When such screens are processed, every menu field with a non-blank first character will be treated as a menu selection field. Each one should have a fon fields after the current field control field containing the processing rule for that selection field. Fields with no menu field edit and those that begin with one or more blanks are ignored in jam_menu processing.

### 7.3.3 Displaying Field Names (SPF3)

Pressing the SPF3 key causes all fields on the screen, including JAM control fields and other non-display fields, to be made visible; then the contents of all named fields will be replaced by their names. This enables you to look at all the control fields at once rather than checking each field individually to see what purpose it serves. When you press any function key, the display is restored to the state that existed before SPF3 was selected.

If a named field is too small to hold its entire name, the name will be truncated. For example, if the jam_menu field is only five characters long, jam_m will appear there.

### 7.3.4 Data Dictionary Search (SPF4, SPF5, SPF6)

These function keys enable you to use the data dictionary for creating and checking onscreen fields. Their use is explained in Section 8.

### 7.3.5 JAM Name Field (SPF7)

The JAM name field can help you learn screen names for use with the goto function (SPF3), as well as helping document an application. Whenever a screen is displayed that has a field named jam_name, JAM initializes that field to the name of the screen on which it appears. For example, suppose the form ABC.jam

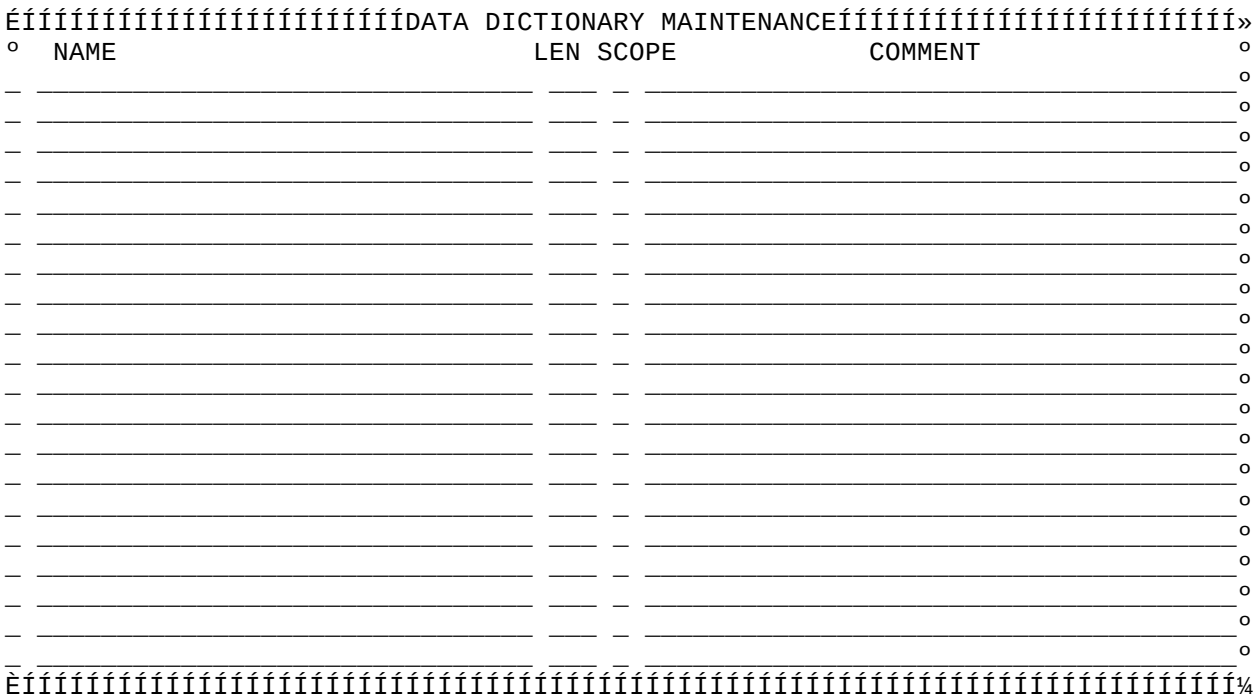


Figure 31: Data Dictionary Edit Window

includes a jam_name field. Whenever JAM displays the screen, the field will be given the value ABC.

Pressing SPF7 creates a jam_name field at the current cursor position.

## 8 Data Dictionary Editor

The JAM data dictionary serves two purposes. During application authoring and prototyping, it serves as a clearinghouse for field definitions. With field characteristics maintained in a central place, you can easily copy them into new screens and check them for consistency.

During application execution, the data dictionary supports the transfer of data between different parts of an application (data links). The JAM run-time environment keeps a subset of each item's data dictionary entry in memory, along with the item's current value. That value is automatically kept consistent with data entered through screens, and is copied into newly presented screens. This name-value map is called the local data block, or LDB.

You enter the data dictionary editor from the JAM authoring environment by pressing the SPF6 key. The data dictionary editor is not accessible from the screen editor, nor from the operating system. It reads a disk file, normally one named data.dic in your current directory. It then brings up a screen like Figure 31, displaying the data dictionary twenty items at a time. On the main screen it displays the name, scope, length, and a comment field for each:

| | |
|---|---|
| Name | A data dictionary entry's name is unique, and is used to identify it with a field on a screen. The name may be up to 31 characters long, and must begin with a letter; the rest must be letters, digits, or underscores. If you enter a name that already exists, an error message will appear. |
| Length | The length corresponds to field length in the screen editor. Unless the field is shiftable, it also determines the number of characters that will be reserved for the entry in the LDB. |

A screen field may be associated with a data dictionary element having a different length.

Scope — An item's scope groups it with other items, and controls its initialization in the LDB. The scope is a number from 1 to 9, with a default of 2; see section 8.1 for details. You can also enter an asterisk * in the scope field to create a data dictionary record; see Section 8.3.4.

Comments — The comments field usually contains a description of the element; it may contain anything at all. It may also be useful in searching the data dictionary for an element whose name is not known.

All the other field characteristics described in earlier sections of this chapter are stored in the data dictionary as well; you can examine and modify them, as in the screen editor, by pressing the PF4 key. The order of items in the data dictionary is entirely up to you; JAM does not sort them. After the last item there appears a line with EOF in the name field, and other fields blank.

## 8.1 Scope of Data Dictionary Entries

The scope of a data dictionary entry is a number between 1 and 9. Entries having the same scope can be cleared and initialized as a group, using the library functions lclear and lreset. Items having scope 1 are constant, that is, they cannot be altered at run-time. Initialization files may be listed in the SMININAMES setup variable for convenience.

Here is a situation in which multiple scopes are useful. Suppose you are executing multiple transactions on behalf of a customer. Suppose that after each transaction you wish to clear out certain variables, so they do not confuse the operator by appearing on the new transaction screens, but not to clear out the customer's name and address until you are finished with that customer. Give the customer name and address scope 2 (say), and the others scope 3. After each transaction, clear out the scope 3 variables using the library routine lclear(3); then, when all transactions for a customer are done, call lclear(2) to clear out the customer variables.

When JAM Release 3 data dictionaries are converted to Release 4 format, the Release 3 scopes are mapped as follows:

| Release 3 Scope | Release 4 Scope |
|---|---|
| Constant | 1 Global |
| | 1 Transaction |
| | 2 Local |
| | 3 |

## 8.2 Saving the Data Dictionary

When you are finished editing the data dictionary, press the EXIT key. An exit menu (Figure 32) will be displayed, with the following functions:

save — This function will write the changes that were made to the data dictionary to disk. If you want to save your changes, select this option before exiting. If you want to discard your most recent changes, do not select this function. rebuild This function writes the data dictionary to disk, then reinitializes the local data block index. A rebuild is necessary only if you want to continue the authoring session with an LDB that reflects the latest changes to the data dictionary. It has the same effect as exiting to the operating system and the re-entering JAM.

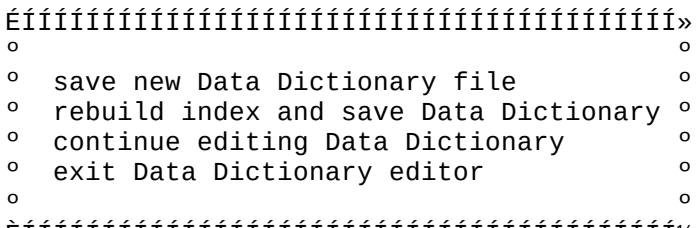


Figure 32: Data Dictionary Editor Exit Menu

| | |
|---|---|
| continue | This selection returns you to the data dictionary editor, without saving it or rebuilding the index. |
| exit | This function exits the data dictionary editor, without saving changes to disk. |

## 8.3 Data Dictionary Editor Functions

The function keys PF2 through PF10 are active in the data dictionary editor; what each one does is explained below. Most keys operate on the item at which the cursor is positioned.

### 8.3.1 Adding Entries (PF2)

To add an entry, position the cursor to the item before which the new item is to be added, and press the PF2 key. One or more lines will open up for entry of new items. To add an item to the end of the data dictionary, press PF2 with the cursor at the end (the line containing EOF in the name field) of the data dictionary.

The cursor will then be restricted to the line where the new item is to be added. Enter the desired value in each field, and press TRANSMIT to add the entry to the data dictionary. To quit without creating a new entry, press EXIT. If you press RETURN instead of TRANSMIT, the current item will be added and another line will immediately open up below it.

### 8.3.2 Modifying Entries (PF3)

The displayed text in the data dictionary is normally protected, to prevent inadvertent changes. Existing items may be modified by positioning the cursor to the desired item and pressing the PF3 function key. You may change any parameter, including the name, by typing over it. When you hit TRANSMIT or RETURN, the update is applied to the data dictionary; you may cancel it by pressing EXIT. RETURN also positions you to the next item, which you may then update.

### 8.3.3 Modifying Field Characteristics (PF4)

While either an add or a modify is in progress, you can examine and change more characteristics of a data dictionary entry by pressing the PF4 key. The PF4 key alone will implicitly begin a modify. It presents you with the screen editor's field definition menu; refer to Section 6 for instructions.

Defining field characteristics in the data dictionary is helpful in treating fields consistently across an application system. Most field characteristics have no effect on local data block processing, but several do: date, time, currency format, and field size are the most commonly used.

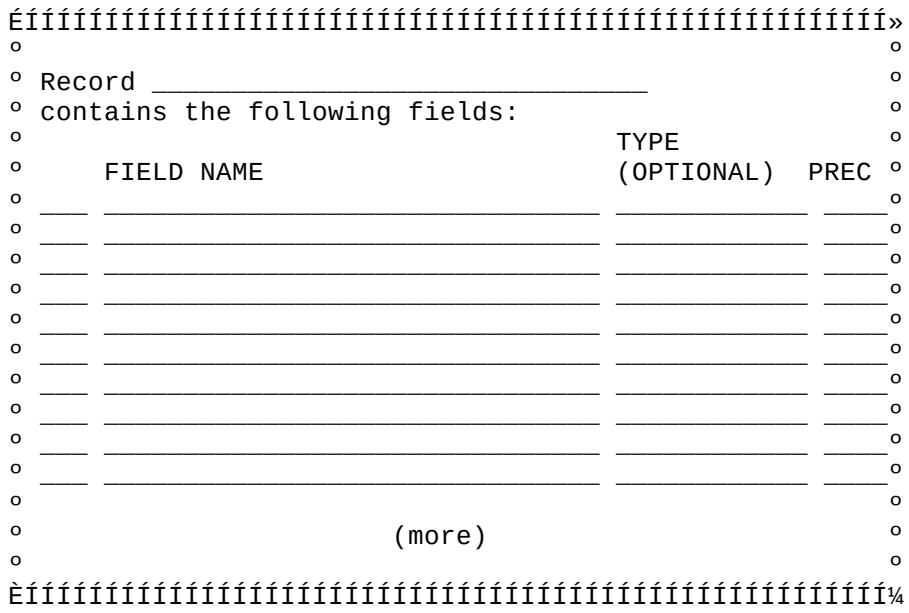


Figure 33: Data Dictionary Record Window

### 8.3.4 Creating Data Dictionary Records

If, when creating or modifying an entry, you type an asterisk * in the scope field, the entry becomes a record rather than a field. Instead of normal field characteristics, it has a list of component fields, which you enter in the window shown in Figure 33. You bring up this window by pressing PF4 with the cursor in a field having scope *, or by retyping the asterisk.

For each component field in the record, you enter a name, and an optional data type and precision. The name may or may not exist as a regular data dictionary entry; it may not be the name of another record, as sub-records are not currently supported. If you give no value for the type and precision, the values belonging to the named data dictionary entry will be used. The precision applies only to fields of type float or double.

### 8.3.5 Deleting and Undeleting Entries (PF5 and PF6)

To delete an entry, position the cursor to the desired item and press the PF5 key. The element will immediately be removed from the data dictionary. An undelete key is provided, to recover the item most recently deleted. When you hit PF6, a line opens above the cursor, and the most recently deleted item appears there.

PF5 and PF6 may be used in combination to move entries within the data dictionary, by deleting the entry from one position, then undeleting it in a new position. An entry may be undeleted only once, since entries in the data dictionary must have unique names. If PF6 is pressed twice in succession, an error message will be displayed the second time.

### 8.3.6 Searching (PF7 and PF8)

The PF7 key prompts for a text string, using the window shown in Figure 34, then searches for it through either the item names or their comments. The search begins at the item under the cursor, proceeds to the end, then wraps around to the beginning. The text string may contain, in addition to ordinary text, certain special characters:

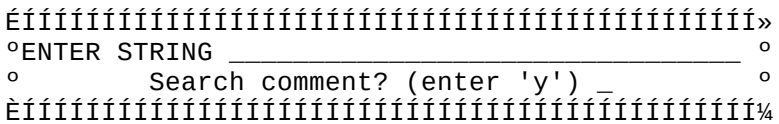


Figure 34: Data Dictionary Search Window

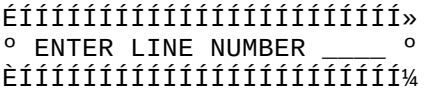


Figure 35: Data Dictionary Goto Window

- To search for a name that begins with a known sequence of characters, enter those characters preceded by a caret ("^").
- The query (?) is a wild-card, matching any single character.
- The asterisk (*) is a wild-card, matching any string of characters.

These wild-card characters are not to be confused with the special characters used in the regular expressions JAM uses to validate data entered into fields. Here are some examples of search strings:

1. The search string ^abc finds the first element after the current cursor position that begins with abc. If the caret were missing from the search string, the first element that contained the character sequence abc would be found.
2. To find an element whose name ends with xyz but whose beginning characters are unknown, enter xyz as the search string. This string will also find names with xyz in the middle.
3. To find an element whose first character is a, whose second character is b or c, whose third character is d, and have a z somewhere following the d, enter the string ^a?d*z. Of course this string would also match elements whose second letter is other than b or c.

### 8.3.7 Searching Comments (PF7, PF8)

To search for a comment instead of an entry name, press PF7 and enter the search string exactly as for a name search; then tab to the next field and enter y. A search on either names or comments may be repeated, starting at the current cursor position by pressing the PF8 key. Searches always begin at the cursor position and go to the end; they then wrap around to the beginning of the data dictionary.

### 8.3.8 Going to a Specified Line in the Data Dictionary (PF9)

To go to a specific line in the data dictionary, press the PF9 key. You will be prompted for the desired line number, with the window shown in Figure 35. Entering 0 or 1 will position you to the first line of the data dictionary; entering a number greater than the number of items in the data dictionary will position you to the EOF line.

### 8.3.9 Data Dictionary Default Settings

The PF10 key will bring up the window of Figure 36. It is similar in appearance and operation to the screen editor's field summary window (Section 6.10), but its function is to define default characteristics for newly created data dictionary entries. Here is a summary of the fields; unless otherwise mentioned,

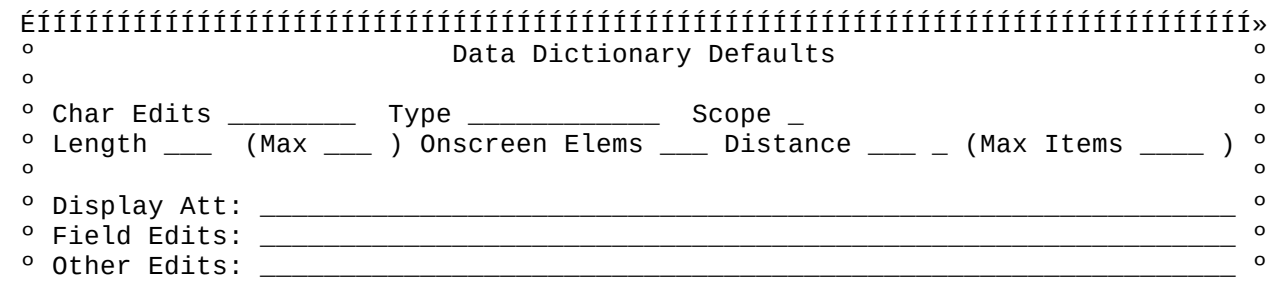


Figure 36: Data Dictionary Defaults

you just type in the appropriate value. Hitting TRANSMIT saves the changes you have made, while hitting EXIT in the window discards them.

| | |
|---|---|
| Char edits | This is an entry-protected, circularly scrolling field (do not hit RETURN to get to the Length field!), which defines the item's character edit. Use the up and down arrows to get to the one you want, then tab out. |
| Type | Another circular scroll, listing the possible data types. |
| Scope | Enter a number from 1 to 9, or * for a default scope of record. |
| Length | The onscreen length of the data item. |
| (Max) | If greater than Length, the offscreen or shifting length of the item. |
| Onscreen Elems | The number of visible array elements. |
| Distance | The number of lines or columns between elements, as appropriate. |
| Max Items | If greater than Onscreen Elems, the number of scrolling items in a field or array. |

You can set a default for any other field characteristic by pressing PF4 within this window and selecting from the usual menus.

## 8.4 Initializing the LDB

The values of data dictionary items are initialized in the local data block at the start of each JAM session. The initialization files consist of a pair of entries for each data dictionary element to be initialized. The first entry in each pair is the element name; the second is its value. For example, to initialize the data dictionary item co_name to the value JYACC, the file should contain the pair:

```
"co_name" "JYACC"
```

Items with multiple occurrences can be initialized by subscripting the name, as in the example below. Both the name and the value must be enclosed within quotation marks. All white space and line separators outside quotes in the initialization files are ignored. The following two sequences are equivalent:

1.
```
"co_name""JYACC""co_city[1]""New York""co_city[2]""Paramus"
```

2.
```
"co_name"      "JYACC"
"co_city[1]"   "New York"
"co_city[2]"   "Paramus"
```

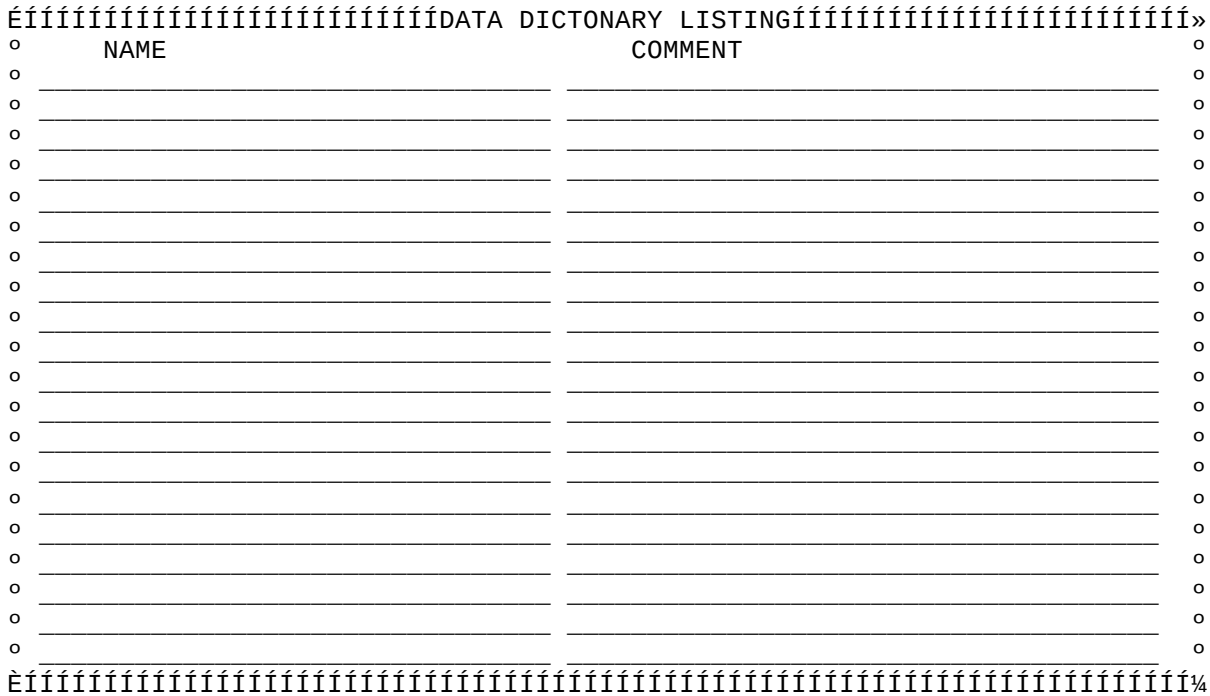


Figure 37: Screen Editor Search Window

For readability, we suggest that each pair be placed on a separate line. Data dictionary initialization files can be created and updated with any text editor or word processor.

JAM defines several default initialization file names; they are const.ini, global.ini, tran.ini, and local.ini. You may add to or change these names using the SMININAMES configuration variable.

## 9 Data Dictionary Operations in the Screen Editor

To speed the process of creating screens with shared fields, you can define the common fields, and enter them into the data dictionary with the data dictionary editor (Section 8). Then, use the data dictionary search feature of the screen editor to place common data items into screens as named fields.

You can search the data dictionary in the screen editor, in order to create a field that matches the specifications of an element of the data dictionary, or to compare an existing field with a data dictionary element.

### 9.1 Data Dictionary Search in the Screen Editor

To create a new field, place the cursor where the field should start, then press SPF6. To compare an existing field to a data dictionary element, place the cursor in the field before pressing SPF6.

When the "data dictionary" key (SPF6) is pressed, a large window opens (Figure 37), and a number of data dictionary entries are displayed. You may search by scrolling up and down through the display, or with the PF7 key (see Section 8.3.6). Make a selection by positioning the cursor to the desired data dictionary element, and pressing TRANSMIT.

If the cursor is in a field at the time the data dictionary search key is pressed, the window is opened and the data dictionary shows one of the following three ranges of data dictionary elements:

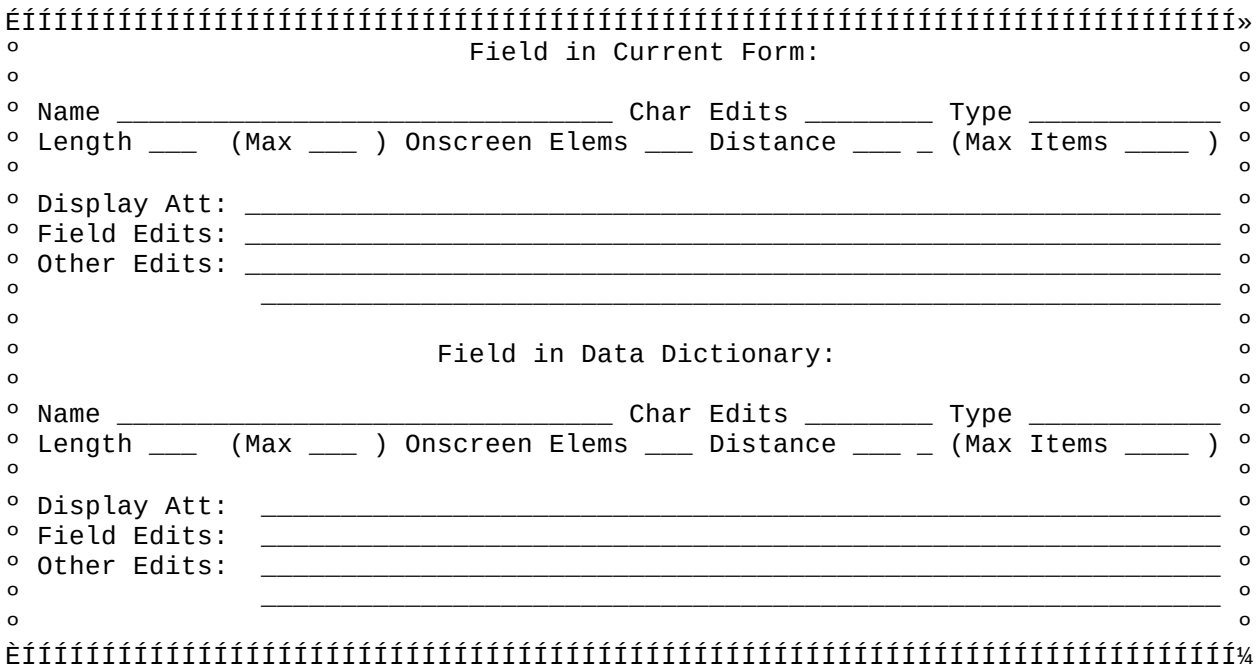


Figure 38: Screen Editor Comparison Window

1. If the cursor is in a named field and the name is in the data dictionary, the display shows the data dictionary element with that name under the cursor.
2. If the field is not named, or the name is not in the data dictionary:

    a. If the data dictionary has been used since the start of the authoring session, the same elements displayed at the time the window was last closed are displayed.
    b. If this is the first use of the data dictionary function, the first several elements are displayed.

### 9.1.1 Comparing a Field to a Data Dictionary Entry

When you invoke data dictionary search with the cursor in a field and select an element, the data dictionary window closes and a new window (Figure 38) appears. It displays characteristics of the current field above, and the characteristics of the selected data dictionary item at the bottom. You can now alter any of the field's characteristics, with the data dictionary description and help windows as a guide. You can change the field characteristics displayed in the window directly, as in the field summary window (Section 6.10). In addition, the following function keys are active:

| | |
|---|---|
| TRANSMIT | Update the screen field with whatever modifications have been made in the comparison window. |
| EXIT | Close the comparison window, leaving the screen field unaltered. |
| PF4 | Modify the screen field using the regular PF4 menus, as in Section 6. When you come out of the field menu, your changes will be reflected in the comparison window. |
| PF5 | Change the characteristic under the cursor to whatever appears in the data dictionary description. |
| PF6 | Restore the characteristic under the cursor to whatever it was in the original screen field. This undoes any changes made by direct entry, PF5, or PF4. |

### 9.1.2 Making a Field from a Data Dictionary Entry

When you invoke data dictionary search with the cursor not in a field, and select an element, the search window closes immediately and a field with the characteristics of the selected item is created at the cursor position. If the selected item is an array or scroll, the resulting field will be too. If the new field or array would overlap an existing field, it is not created; an error message appears instead.

### 9.1.3 Making a Data Dictionary Entry from a Field

To create a data dictionary entry from an existing field, place the cursor in the field from which the entry is to be created, and hit the SPF4 function key. If the field is named, and the name is not already in the data dictionary, a new entry will be added to the end of the data dictionary, and a confirmatory message will be displayed. An error will result if the cursor is not in a field, if it is in an unnamed field, or if it is in a field whose name already occurs in the data dictionary.

Note that data dictionary entries thus created will not appear in the local data block until the LDB index is rebuilt, using the rebuild index option on the data dictionary editor's exit window.

# 10 Special-Purpose Screens

The JAM library includes routines for menu processing and automatic display of help screens. The following sections explain how to create forms for use with those functions. The functions themselves are described in the JAM Programmer's Guide.

## 10.1 Menus

JAM supports two distinct kinds of menu processing:

1. under explicit application control, through the library functions menu_proc and choice, and
2. under JAM control, using control fields.

(Help screens with menus use the first kind; see Section 10.6.)

### 10.1.1 Menus for Use With menu_proc or choice

The library functions menu_proc and choice allow for easy selection of menu items. The current (or tentative) selection is always displayed in reverse video. To create a screen that can make use of this feature:

1. Create a selection field for each menu entry. Each field must be long enough to contain all the text to be displayed. You can make every field equal in length to the longest text to be entered in any of them, as on jxform's field characteristics menu (Figures 39 and 40), or you can make them different lengths, as on jxform's exit options menu (see Figure 2). The area defined as a field, not just the text, will be displayed in reverse video when the cursor is at an item.
2. Set the fields' display attributes as desired, but do not include reverse video.
3. Give each menu field a menu field edit (Section 6.6).
4. Enter the menu item text in the menu fields. Each item should start with a distinct character (Figure 40). The menu shown is used by jxform for the selection of field characteristics (Figure 9); on the actual menu the fields are not underlined.

If possible, initial letters should be all upper case or all lower case. This permits the menu processing function to ignore the case of letters entered from

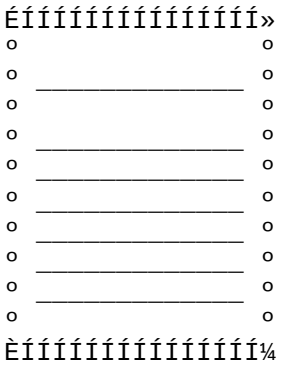

Figure 39: Fields Defined for Field Characteristics Menu

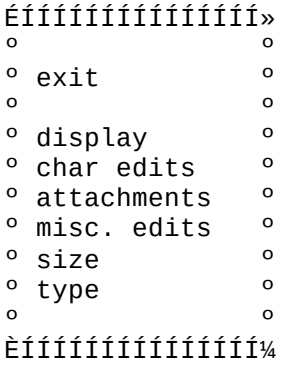


Figure 40: Initial Data Entered in Field Characteristics Menu

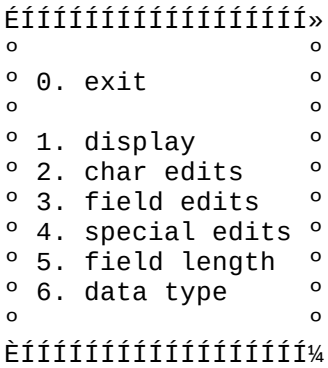


Figure 41: Menu Using Digits for Item Tags

the keyboard, treating a and A as equivalent. If several items would naturally start with the same letter, the item names can be preceded by digits (Figure 41), or by letters that are not part of the names themselves.

Alternatively, upper and lower case initial letters can be mixed, allowing a distinction between a and A, but requiring the operator to use the shift key for upper case.

### 10.1.2 Menus for Use with JAM

Menu screens may contain display data and menu selection fields; they will not usually contain data entry fields, although JAM now supports dual-use screens. JAM identifies a screen as a menu screen by the presence of a field named

jam_menu on the screen. It is typically an array containing the menu descriptions, but this is not a requirement.

The menu selection fields consist of one pair of fields for each menu item. Each pair consists of an unprotected field referred to as a selection field, which contains the text of the menu selection, and a control field referred to as a processing field, which contains the control link associated with the menu item. The processing field of the pair must have a field number one greater than the selection field. The easiest way to do that is to place the processing field on the same line as the selection field and immediately to the right. (The create menu function of jxform creates an array named jam_menu, and a parallel array for the processing fields.)

A user running JAM selects items from menus by typing the first character(s) of the value of the desired menu selection field, or by moving the cursor to the desired selection and pressing the TRANSMIT key. If the first method is chosen, you must type enough characters to differentiate between the selections; as soon as this occurs, the selection is made. It is therefore convenient to have the initial character or characters of menu selection fields be unique. This may be accomplished by creative naming, or by using index letters or numbers as the first character. Examples of three different types are shown in figures 42, 43, and 44.

While most menus consist of a vertical list of options (Figure 42), JAM permits any organization for menus, including horizontal (Figure 43), or more exotic menus, such as Figure 44, in which the menu selections are the headings of introductory paragraphs. These other menus work because JAM considers all menu fields that don't begin with a blank on a jam_menu screen to be menu selection fields.

It is sometimes desirable to create menus whose contents change as circumstances dictate. For example, in a system where security is an issue, users should be shown menus that include only the functions that they have the authority to execute. Similarly, in a sales system, you might want to customize menus to include those items that a client is most likely to purchase based on what you know about the client (such as net worth, marital status, etc.). To create such "dynamic menus" is a simple matter of combining the menu and data dictionary capabilities of JAM as follows:

Create the menu selection and menu processing fields as described above, but give each field a unique name that corresponds to a data dictionary element. The processing that decides which items to place on the menu simply initializes the appropriate data dictionary elements before displaying the menu. When the menu is displayed, the selection and processing fields will get their values from the local data block (as all fields do), and you have a dynamic menu. You can also use the screen entry function (Section 4.7) to populate a menu at run-time.

## 10.2 Creating Display-only Screens

Display screens have no unprotected fields. They are typically used to provide product information (such as an on-line catalog) or instructions to users. They may have control fields initialized to an action to take when TRANSMIT or other function keys are pressed (either display a screen, display a window, invoke a function, or execute a program). In absence of control fields, the only action you may take is to press EXIT to return to the previous screen.

Display text, border, background color, and display characteristics may be created freely, using jxform. You may display variable data on a display-only screen by defining protected fields that have the same names as items in the data dictionary.

To create a jam_xmit control string, enter jxform and press the SPF1 key, move to the field marked XMIT, and enter the desired action for the TRANSMIT key. The

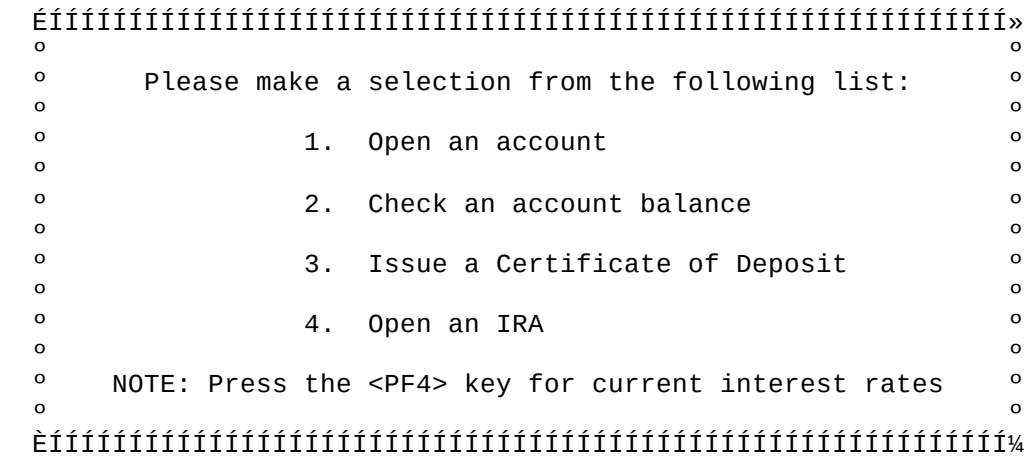


Figure 42: Vertical Menu

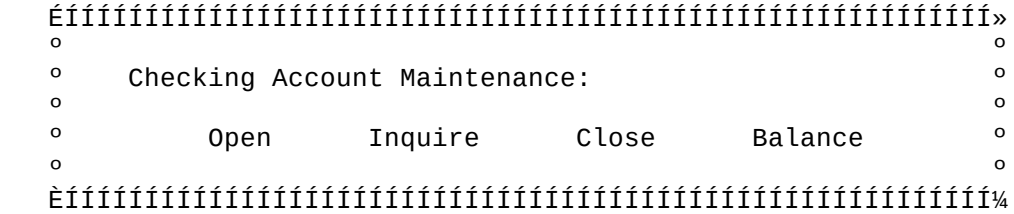


Figure 43: Horizontal Menu Window

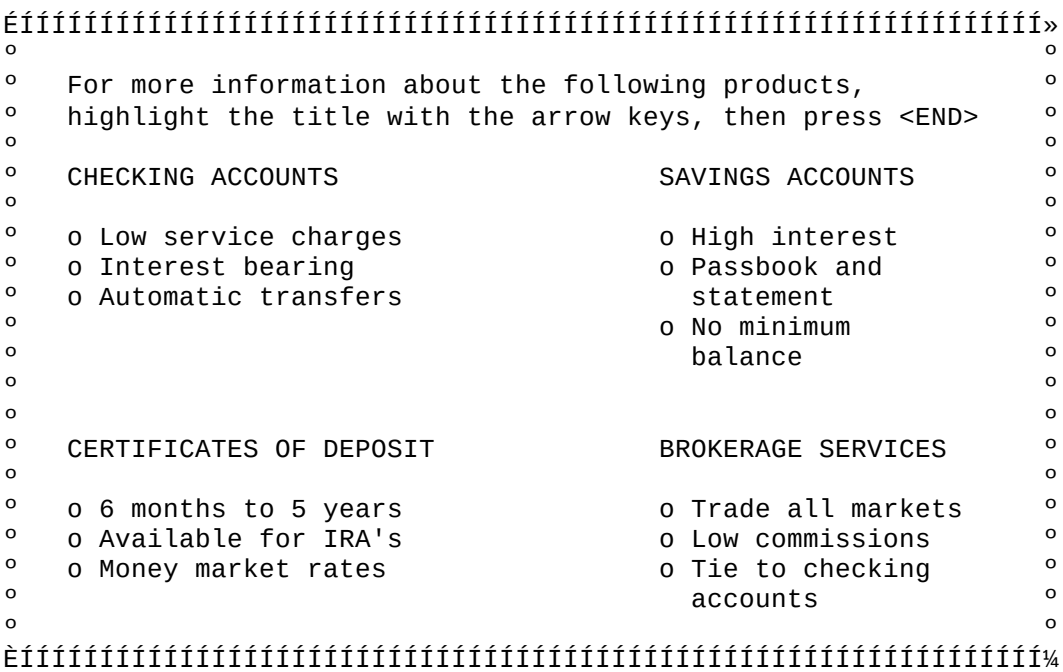


Figure 44: Topical Menu

SPF1 key will similarly create control strings associated with EXIT and the user-defined function keys.

Display-only screens may be invoked as windows; in that case, they exhibit certain characteristics that will be discussed in section 10.4.

## 10.3 Creating Data Entry Screens

The only difference between display screens and data entry screens, from JAM's point of view, is that data entry screens have at least one unprotected field. They are typically used to implement transactions through the use of procedures attached to fields or function keys, or to simulate them in a prototyping environment, or just to load data into the LDB.

You may automatically display variable data in data entry screens by defining fields that have the same names as items in the data dictionary. Similarly, you may pass data to other screens by assigning field names on the different screens to the same data dictionary item.

They should have control strings that specify the action to take when TRANSMIT or other function keys are pressed (either display a screen, display a window, invoke a function, or execute a program). In the absence of these control fields, the only action you may take is to press EXIT to return to the previous screen. To create a jam_xmit control string, enter jxform and press the SPF1 key, move to the field marked XMIT, and enter the desired action for the TRANSMIT key. The SPF1 key can similarly create control fields associated with EXIT and the user-defined function keys.

Data entry screens may be invoked as windows. This will be discussed in the section 10.4.

To implement transaction applications, you may attach functions to fields as described in section 6.8.1. You can be sure that the attached functions will be executed because TRANSMIT key processing invokes screen validation (s_val) before invoking the jam_xmit control link.

## 10.4 Creating JAM Windows

JAM windows may be any type of screen: display, data entry, or menu. There is very little difference to JAM between windows and non-windows; in fact, the same screen may be used both as a window and as a base form. Whether a screen will replace or overlay the existing screen is determined when the window is displayed, rather than when it is created.

A window may be full size; if it is, it will completely cover the previous screen. If a window is smaller than the underlying screen, all of the underlying screen not covered by the window will be visible, but you will not be able to enter data into the underlying screen. The characteristics of a window are:

1. If a screen displayed as a window has a control field that invokes a screen without the window attribute ("&"), JAM will close the window (and all underlying windows) before displaying the selected screen.
2. If a screen displayed as a window has a control field that invokes a screen with the window attribute ("&"), the new screen will be displayed as a window overlaying part or all of the existing window or windows.
3. A screen displayed as a window with unprotected fields will update the LDB as any other screen does.
4. A screen displayed as a window with unprotected fields will perform all edits and validations, including attached functions, as any other screen does.
5. If a window or form has an AUTO control string, the screen specified in the AUTO string will be displayed immediately, without pausing for operator action.

Most of these characteristics are clear. Essentially, windows are transient, and leave no record of having been displayed. They are thus especially useful for providing extra help to novice users, and for nesting menu selections.

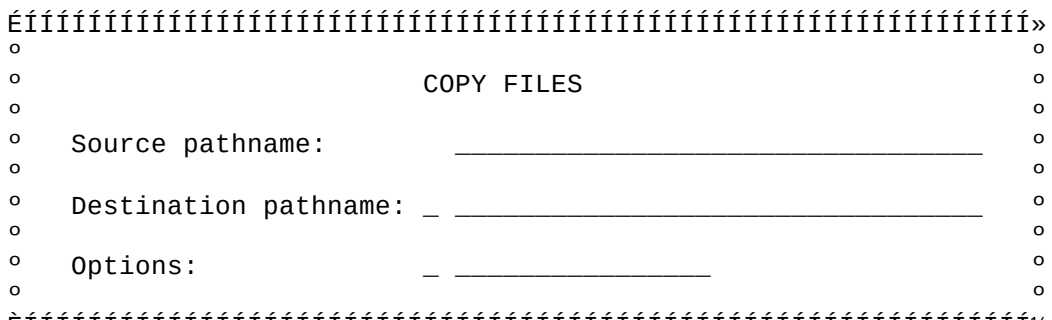


Figure 45: Argument Window

It is sometimes desirable to display several blocks of text at the same time, in different places on the screen. This may be done by creating a set of nested windows, and specifying the desired placement of each window in the AUTO field of the previous window.

## 10.5 Creating Windows for Command Line Arguments

Argument windows are used by programs and functions called from control fields. A program is called by placing the command line in the control field, preceded by an exclamation point (!). A function is called in the same way, except that it is preceded by the caret (^) symbol rather than the exclamation point. In either case, run-time arguments may be included in the command line by using an argument window.

An argument window name is placed in the command line preceded by a percent (%) symbol. More than one argument window may be used if required. At run-time, all of the fields in the argument window are concatenated and inserted into the command line, replacing the percent symbol, the argument window name, and the blanks (if any) following the argument window name.

In general, trailing blanks in each field are removed before concatenation, so several fields may be used to make up a single argument; the one exception is that empty fields are inserted as a single blank character. This permits the use of protected blank fields to separate arguments from each other.

An example of a window for the MS-DOS COPY command is shown in Figure 45. The single underscores represent protected, non-display fields that cause single blanks to be inserted into the control field.

In this example, the protected fields before each of the unprotected fields insert one space to separate arguments. If desired, default values may be inserted into fields at the time the screens are created. Fields may also be mapped to and from the LDB through the data dictionary. To use this feature, name the field with a data dictionary element name. One use of the data dictionary with argument screens is to permit the screen to "remember" the last argument values used. If each of the fields is given a data dictionary transaction variable name, any value entered into the field will be displayed the next time the argument window is displayed.

## 10.6 Creating Help Screens

JAM's help screen feature associates a help window with a field or an entire form; each help screen is itself created with the JAM authoring utility. Help screens can function in several different ways.

The simplest help screens, like Figure 46, are display-only; they contain no fields, just display data. When the HELP key is struck with the cursor in the

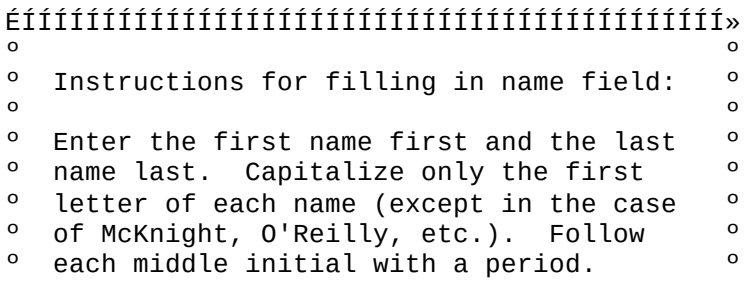


Figure 46: Help Screen with Display Data Only

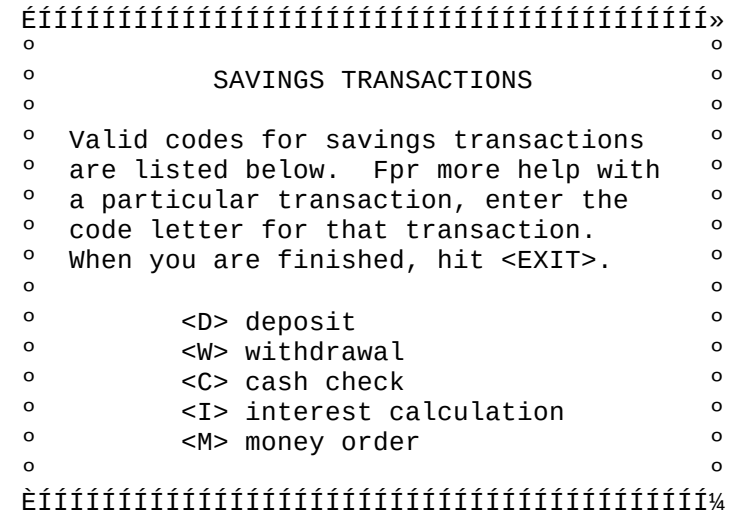


Figure 47: Help Screen Menu

associated field, the help screen is displayed as a window. The help window is closed at the next keystroke.

### 10.6.1 Help Screens Containing Menus

Help screens can contain menus which call up additional help screens. To create a help screen with a menu:

1. Create a screen with menu fields, as described in Section 10.1.1.
2. For each menu field, use the help screen option (Section 6.7.3) to specify an associated lower level help screen.

When the operator makes a selection from the menu, the associated lower level help screen will appear. The operator exits from the help screen menu by hitting the EXIT key.

### 10.6.2 Help Screens with Data Entry Fields

In some cases, it may be desirable for the operator to be able to enter data into a form while a help screen is displayed. This can be done if the help screen is created with a field for data entry.

When the help screen is displayed, the contents of the associated field are copied into the field on the help screen. You may then enter data in the help screen field. When you hit TRANSMIT, the data in the help screen field are

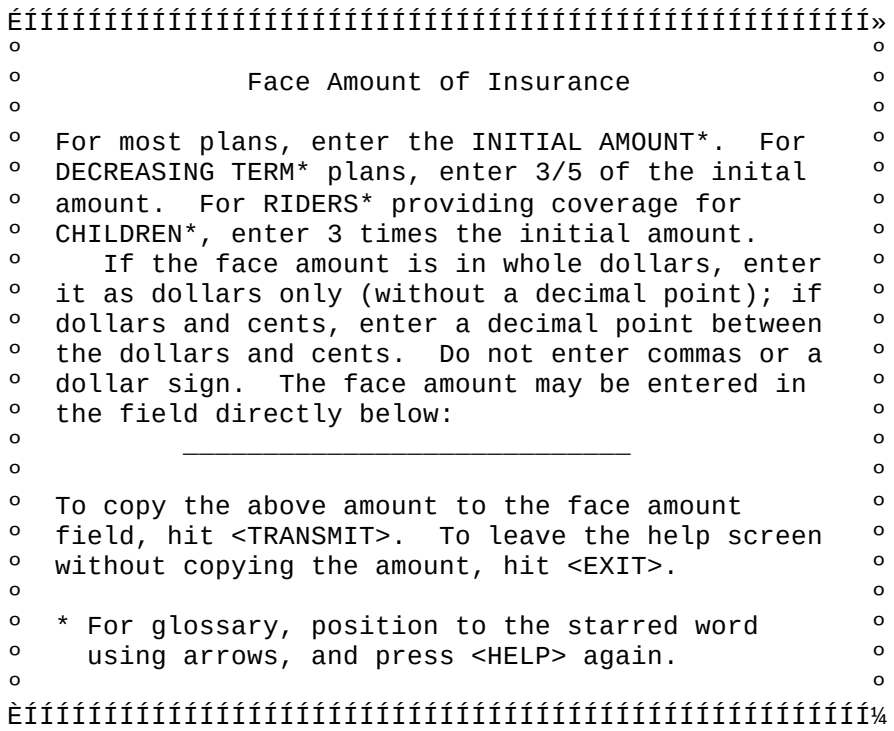


Figure 48: Help Screen with Data Entry Field and Protected Fields

copied back into the associated field, and the help window closes. If you hit EXIT instead, the field is not copied.

The help function will provide automatically for data entry whenever the help screen contains exactly one unprotected field that has no associated procedure name. The data entry field is normally defined to be as long as the field(s) the help screen is associated with. If the data entry field is too short for the data, the help function will automatically make the field shiftable, with a maximum length equal to the associated field length. The help function will also copy from the associated field:

1. the character edits (Section 6.5)
2. the following field edits (Section 6.6):

   - right justified
   - upper or lower case
   - data required
   - must fill

3. the following special edits:

   - check digit (Section 6.8.3)
   - currency format (Section 6.8.4)
   - range (Section 6.8.5)

The help function will not process both data entry and menu fields on the same help screen. However, it is possible to provide additional help screens for a

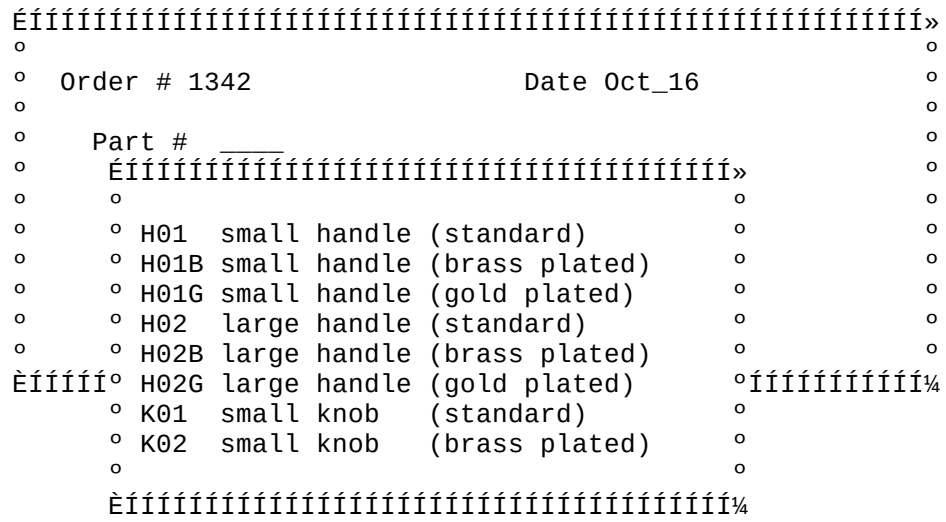


Figure 49: Item Selection Screen

data entry help screen, by using protected fields, as described in the next section; see Figure 48.

### 10.6.3 Help Sub-screens Using Protected Fields

An alternative to menu processing uses fields containing explanatory text, which are protected from data entry but not from tabbing into. When the cursor is positioned within such a keyword field and the HELP key is hit, the associated lower level help screen is displayed. To create a help screen using this feature:

1. Create a help screen with all text entered as display data.
2. Replace the key words by underscores, and hit TRANSMIT to compile the underscores into fields.
3. Re-enter the key words in the fields. Make the key words easily distinguishable from the rest of the text by use of distinctive display attributes, or by some other means such as capitalization.
4. Make each key word field protected from data entry, but not from tabbing (Section 6.6), and give it a help screen (Section 6.7.3).

The help function will not process both menu fields and protected keyword fields on the same help screen.

## 10.7 Item Selection and Table Lookup Screens

An item selection screen is similar to a help screen attached to a field. The contents of each field on the item selection screen correspond to a valid entry in the associated field; you choose an item by a process similar to menu selection. Because a list of valid entries for a given field can run to more items than could be displayed on the screen at one time, item selection screens may need to use scrollable arrays.

If there are many items, several could easily start with the same character, so the "initial character" selection feature is modified. Entering the initial character of one of the items simply positions the cursor to the next item on the list that starts with that character. (Note that, if the fields are defined as a scrollable array with offscreen data, this next item might not be visible at the time the character is keyed in.) The TRANSMIT key must be hit to complete the selection.

When you hit the TRANSMIT key, the item selection window is closed, and the selected item is copied into the associated field. If the EXIT key is hit, the window is closed but nothing is copied.

An item selection screen should consist of several fields, a simple array, or a scrollable array; all must be menu fields. If a scrollable array is used, the actual number of items entered should be the same as the maximum number. The lengths of fields on the item selection screen need not be the same as the length of the associated field. If the item selection fields are longer, as in Figure 49, only the first part of the field is copied, but the rest of the field can contain other information that is helpful to the operator.

An item selection screen is frequently also used as a table lookup screen, to ensure that a field's contents actually match one of the items; or it may be used for lookup alone. Lookup screens are created in exactly the same way as item selection screens; however, they are never displayed at run-time, so no attention need be paid to the appearance of the screen.

# Index

In this Index, library functions are displayed in boldface, without the prefixes specific to the language interface. Video and setup file entries appear in ELITE CAPS, while utility programs and JPL commands are in elite lower-case. Function key names are in ROMAN CAPS.