Introduction to JAM

Contents

1 About This Manual

Congratulations on your purchase of JAM, the JYACC Application Manager. Your documentation includes the following chapters:

> The Introduction to JAM (this chapter) describes the rest of the documentation, and explains the basic concepts and mechanisms of JAM.
>
> The JAM Author's Guide describes in detail how to create the screens and links that constitute JAM prototypes.
>
> The JAM Programmer's Guide explains some of JAM's internal operation, shows how to code application routines in the language of your choice, and describes in detail the support functions supplied with JAM.
>
> The JAM JPL Programmer's Guide describes the JYACC Procedural Language, a specialized interpreted programming language.
>
> The JAM Configuration and Utilities Guide explains how to create and alter configuration files for terminals and displays, and how to use numerous utility programs.

Those who are familiar with JYACC FORMAKER may find that Section 6 of this chapter and the section on JAM control links in the JAM Author's Guide are sufficient to get them started.

Even those whose motto is "when all else fails, read the manual" may find the following nuggets useful:

- Section 6 of this chapter
- The sections on JAM control fields and data entry keys in the JAM Author's Guide
- The keyboard maps and Glossary appended to this chapter

Finally, for those who choose to read it through, this Introduction will provide a framework of understanding for designing applications with JAM: how the design process flows, what facilities JAM provides and how they work, and how to use the individual tools in the JAM package.

Words printed in italics are defined in the Glossary following this chapter.

2 Working with JAM: Two Scenarios

This section presents a quick look at two aspects of JAM: designing an application and using it.

2.1 Application Design

The first step in designing an interactive application is to break it down into screens and groups of screens. Very often, things will fall into a simple tree structure like the one in Figure 1. At other times, the structure of the application is more complicated, as in Figure 2: here two screens share the sub-screen marked Application #5, and there are links up as well as down the menu tree. It is often a good idea, if your application is at all large, to draw such a picture; it will help to visualize the way your application is organized.

With a basic idea of what you want, you are ready to start up JAM and begin creating screens. With JAM's free cursor motion and drawing features, your ideas for screen layout quickly appear; just as quickly, you can test and revise them. It is easy to create several variations on the same idea and compare them. You

Figure 1: Tree Structure of Screens

Figure 2: Complex Structures of Screens

can also add prompts and help displays, to show others (or to remind yourself) what is to happen next.

Next, you create links between screens to direct the flow of control and data in the application. You can group screens under menus, call up sub-windows, have data entered in windows appear automatically in their parent screens, and call up existing programs. In fact, you can create a prototype application that looks exactly like the real one. We call this prototype the application shell, because it shows what the application will look like on the outside.

Now the process of refining the application shell can begin. This may involve demonstrating it to a number of people; you can solicit suggestions, and make and revoke changes on the spot. Since it is just as easy to rearrange the links between screens as to alter the layout of an individual screen, you can try out different organizations. In this way, the user interface to an application can be thrashed out quickly and thoroughly before coding begins.

At this point, the application shell embodies the structure of your application. Now, you simply add processing routines to the application shell, within the control structure it provides, and your application is finished!

There is a particular class of application, called "transaction-based applications", for which JAM is particularly suited, and we sometimes refer to transactions in this manual. The term connotes a group of related data items and screens, but is difficult to define precisely. If it is unfamiliar to you, simply think of a screen or a group of related screens.

2.2 Function Keys and Application Execution

JAM makes heavy use of function keys to control the execution of applications. Function keys are special keys on a computer or terminal's keyboard, distinct from those used to enter data. Because the names and positions of keys vary from one terminal to another, JAM has a notion of logical keys: we speak of "the EXIT key" or "the TRANSMIT key", even though there is probably no key on your terminal labeled "EXIT" or "TRANSMIT". JAM has configuration files that tell it what real keys correspond to its logical keys.

At any rate, function keys tell JAM how to behave. In a menu screen, for instance, striking arrow or tab keys makes the cursor move from one menu selection to another. Pressing the Transmit key tells JAM that the item under the cursor is the right one, and it proceeds to bring up a new screen corresponding to the menu selection.

In data entry screens there are function keys for moving between fields, clearing fields, and altering data in fields, as well as the normal keys for entering data in fields. Again, the Transmit key is often used to tell JAM that the data on the screen is complete and correct. There are also help keys that cause explanatory text to be presented. Along with these and other predefined keys, JAM defines a number of function keys that an application may interpret in any way it chooses.

3 JAM Concepts

This section defines the major components of a JAM application: screens, control links, data links, and programs. Here, we simply describe what they do; in Section 4, some details of how they do it appear.

3.1 Screens

Screens, arrangements of data on the computer's display, are the basis of an application designed with JAM. There are a few different types of screens, described below, but they share the same basic components. Display data are text, graphics, and borders that do not change. They serve to identify the screen and its constituent fields, as well as to tell the user what to do. Data

entry and presentation take place in fields, the variable part of a screen. Fields are often highlighted visually in some way, such as by underlining.

With JAM, you can define many of the characteristics and actions associated with a field within the screen itself, without resort to programming. Here are some of the most frequently used:

- The field's name
- Its data type (dollar amount, character string, etc.)
- The field's display attributes (color, highlighting, etc.)
- A prompt or help screen explaining the field
- Whether data entry in the field is allowed
- Whether data in the field should be left- or right-justified
- A routine to be called when the field is tabbed through
- A calculation to compute the field's value

Items associated with a field are variously called attachments, edits, and validations. See the Glossary for an explanation of these categories.

Menus are a special kind of screen that guide users to a particular place in an application. A menu presents a list of choices. When the user picks one, another screen corresponding to the choice comes up. It may be a sub-menu with further choices, or it may be a data entry screen for some transaction.

Both data entry screens and menus may be displayed either as base forms or as windows. A base form covers the entire display; a window typically (but not always) occupies part of the display, leaving a form partially visible "beneath" it. Windows are normally subservient in some way to the forms (or other windows) they overlay, while forms are not.

3.2 Control Links

In the broadest terms, a control link associates something the user of a JAM application does with the application's response. A common example is that the user makes a menu selection and JAM responds by bringing up a new screen. More generally, JAM recognizes two types of action that trigger control links:

- Menu selections
- Function keys

and two types of response:

- Bring up a new screen
- Invoke a program or function

The first response leaves control of the application with JAM. In the second case, however, control is transferred to the application code, and JAM control links have no effect until that code returns.

Finally, there are a few specific function keys whose action links are restricted to one type. There is a GOTO key, which when struck prompts for the name of a screen and brings it up; this provides knowledgeable users with

shortcuts through a menu-based system. There is a SYSTEM key, which will prompt for and execute an operating system command (possibly an escape to the operating system's command interpreter). And there is an EXIT key, which normally causes JAM to erase the screen that is currently displayed and return to the previous one.

3.3 Data links

Data links are most easily described by explaining how they work, which is done in Section 4.3. Here we shall simply say that important data items can be given names and shared among various JAM screens and transactions, thereby linking the data belonging to those transactions. This makes it possible, among other things, for an item entered in a window to be displayed automatically in the window's parent form, or even in an unrelated form; and for application code to refer to data entered in screens that are no longer displayed.

The JAM Database Interface, or JAM/DBi, is an optional subsystem provided with JAM. It extends data links to a relational database manager.

3.4 Programs

JAM supplies much of the logic for controlling the flow of data and execution in an application. Nevertheless there remains, of course, a need for application-specific code. JAM provides several types of hooks on which application code can be hung. Among these are the control links just discussed; attached function field edits, which specify routines to be called when the cursor enters or leaves a field; and hooks where you can insert special processing into a number of generic operations, such as keyboard input. These and other hooks are fully discussed in the Programmer's Guide.

As for the application code itself, it may either be part of the currently running JAM application, or it may be a separate program (possibly one written well before the JAM application). The main difference from JAM's point of view is that code within an application can share control and data links, but code in separate programs cannot.

In addition to standard programming languages, JAM contains a specialized programming language called JPL, for JYACC Procedural Language. Since JPL is interpreted, it speeds up prototyping by eliminating the edit-compile-link cycle; and it contains special features for manipulating JAM data items.

4 JAM Mechanisms

In this section we explain a bit about how JAM works, with a view toward giving application designers a firmer conception of how to use its features.

4.1 Screens

Screens are stored as files and are referenced by name, although they may reside either on disk or in memory. All information about a screen's display data, fields, and edits are stored in this file. Also in the screen file are the control and data links active when that screen is displayed; they reside and are changed with the screen, not in some central location. When JAM displays a menu, form, or window, it reads the file with the corresponding name and builds structures in memory to represent the fields and links.

Menu screens are distinguished from data entry screens by the presence of a field named jam_menu.

4.2 Control links

Control links are implemented through control strings stored in the screen. On a menu, the control strings appear in special fields following the menu selections; the control strings associated with function keys are stored

offscreen. In either case, the first character of the string identifies the type of action, as follows:

```
^         call an application routine
!         execute another program
&         bring up a window
anything else
     bring up a base form
```

Following that is the name of the screen, program, or application routine. (For a base form, the first character is included in the name.)

JAM traces an application's flow of control. Every time a JAM application brings up a new screen, JAM stores the name of the screen. This makes it possible to return up the path to a previously visited screen by using the EXIT key.

4.3 Data links

Each JAM application has a data dictionary file which holds a list of named data items, with all their characteristics. These data items are linked to screens by name; that is, a field whose name is the same as that of a data dictionary entry will share its value with that entry. When the screen is brought up, the field will be initialized to the shared value.

If new information is subsequently entered into such a field, the resulting value will be stored under the shared name. It will be stored not in the data dictionary itself, but in a memory-resident structure called the local data block or LDB, which provides a dynamic copy of the data dictionary. When you move between screens, the LDB preserves the newly entered values of all data items listed in the data dictionary.

The exact behavior of a named data item is governed by the scope of the data item, one of its permanent characteristics stored in the data dictionary. Items having the same scope can be erased and reinitialized as a group.

The JAM Database Interface uses a query and data management language, such as SQL, to link database fields to the LDB and screen. Here the database field or column names correspond to data dictionary names.

4.4 Programs

We defer detailed discussion of JAM programs to the Programmer's Guide, and make only a couple of general points. Programs, like other items in JAM, are referred to by name, and the names and addresses of application routines referenced by control links must be installed in a special list called the function list. Programs can access named data and screen items through function calls supplied in the JAM library.

JPL procedures, on the other hand, can access named data and screen items directly, treating them like global variables. If you have JAM/DBi, your SQL or other database language statements are incorporated into JPL procedures. Refer to the JPL Programmer's Guide, and the JAM Database Interface Guide.

5 JAM Tools

This section summarizes the authoring and programming tools supplied with JAM, and how they support the features described in Sections 3 and 4.

JAM includes two distinct environments, called the authoring environment and the application environment. All prior discussion has referred to the application environment. The authoring environment is identical, with two significant additions: someone running in the authoring environment can call up the JAM screen editor and the data dictionary editor.

The authoring utility, JXFORM, is a tool for defining screens and control links; it includes a screen editor, which you use to create display data, fields and their edits, and the control fields that specify control links. JXFORM also contains a syntax-checking data dictionary editor. There are special functions in the screen editor for automatically creating fields linked by name to data dictionary items. Both the screen editor and data dictionary editor are fully documented in the JAM Author's Guide.

The JAM library is an extensive collection of functions for reading and writing data contained in forms and in the Local Data Block. There are functions for displaying screens and getting keyboard input, for applications where standard JAM control is not appropriate. Much more information, and a description of each entry point, are available in the Programmer's Guide.

The following utility programs, and more, are described in the Configuration Guide; the introduction to that chapter contains a summary of all utilities.

All configuration files are supplied as text files you can modify with an editor; each also has a binary format that JAM uses at run-time. Msg2bin, key2bin, vid2bin, and var2bin convert ASCII message, key, video, and setup files (respectively) to binary. Modkey is a specialized editor for keyboard configuration files, and term2vid can create a primitive video file from a terminfo or termcap database. There are also several utilities for managing the configuration of JAM applications. One called jammap creates a cross-reference listing of all the JAM screens in a application, with their control strings. Another, bin2c, converts binary configuration files to C language source files; f2struct and dd2struct create programming language data objects from screens and data dictionary records. Dd2asc and ddmerge can be used to list, edit, transport and combine data dictionary files. Finally, f2r4 and dd2r4 convert JAM release 3 screens and data dictionaries to Release 4.0 format.

6 What do I do now?

If JAM has not yet been installed on your computer, please refer to the Installation Notes for guidance, and return to this section when you are ready to try out the newly installed software.

Sit down at your computer or terminal and invoke the JAM authoring utility by typing

```
jxform
```

at the system prompt. If all is well, the screen will clear, and you may see some messages at the top referring to missing data dictionary files; you are ready to begin. Turn to the JAM Author's Guide and follow the instructions there. If, on the other hand, the computer prints only a single error message, there are some things to set up in your environment.

6.1 Setting Up Your Environment

Execution Path

If the message you see resembles one of the following:

```
Bad command or filename(MSDOS)
jxform not found       (UNIX/XENIX)
Not found. JXFORM (std$cp)
                       (PRIMOS)
```

then the directory where JAM resides is not in your execution path, and you must add it. Here are some examples of how to do this:

```
PATH=$PATH:/usr/jam; export PATH  (UNIX, Shell)
set path=($path /usr/jam)       (UNIX, C shell)
```

```
        PATH c:\bin;c:\usr\bin;c:\jam          (MSDOS)
```

MS-DOS note: You must type in the old value of the path by hand; it can be obtained by typing 'path' with no argument.

PRIMOS note: There is no environment; you must either install the screen editor in CMDNC0, or define an abbreviation to run it from the installation directory.

JAM Configuration Variables

If the message you see is

```
     SMMSGS not found
```

then you need to define JAM configuration variables. (If it is SMVIDEO not found or SMKEY not found, you probably just need to set SMTERM; but bear with us for a moment.) Here are the variables and their meanings:

```
   SMMSGS         pathname of a file containing error message text SMKEY
                  pathname of a keyboard configuration file SMVIDEO
                  pathname of a display configuration file SMVARS
                  pathname of an abbreviation file containing all three
                  SMTERM
                  abbreviates your terminal's make and model
```

The first three are the ones you really need. They tell JAM where to find its configuration files: one with error message text, another that maps your terminal's keys to JAM's logical keys, and a third that tells JAM how to control your terminal's display.

These files are normally installed in a subdirectory named config of the directory where JAM was installed. The default message file is called msgfile.bin. The video and keyboard files come in pairs; their names consist of a prefix corresponding to the terminal type followed by vid.bin and keys.bin respectively, as in

```
     vt100keys.bin       vt100vid.bin
```

for the DEC VT-100. The config subdirectory contains a file named smvars, with pathnames of all the configuration files qualified by the terminals to which they belong.

Anyway, once you've found the files you need, the most straightforward thing to do is to assign their full pathnames to the SMMSGS, SMKEY, and SMVIDEO variables. An alternative is to set the SMVARS variable to the pathname of the smvars.bin file in the JAM config directory, and your SMTERM to your terminal abbreviation. Then, JAM will find the files flagged with your terminal type in the smvars file.

PRIMOS note: The configuration files are in a top-level directory named FORMAKER*, and there is no environment; JAM will prompt for your terminal type.

MS-DOS/XENIX note: For consoles, the key file is IBMkeys.bin; the video file is bwvid.bin for monochrome monitors, and colvid.bin for color monitors.

If Your Terminal Isn't Configured

There is a list of terminals for which JYACC distributes configuration files in an Appendix to this chapter. If you cannot find distributed configuration files for your specific model, check for emulations. Many popular terminals, for instance, emulate the DEC VT-100; others may support the ANSI standard escape sequences. If that doesn't work, you will need to create your own; the JAM Configuration Guide will help you through that process.

6.2 JAM Function Keys

JAM interprets a number of keys specially. Here is a list of their names and functions. To find out how these logical functions are assigned to your terminal's keys, examine the key translation file, or run the modkey utility on it. Modkey, described in the Configuration Guide, contains a key translation test screen that you can use to check your key mappings. To find the key file, see the previous section; listings for the IBM PC and Wyse 85 are appended as examples. There is a much more detailed summary of special keys in the Author's Guide, in the section on data entry.

JAM Navigation Functions

TRANSMIT	Menu selection or end of data entry EXIT
	Abort data entry, return to previous screen SPF1
	Return to top-level screen SPF2
	Escape temporarily to the operating system SPF3
	Go directly to a named screen

JAM Authoring Functions

SPF5	Invoke the screen editor SPF6
	Invoke the data dictionary editor

Cursor Motion

Up Arrow	Cursor up one line or field Down Arrow
	Cursor down one line or field Left Arrow
	Cursor left one column or field Right Arrow
	Cursor right one column or field Tab
	Next field Backtab
	Previous field Return
	Next field on following line Page Up
	Scroll data up in scrolling field Page Down
	Scroll data down in scrolling field

Data Editing

Insert	Toggles insert/overwrite mode Delete
	Deletes character under cursor Backspace
	Deletes character to left of cursor Field Erase
	Erase from cursor to end of field Clear Screen
	Erase all unprotected fields

Application Functions

PF1-PF24	These are commonly assigned to the otherwise APP1-APP24 unnamed function keys on a terminal. SPF7-SPF24

6.3 Organizing JAM Applications

It is a good idea to keep all the screens belonging to a JAM application in one directory. Limited sharing of screens among applications can be accomplished by use of the SMPATH environment variable, which is explained in the Programmer's Guide. Application code can be maintained in subdirectories of the application directory. This is generally more convenient than placing screens and code in subdirectories of a common parent.

7 A JAM Glossary

application environment
See run-time environment.

application shell The structure of JAM screens and links that defines the look and control flow of an application; everything but the code.

array Several fields grouped together in one place, that can be treated as a unit. The elements of an array share all characteristics, such as scrolling, and can be referred to as occurrences of the first field in the array.

attached function An application routine associated with a field that is called with certain parameters whenever the cursor enters or exits the field. Also, an application routine associated with a screen and called upon screen entry or exit.

author A person whose task is to create a JAM application shell or prototype; connotes an application designer without training in programming.

authoring environment The tools used to create and test JAM screens and links, comprising the screen editor, data dictionary editor, and run-time environment.

border Text or highlighting used to mark the outline of a screen.

character edits A field's character edit defines what type of character may be entered in a field, such as digits, letters, or a yes-or-no answer.

control field In a JAM menu, a field following a menu selection that contains a control string to be executed when that selection is chosen.

control link An association between something a JAM user does (menu selection or function key) and JAM's response (bring up a new screen or call an application program).

control path A list (actually a stack) of the names of all the screens entered via JAM control links, and not yet exited.

control string A special string, usually not displayed, used to implement control links. The name or location of a control string identifies the event that triggers the link, while its contents define the action to be taken.

cursor A special marker on the display, commonly a blinking block or underline, that shows where the text you type will go.

data dictionary A list of named data items and all their characteristics.

data link The sharing of values among data items in different screens that share a name with a data dictionary entry.

display A physical screen, such as a terminal on a multi-user computer or the monitor on a personal computer.

display attribute Visible characteristics of data on the screen, such as color, highlighting, underlining, or blinking.

display data	The fixed part of a screen: text, borders, and graphics that do not change. Distinct from fields, which may be altered by the program or by data entry.
element	A field that is part of an array. An array element may be referred to either by its own field number, or by the name of the array plus its element number. A non-array field is considered to have a single element.
element number	A field's element number is its position within the array it belongs to. The element number of a simple field is 1.
field	A variable area of a screen, used for the exchange of data between an application and its user. A single field occupies part or all of one line. It may be extended horizontally through shifting, and vertically through scrolling. Fields may have many characteristics and actions associated with them, known variously as edits, attachments, and validations.
field attachment	An item associated with but distinct from a field. Examples include a help screen, a prompt, an attached function, a calculation, or a menu of possible items for data entry.
field edits	Field edits either restrict the data that can be entered into a field, or alter its appearance. Examples include a range of permissible values, right justification, conversion to upper case, and dollar amount format.
field number	JAM numbers fields according to their positions within a screen, from left to right then top to bottom, beginning at 1. When a field is spoken of as "next" or "following" another field, this is the ordering that applies.
field validation	An action associated with a field that checks data entered there for correctness.
form	A screen that occupies the entire display and does not overlay another screen, as opposed to a window. Often used loosely as synonymous with screen.
function key	A key that has some special function other than data entry, for instance cursor motion. JAM treats such keys as logical keys, referring to what they do rather than to their labels on the keyboard, since the labels are different for the many keyboards it supports.
function list	A list of pairs of function names and addresses, compiled into JAM applications to provide necessary linkage.
help screen	A screen containing any information helpful to the user of a JAM application. Help screens may be attached to JAM screens and fields; they appear when the HELP or FORMHELP function keys are struck.
hook	A software device by which an application routine is made known to JAM, specifying its name, address, and language. The two most important kinds of hooks are attached functions and invoked functions.
invoked function	An application routine that is triggered by a control link. Also called a caret function.

item	Data entered into a scrolling field, or into one field of a scrolling array.
item number	The position of an item within its scroll list. The item number does not depend on the item's position on the screen.
justification	Data in JAM fields may be either right- or left-justified, that is, pushed all the way to the right- or left-hand end of the field.
LDB	See local data block.
library	The JAM function library, which contains routines application programmers can use to access data in screens and the local data block.
link	See control link and data link.
local data block	A dynamic copy of the data dictionary, which holds the values of items that have been changed by data entry or program action.
logical key	JAM's interpretation of a function key, as opposed to the physical key on a terminal. Physical keys are mapped to logical keys by a configuration file.
menu	A screen containing a list of choices, from which the user may select one.
occurrence	A general term covering simple fields, array elements, and items of scroll lists. In a scrolling field or array, occurrence is equivalent to item; in a non-scrolling array, it is equivalent to element.
occurrence number	A data item's element number or item number, whichever applies. If the field is neither scrolling nor part of an array, the occurrence number is 1.
parallel array	Scrolling arrays placed next to one another will scroll in parallel, i.e. whenever one array is scrolled with the cursor or page keys the associated arrays scroll simultaneously.
prompt	Text associated with a field that appears on the terminal's status line whenever the cursor enters the field. Also called status text.
prototype	A collection of JAM screens and control links used for testing the user interface to a new application; similar to the application shell, but used in different circumstances.
protected field	A field into which no data may be entered from the keyboard.
run-time environment	The JAM code that processes control links, displays screens, controls data entry, and handles the calling of application routines.
scope	Data dictionary entries all have a scope, which is a number between one and nine. Entries having the same scope can be erased and reinitialized as a group.

screen	Data to be displayed on a computer's terminal or display, such as menus and data entry forms. When the hardware display itself is meant, the terms physical screen or display are used.
screen editor	A JAM tool used to create and alter screens and control links.
scrolling	JAM screens may contain data lists that are too long to fit in available space; such lists may be scrolled, either in a single field or in an array of fields. The cursor motion keys cause different parts of the list to appear on the screen.
scroll list	A data list displayed through a scrolling field or array.
shifting	A data item too wide to fit in a field may be shifted horizontally; the cursor motion keys will cause different parts of the item to appear in the field.
status line	JAM sets aside one line of the physical screen, usually the bottom one, for error and status messages; it is called the status line.
status text	See prompt.
system date	The current date, as stored in the computer. JAM date fields can be automatically initialized to the system date.
system time	The current time, as stored in the computer. JAM time fields can be automatically initialized to the system time.
transaction	A related group of screens and data items.
user date	A date entered into a JAM date field by the user of an applciation.
user time	A time entered into a JAM time field by the user of an application.
window	A screen that normally does not cover the whole physical screen, and overlays some other screen or screens.
word wrap	Fields and scrolling arrays may have a word wrap edit, which will cause whole words to be kept together on the same line of text. (JAM's default is to fill the field with characters, without regard to word spacing.)
zoom	Shifting and scrolling fields may be viewed and edited as a whole, in a pop-up window, using a special zoom key.

Appendix A Sample Key Assignments

You will find explanations of the key names used here in the section of the Author's Guide entitled Data Entry.

JAM key assignments for the IBM PC family:

```
EXIT         = Esc TRANSMIT
             = End HELP
             = control-F1 FORM HELP
             = alt-F1 LOCAL PRINT
             = control-P RETURN
             = Enter TAB
             = Tab BACKTAB
             = shift-Tab BACKSPACE
             = control-H HOME
             = Home PAGE UP
             = Pg Up PAGE DOWN
             = Pg Dn INSERT MODE
             = Ins INSERT LINE
             = control-K DELETE CHAR
             = Del ERASE
             = control-Pg Up CLEAR ALL
             = control-Pg Dn ZOOM
             = control-Z PF1
             = F1 ... PF10
             = F10 SPF1
             = shift-F1 ... SPF10
             = shift-F10
```

JAM key assignments for the Wyse 85:

```
EXIT         = F11 TRANSMIT
             = Do HELP
             = Help TAB
             = Tab or control-I
             BACKTAB
             = F12 HOME
             = F14 BACKSPACE
             = control-H DELETE CHAR
             = Remove INSERT MODE
             = Insert Here ERASE
             = Select CLEAR ALL
             = control-Z PAGE DOWN
             = Next Scrn PAGE UP
             = Prev Scrn RESCREEN
             = Find ZOOM
             = control-E PF2
             = F6 ... PF6
             = F10 PF7
             = F17 ... PF10
             = F20 SPF1
             = PF4 1 ... SPF9
             = PF4 9
```

Appendix B List of Supported Terminals and Emulators

The following list is subject to constant revision, usually by having more things added to it. The mnemonics listed can be found as prefixes to key and video files in the config subdirectory of your JAM distribution. As distributed by JYACC, names of video files end in vid and names of key files end in keys. You may find that you need to shorten or otherwise alter some of the names, to suit your operating system or your own naming conventions.

Terminal mnemonic	Description
5425t	AT&T 4425 terminal.
7900	NCR 7900 M1+ terminal.
FT	Fortune Systems terminal.
TV9220	TeleVideo 9220 terminal. Also found as NTV9220, WTV9220 for 80- and 132-column modes respectively.
TVO	TeleVideo 955 terminal with onscreen attributes.
W85	Wyse 85 terminal. Also found as NW85, WW85 for 80- and 132-column modes respectively.
a219	Ampex 219 terminal (in native mode).
ansi	Color PC console for SCO XENIX.
avt	HDS AVT terminal.
bw	Monochrome monitor on MS-DOS system.
c108	Concept 108 terminal.
col	Color monitor on MS-DOS system.
cpt200	Color PC with PCLINK emulating a PRIME PT200 terminal.
d214	Data General Dasher 214 terminal (in DG mode).
f100	Freedom 100 terminal.
h0	Honeywell VIP 7300 terminal.
hds	HDS model 200 terminal.
host	Basic ANSI terminal or emulator, with color.
hostpc	Stratus PCTERM emulator, monochrome or color.
hp	Hewlett-Packard 2392a terminal.
iii	TeleVideo 955 terminal, with onscreen attributes and color.
j8, j8c	Monochrome and color PC, respectively, with JYACC jterm emulator and 8-bit control sequences.
jterm, jtermc	Monochrome and color PC, respectively, with JYACC jterm emulator.

opus220	Opus 220 terminal.
pc	Either monochrome or color PC with generic VT-100 emulator.
pt200, pt132	PRIME pt200 terminal, in 80- or 132-column mode; also monochrome PC with PCLINK emulator.
pt200w	PRIME pt200 in 48-line mode.
svt200	Sperry SVT1220 terminal.
ti931	Texas Instruments 931 terminal.
tvi921	TeleVideo 921 terminal with onscreen attributes.
tvi950	TeleVideo 950 terminal with area attributes.
tvi955	TeleVideo 955 terminal with area attributes.
v101	Stratus V101 terminal.
v102	Stratus V102 terminal.
vt100	DEC VT-100 terminal or emulator.
vt200	DEC VT-200 terminal or emulator.
vt220	DEC vt220 terminal or emulator.
wy30	Wyse 30 terminal, or HP 700/41 emulating the same.
wy50	Wyse 50 terminal.
wy75	Wyse 75 terminal.
x100, x100c	Monochrome or color PC with Crosstalk emulating a VT-100.

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.

definition 1-13