

PRODUCTION NOTE: This printed journal was produced in camera-ready format with a desktop publishing system employing an Apple Macintosh and LaserWriter system, and the following software: MacPaint, MacDraw, and MacWrite from Apple Computer, Inc; PageMaker from Aldus Corporation; ReadySetGo from Manhattan Graphics; plus Desktop Magic and LaserMagic from World Class Software.

Creative Staff:
Randy Thompson
Roger Wood
Norman Winney

Editorial Consultant
Gary M. Kaplan

HCJ FEATURES

- 2 FormFlex™**
No more writer's cramps. . . no more loose papers cluttering your desktop. With this powerful-yet-flexible utility, you'll put everything in perfect form.
- 9 Re-Weaver™**
Re-Weave your *WordWeave* files *your way*. . . and print out a customized masterpiece.
- 12 Upsets™**
A challenging, yet very easy-to-learn educational brain-game based on the concepts of set theory.

HCJ TECH NOTES

- 15 Apple:**
Applesoft Line Input
In ProDOS
- 16 Atari:**
Joystick-Controlled
Cursor
- 17 Commodore:**
Alpha Lock
- 18 IBM PC & PCjr:**
Get Seg
- 19 TI:**
File Verify

HCJ FOCUS

- 20 Apple:**
Text On The Hi-Res Screen
Now you can display any of the Apple's characters on the hi-res screen with a simple PRINT command.
- 22 Atari:**
Pull-Down Menus
Order up some pull-down menus—the perfect choice for the byte-conscious gourmet.
- 24 Commodore:**
Hi-Res Graphics
Get out your 3D glasses, because the C-64 is crossing the dimension barrier. . .
- 26 IBM PC & PCjr:**
PC Character Editor
Adding character to your BASIC programs.
- 28 Texas Instruments:**
HCJWord
Ti-Writer, look out! Here comes HCJWord. . .

HCJ DIRECTOR

- 30 HCJ ON DISK™**
Directories
Menu-driven program selection plus HCJ CodeWorks secrets revealed.

Home Computing Journal (HCJ) is a quarterly multi-media software subscription service containing ready-to-run productivity, education, entertainment, and utility programs on a floppy disk. The accompanying workbook contains the required support documentation plus additional technical notes and programming aids. For current single-copy and subscription pricing, please see last page. HCJ assumes no liability for errors in articles, programs, or workbook material.

Each separate workbook selection and software contribution to this Volume and the Volume as a collective work is Copyright © 1986 by Home Computing Journal. All rights reserved. Visual elements and design of all pages Copyright © 1986 by Home Computing Journal. All rights reserved.

Home Computing Journal is the owner of all rights to the computer programs and software published in this printed workbook and on its companion disk. To allow for convenience in the use of this software by the purchaser, HCJ grants to such purchaser only, the Limited License to enter these programs into the purchaser's computer, place a disk-copy of this software on the purchaser's hard-disk drive, and/or make a reasonable number of back-up safety copies for the purchaser's personal use. Any other use, distribution, sale, or copying of this software is expressly prohibited and is in violation of this Limited License and the copyright laws.

All indicated trademarks (TM), unless otherwise specified, are the property of Home Computing Journal.

Individual volumes, bulk orders, and subscriptions are available.
See last page and/or inquire for current pricing and catalog:

Home Computing Journal
P.O. Box 70248 • Eugene, OR 97401
Tel. (503) 342-4013

*No more writer's cramps...
no more loose papers
cluttering your desktop.
With this powerful-yet-
flexible utility, you'll put
everything in perfect form.*

Reports, expense records, data sheets, contracts, purchase receipts... In today's complex society, it seems like the most common means of communicating and organizing our transactions, thoughts, records, and activities are forms. All-purpose forms are not always suited to our needs, however, and we often find ourselves altering them to fit our requirements. But rather than scratching out words or scribbling in text between the lines, why not create your own custom forms? *FormFlex* lets you do just that.

With *FormFlex*, you can design forms and print them with a standard 80-column printer. These forms have an advantage over pre-made forms in more ways than one: You can store the forms on a disk to edit and print as the need arises; you can

create special data fields; and you can actually perform calculations in these fields, using addition, subtraction, multiplication, and division.

Getting Started

Before you can use *FormFlex*, you must first create a work disk. This is done by copying certain files from your HCJ Volume 2 disk to a notched (non-write-protected) disk. See your computer's accompanying sidebar for specific instructions on creating a *FormFlex* work disk for your computer.

In Good Form

When you RUN *FormFlex*, the program displays a title screen during initialization, then takes you to the main menu screen:

FormFlex

- 1) Make Form
- 2) Use Form
- 3) Exit FormFlex

The first two options actually represent two separate programs that are loaded from disk when selected. For this reason, it is important that you keep your work disk in the drive at all times.

Using *FormFlex* requires two steps: The first step is to *make* a form (option 1). This master form comprises the standard information that will appear on the final printout, and a list of all the different information fields. You can save each form to a disk and call it up later for re-use or modification. The second step is to *use* the form (option 2) by entering data into each field of the master form, and printing the finished product. You may also save the field information to disk for later editing or for merging into other forms.

To help you learn how to use *FormFlex*, we are going to show you a simple form that we have created, and how it can be customized for your own purposes. We have provided this form on your HCJ Volume 2 disk under the file name COSTCOMP. Note that an extension of 2 to 4 characters is used by the program, and is not entered by the user when Loading or Saving from the program. For a list of these extensions, see your computer's "Special Hints" sidebar. We suggest that you work directly with the program while reading these instructions. For starters, create a work disk, load and run *FormFlex*, and then select the Make Form option from the above menu.

ABC Company
Cost Comparison Form

Project No.: _____ Date: _____
 Company Name: _____
 Contact: _____ Tel: _____
 Address: _____
 City: _____
 State: _____ Zip: _____

Item Descriptions

Item A: _____
 A1 Source: _____
 A2 Source: _____
 A3 Source: _____
 A4 Source: _____

Item B: _____
 B1 Source: _____
 B2 Source: _____
 B3 Source: _____
 B4 Source: _____

Low Quote 1	Quote 2	Quote 3	High Quote 4	Average Quote	High-Low % Diff.
A1 \$ _____	A2 \$ _____	A3 \$ _____	A4 \$ _____	Aav\$ _____	Adf % _____
B1 \$ _____	B2 \$ _____	B3 \$ _____	B4 \$ _____	Bav\$ _____	Bdf % _____

High A \$ _____ High B \$ _____ High Total \$ _____
 Low A \$ _____ Low B \$ _____ Low Total \$ _____

Average: _____
 % Difference: _____

Figure 1.
This is the sample form that you will find on your HCJ Volume 2 disk with the file name: COSTCOMP.



Make Form

When selected, the Make Form program module is loaded from disk, and the following menu is presented:

Make Form

- 1) Edit Form
- 2) Erase Form
- 3) Change Formula or Edit Field
- 4) Disk Routines
- 5) Printer Routines
- 6) Return to FormFlex Menu

First, let's load our sample form into memory. This is accomplished via the Disk Routines menu. To get this menu, select option 4 from the above menu. This is what you should see:

Disk Routines

- 1) Load Form
- 2) Save Form
- 3) Return to Make Form Menu

By choosing option 1, we can Load our form. When the computer prompts you, enter the file name COSTCOMP. As you can see, if we were to create a new form or modify an old one, we could Save the form using option 2 of the Disk Routines menu. Whenever you Save a form, the computer checks the disk to see if a form with the same name already exists. If it does, you are asked if you wish to replace the existing form with the current one.

Once our file is loaded, choose option 3 to return to the Make Form menu. Now, let's take a look at our form. To do this, select option 1, Edit Form.

Edit Form

This part of the program is where you create new forms and edit previously existing ones. Here, you are provided with a simple text editor for the designing of forms. Before we make any changes to the form that we have just loaded, let's go through a brief explanation of the Editor.

The Editor's screen layout is as follows: The top few lines of the screen are for displaying messages and receiving information such as file names. The rest of the screen is for the actual editing of a form. The upper-left corner of the screen shows the current row and column position of the cursor. The row can vary between 1 and 60 and the column between 1 and 80. These dimensions cover an 8-1/2" by 11" piece of paper. The computer's screen acts as a window into this page of text. You can move your window to display any

part of the page that you desire. When you move your cursor off the screen, the computer automatically updates its viewing window in order to show the section of the page that the cursor has moved to.

All of your computer's normal editing functions are available. There are even some extra ones such as deleting and inserting whole lines of text, and paging quickly through the form. For a list of editing keys and their functions, refer to your computer's Control Capsule.

Text & Fields

Any text you enter during Make Form will appear on the final printed form just as you enter it. On the other hand, if you wish to designate a "blank" on the form to be filled in with certain information, you must declare it as a "field." There are 5 different types of fields that you can specify on any form: Alpha, Uppercase, Edit Characters, Numeric, and Formula. These fields are represented on the screen by the inverse letters a for Alpha, u for Uppercase, e for Edit Characters, n for Numeric, and f for Formula. Your computer's Control Capsule shows you the exact keystrokes required to enter these characters. The only other inverse character is one designating a decimal-point location in either a numeric or formula field. By selecting a decimal location, you determine the format any number will have in a filled-out form. Again, your Control Capsule shows you how to enter this character.

Our Sample File—Cost Comparison Form

To best understand the rules for using *FormFlex*, we will refer to our sample form, COSTCOMP (see Figure 1). Note that this sample is only provided to demonstrate the different functions available in *FormFlex*; it should *not* be viewed as all this powerful "form processor" is capable of producing. COSTCOMP is designed for a hypothetical company (the ABC Company) to compare the cost of various bids for parts from four sources. By using the *FormFlex* Formula feature, the program provides calculation of the average bid for each of two items (designated A and B), and the percent difference between the highest and lowest bid. It also calculates the highest and lowest total bid for both items, along with the average total bid and the percent difference between high and low bids.

Across the top, is the header for the form—ABC Company, etc. This is not a field or a field name—it's just text entered from the Make Form Editor. A field name comprises the characters immediately to the left of any field of inverse characters. In this case, the first field name is "Project No.:"—with the 8 inverse n characters on the screen (depicted

Figure 2. Field Types

Name	You see	Legal Characters
Alpha	a	All printable characters
Uppercase	u	Uppercase letters only
Edit Characters	e	You choose the legal characters
Numeric	n	Digits, decimal point, and minus sign
Formula	f	Contents of field is calculated by formula

Maximum Lengths of Fields and Field Names

Computer	Maximum Field Length	Maximum Field Name Length
Apple	79	79
Atari	25	15
C-64	25	15
IBM	79	79
TI-99/4A	25	15

Atari, C-64, & TI-99/4A Users: A field name may actually exceed 15 characters, but only the rightmost 15 characters are considered to be the field name. In other words, if you declare a field name to be

Cost per thousand

the computer will only consider the rightmost 15 characters as the actual field name in formulas, etc.—i.e.,

st per thousand

Apple & IBM Users: All fields must be on the same line. Therefore, both the Apple and IBM programs allow a combined total-length maximum of 80 characters for Field Name Length and Field Length (i.e., Field Name Length + Field Length must be less than or equal to 80).



SPECIAL HINTS

Special Note: *FormFlex* requires either an Apple IIe with an 80-column card or an Apple IIc.

Making an Apple *FormFlex* Work Disk

Format a ProDOS disk using the HCJ Format program supplied on your HCJ Volume 2 disk. Return to Applesoft BASIC, then **LOAD** each of the following programs into memory, and, in turn, **SAVE** each file to the newly formatted disk.

FORMFLEX
MAKEFORM
USEFORM

To use the demonstration file COSTCOMP, follow these steps:

1. With your *FormFlex* work disk in the startup drive, type

RUN FORMFLEX

2. Select option 1, Make Form.
3. Select option 4, Disk Routines.
4. Place the HCJ Volume 2 disk in the startup drive.
5. Select option 1, Load Form.
6. Enter the file name COSTCOMP
7. After the file has finished loading, remove the HCJ Volume 2 disk from the drive and put your *FormFlex* work disk in the drive and select option 2, Save Form.

8. When the default file name (now COSTCOMP) appears, press [RETURN] and the form will be saved for future use and modification on your *FormFlex* work disk.

File Name Extensions

The Apple version uses the following file name extensions to differentiate the data files created by *FormFlex*:

A raw form	.RAW
An evaluated form	.FRM
Field data	.DAT
Form saved as ASCII	-none used-

Speeding Up Disk Access

It is possible to enter any character on a form, but the BASIC INPUT statement cannot successfully read a comma or a colon that has been written to disk. We could have circumvented this problem using the GET command, but disk access would have been too slow. To remedy this situation, we built a special machine-language routine into *FormFlex* that allows the reading of any text character from disk at the speed associated with the INPUT statement. For a detailed discussion of how this routine works, see the Apple Tech Note in this Volume.

row, the characters to the left of the spaces are not considered part of the field name. Likewise, the field name cannot have more than one space separating it from its field—but a space is *not* required. For instance, the numeric field at the beginning of row 27 (field name "A1 \$") has no intervening space between the name and the field.

Each "field name/field" combination is restricted to a single row on the form—i.e., a field name must start on the same line as the field, and the field cannot go beyond the 80th column and wrap to the next line. Also, you cannot mix field types. For example, a field is either Alpha or Uppercase. Of course, because an Alpha field can contain any printable character, it could contain uppercase characters—but you cannot *force* part of a single field to be uppercase, and allow part of the same field to be any other printable character.

The Two Phases Of Evaluation

After you have completed designing your form, you can evaluate it by pressing the key that returns you to the main menu. Evaluation is a two-phase process. The first phase encompasses the rules defined above. (See Figure 3 for a complete list of possible errors in phase 1.) The second phase allows you to define characters for Edit Character fields and Formulas. The number of characters you can specify for an Edit Character field or within a Formula is limited by the space allowed in your computer's input box at the top of the screen.

A good example of an Edit Character field is the "Tel:" field of our COSTCOMP form. If you load this form into the *Use Form* program, you will only be able to enter the characters used to depict telephone numbers—i.e., 1234567890()-

Formula Hints

A *FormFlex* formula is a mathematical expression that can contain parentheses, digits, decimal points, the 4 major math operators (+, -, *, and /) and variables (which are complete field names). The precedence of operators in a *FormFlex* Formula is the same as in BASIC: Multiplication and division are done first, and then addition and subtraction. You can alter the order of precedence by using parentheses.

There are some things that you should consider when using the *FormFlex* formula feature. First off, the field names should be kept short. Because the length of a formula is limited to the size of the input box, the shorter the field names, the more complex formulas you can enter.

Try to avoid using numbers and operators (+, -, *, and /) within a field's name. This will only confuse you and the computer as to what is supposed to be calculated for a formula. For example, what if you had a field with the name 12, and one of your formulas is 12/2? Because *FormFlex* evaluates all field names before it evaluates numeric quantities within a formula, it would interpret the 12 as a field name and *not* as the number 12.

Formulas are calculated in the order in which they appear on a form—going left to right, and then top to bottom. Because of this, you should avoid creating Formula fields that depend upon a value from a *subsequent* Formula field in its calculation.

Figure 3.

Evaluation Errors

Error Message	Description
Field must have a name	There are no characters to the left of a field to identify it.
Mixed field characters	You have attempted to mix two or more field types inside one field.
Duplicate field name	You have two or more fields with the same name.
Too many fields	You have more than 80 fields.
Field too long*	A field's length exceeds 25 characters.
Field name too long*	A field's name exceeds 15 characters.

*Atari, Commodore 64, and TI-99/4A versions only

by 8 underlines on the sample printout) representing the field identified by its name.

Design & Evaluate

As you design a form, you can enter anything you please. When you escape from the Editor to return to the Main menu, however, (see your Control Capsule for the appropriate keypress) the computer evaluates the form. One criterion of evaluation is that each field *must* have a field name.

There is another restriction on any form you design: Due to memory limitations, the form may have no more than 80 fields. To ensure that each field in a form is identifiable, each must have a *unique* field name. If two fields have identical field names, you will receive a "Duplicate field name" error upon evaluation.

A field name can be any series of characters. Figure 2 shows the different field types and lists the restrictions for length of field names and fields for each computer. The field name may contain spaces, but if the computer encounters 2 spaces in a

Other Editor Features

The *Make Form Editor* is a miniature word processor. Although it does not have fancy features (like cut and paste, merge, or print formatting for bold, italic, etc.), it is an easy-to-use text editor in which each location on the piece of paper is accurately identified by the Row and Column indicator (found in the upper-left corner of the screen). Your control Capsule carefully details all of the editing keypresses.

For added convenience, the Editor allows you to Save and Load "raw" or unevaluated forms. You can use the same name for a raw or an evaluated form because the program adds an extension on to the name for each type of file so it can tell them apart. (See Figure 4 for a chart of the different types of files that are created using the *FormFlex* programs.) This means that you can work on a form in several short sessions before it is evaluated—by Loading and Saving the raw form until you are ready to go through the evaluation process. You can also erase any form from the computer's memory with a simple keypress. (Don't worry, if you press the Erase Form key by mistake, you are given a chance to change your mind.) If you want to return to the Main Menu without evaluating a form, first Save the raw form to disk, then Erase the form from memory, exit the Editor, and no Evaluation will occur.

Main Menu Revisited

The *Make Form* Main Menu allows you to access routines to alter and print out a fully evaluated form. You can Erase the form using option 2. You can change any formula or alter the characters that will be acceptable in an Edit Character field using option 3. Disk Routines (option 4) were explained above.

If you select option 5—Printer Routines—you are presented with this menu:

Printer Routines

- 1) Print Form
- 2) Print Field Specifications
- 3) Return to Make Form Menu

The Print Form option lets you get a printout of the form, and puts blanks (underlines) where all the field data will go. If you just want to make up a quick master form for taking notes, this printout will suffice. If, however, you want to produce a form that allows you to enter data via the computer into a set of related forms, then this printout will just be a hardcopy master for the filled-out forms you create in the *Use Form* program.

The Print Field Specifications option prints out a list of all the field information. The field specification provides you with the field name, the field type, the field length, the row where the field appears on the form, and the column where the field begins in that row. For example, the field specification for the "Project No.:" field is

Field name: Project No.:
Field type: NUMERIC
Field Length: 8
ROW: 4 COL: 19



SPECIAL HINTS

Making an Atari *FormFlex* Work Disk

First use DOS 2.5 to format a disk to be your *FormFlex* work disk. Then use DOS to copy the following files from your HCJ Volume 2 disk to the newly formatted *FormFlex* work disk:

FORMFLEX
MAKEFORM
USEFORM
COSTCOMP.FRM

File Name Extensions

The Atari version uses the following file name extensions to differentiate the data files created by *FormFlex*:

A raw form	.RAW
An evaluated form	.FRM
Field data	.DAT
Form saved as ASCII	-none used-

Making Editing A 'Joy'

This Volume's Atari Tech Note presents a routine that allows you to control the computer's cursor with a joystick. If you run this program prior to running *FormFlex*, you can use a joystick to move the cursor while editing a form.

Now, how about making the joystick control the Page left, right, up and down functions? This can be accomplished fairly easily: Right after you run the BASIC file JOYCSR, execute the following POKES:

POKE 1788,130:POKE 1789,128:
POKE 1790,138:POKE 1791,136

Now, run *FormFlex* and start editing a form. You will find that moving the joystick left, right, up and down has the same effect as pressing the page left, right, up and down keys. To understand why this works, read this Volume's Atari Tech Note.

If the field is a Formula or Edit Characters field, the field specification also includes the formula or Edit Characters for that field. For example, the field specification for the "Aav\$" field is

Field name: Aav\$
Field type: FORMULA
Formula: (A1 \$ + A2 \$ + A3 \$ + A4 \$) / 4
Field Length: 8
ROW: 27 COL: 61

Use Form

Once you have completed a master form design, you are ready to select *Use Form* program from the main menu so you can fill out the form. Any master form that you design using *Make Form* can be loaded into *Use Form* and filled out. Then, that particular set of field data can be saved to its own data file. You may then clear the field data, fill the form out again with new data, and save it to its own file. By repeating this process, you can create any number of filled-out forms—each saved in its own data file.

In our Cost Comparison form example, we can load the COSTCOMP file into memory, then fill it out with data for a particular job, and save it with a file name relating to that job. In short, the COSTCOMP master form becomes a template for any forms we want to fill out with that type of information.

To run *Use Form*, select option 2 from the *FormFlex* Main Menu. If you are using *Make Form*, you must first select option 6 to return to the *FormFlex* Main Menu.

When you select *Use Form*, it is loaded from disk and run. You are first asked to enter a file name. This is the name of a form that you created using *Make Form*. The reason *Use Form*

Formulating Formulas For Forms

Below are two formulas that are used in our sample form, COSTCOMP. The first formula calculates the average quoted price for Item A. The result is stored in the "Aav\$" field. The second formula calculates the total for high quotes. Notice how field names such as "A1 \$" and "High A \$" are used in the formulas.

Formula for the "Aav\$" field:

(A1 \$ + A2 \$ + A3 \$ + A4 \$) / 4

Formula for the "High Total \$" field:

High A \$ + High B \$

Hint: To keep calculating time to a minimum, try to use formulas sparingly, and limit the total number of fields in your form. Calculating time ranges from especially slow on the TI and Atari versions, to fairly slow on the C-64 and Apple versions, to instantaneous on the compiled IBM version.



SPECIAL HINTS

Making a C-64 FormFlex Work Disk

Format a disk using the following command:
OPEN15,8,15,"NO:FORMWORK,V2":CLOSE15

Then Copy the following files to the newly formatted disk using any file-copy program such as COPY/ALL (on the Test/Demo disk that comes with the 1541 disk drive), or UNI-COPY or SD.COPY .C64 (on the Test/Demo disk that comes with the 1571 disk drive).

```
formflex
makeform
useform
boot.form
master64I10u-a
master64I10u-b
costcomp.frm
```

File Name Extensions

The Commodore version uses the following file name extensions to differentiate the data files created by *FormFlex*:

A raw form	.RAW
An evaluated form	.FRM
Field data	.DAT
Form saved as ASCII	-none used-

Quicker Editing

The C-64's editor allows you to shift the screen horizontally across a form one character at a time. When you attempt to type off the right edge of the

screen, for example, your view of the form shifts, allowing you to see where you are typing. Most of the time, this capability is very useful, but having to wait for the screen to update with every keypress can really slow you down. When this occurs, we suggest that you press **f5** (to page right) before attempting to type any further. This will speed up the creation of forms immensely.

Master-64

The Commodore 64 version of *FormFlex* was written in a programming language called Master-64, available from Abacus Software. Master-64 adds several commands to the 64's built in BASIC. When you first run *FormFlex*, Master-64's run-time package (contained in the files master64I10u-a and master64I10u-b) is loaded from disk to install the added commands.

In order to edit the *FormFlex* program files, you need to own Master-64, as the run-time package does not allow program modification. Normally, you are not even able to list the program without owning Master-64. We have, however, discovered a loophole: To list a *FormFlex* program using the run-time package, first exit *FormFlex* by choosing option three from the Main Menu. Now, load in the Master-64 program that you wish to list (i.e., makeform, useform, or boot.form). Finally, to list the program, enter **FIND @@**. The program will now list to the screen.

your input. You can enter any legal characters for that field. In our Cost Comparison form, for example, the cursor will be at the beginning of the "Project No.:" field. Because this is a numeric field, you are only able to enter digits, decimal points, or minus signs. When you exit the field (see the Control Capsule for the appropriate keypress for Fill Out Form mode), the number will be formatted to the proper number of decimal places. In the case of the "Project No.:" field, if you do insert a decimal point when you move to the next field, the number is rounded to the nearest whole number, and no decimal point is displayed. This is because no decimal point was specified when the form was created using *Make Form*.

By using the appropriate keys, you can edit all fields in the form except Formula fields, which are calculated by the computer upon request (see Control Capsule). If your form has very many fields or complex formulas, this figuring process can take a long time. Thus, it is a good idea to only request the computer to calculate the formulas when you have completely finished filling in the other fields. (See previous sidebar, "Formulating Formulas for Forms.")

There are several keypress functions available in Fill Out Form mode that allow you to save and load the field data to and from disk, get a printout of the form, clear the fields of all data, etc. Check your Control Capsule for a complete list of what is available on your computer. These functions are for your convenience, so you don't have to exit to the Main Menu to do general housekeeping chores.

Disk Routines

When you save data from *Use Form*, the program saves the field data associated with each field name. Because you will probably want to have several different filled-out forms for each form that you create in *Make Form*, the name for each *Use Form* data file should be different from the name of the master form created with *Make Form*.

A handy feature of *Use Form* allows you to save any filled-out form as an ASCII text file, and then load it into your favorite word processor (if it can import straight ASCII files). Just select this option from the Disk Routines menu, supply a file name, and the file will be saved.

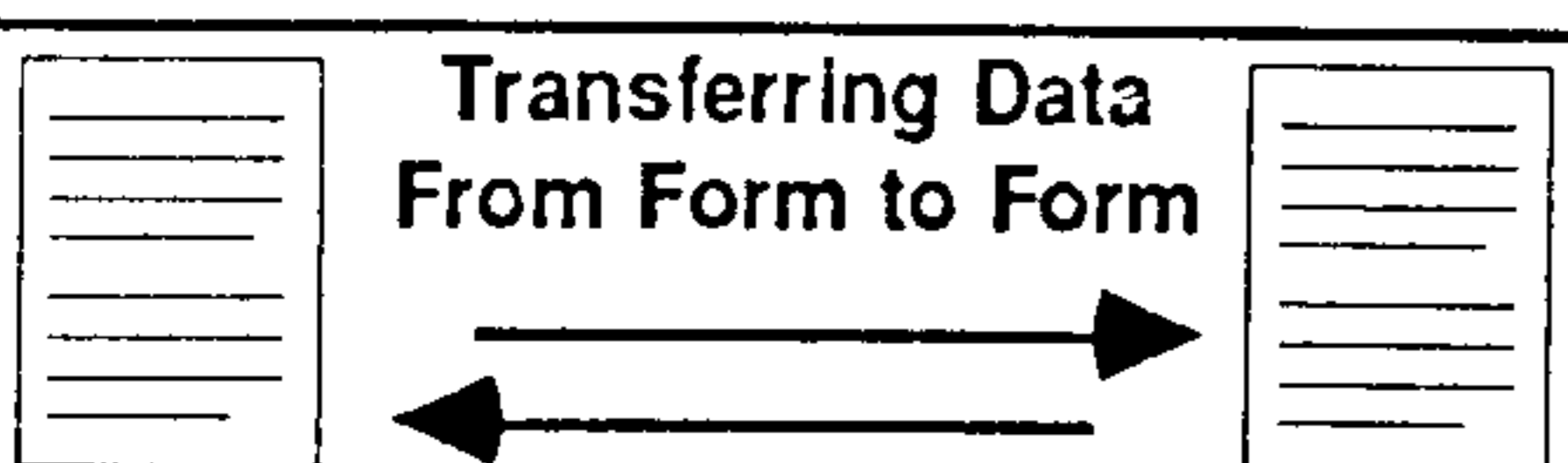
Printer Routines

When you select the Print Routines option from the *Use Form* main menu, you are presented with this menu:

Printer Routines

- 1) Print Filled Out Form
- 2) Print Field Data
- 3) Return to Use Form Main Menu

Option 1 allows you to get a printed copy of the complete form, with all field data in place. Option 2 sends all field names and the associated field data of the form to the printer.



UseForm's Load Field Data option allows you to transfer field data from one form to another. As long as the data being loaded comes from a field with the same name as a field in the present form, the data is transferred. If the data being loaded originates from a field that does not have the same name as a field in the present form, the data is ignored.

For example, let's say you have two separate forms, each with a "Name" and an "Address" field. If you fill out one form and save the field data, and then load this same data into the second form, the "Name" and "Address" data are transferred to the second form.

begins by prompting you for a file name is that you must load a form created by *Make Form* to use any of *Use Form*'s operations. Thus, to enter data into the Cost Comparison form, enter the file name **COSTCOMP**. After you enter the name, the file is loaded and you are presented with the *Use Form* Main Menu:

UseForm

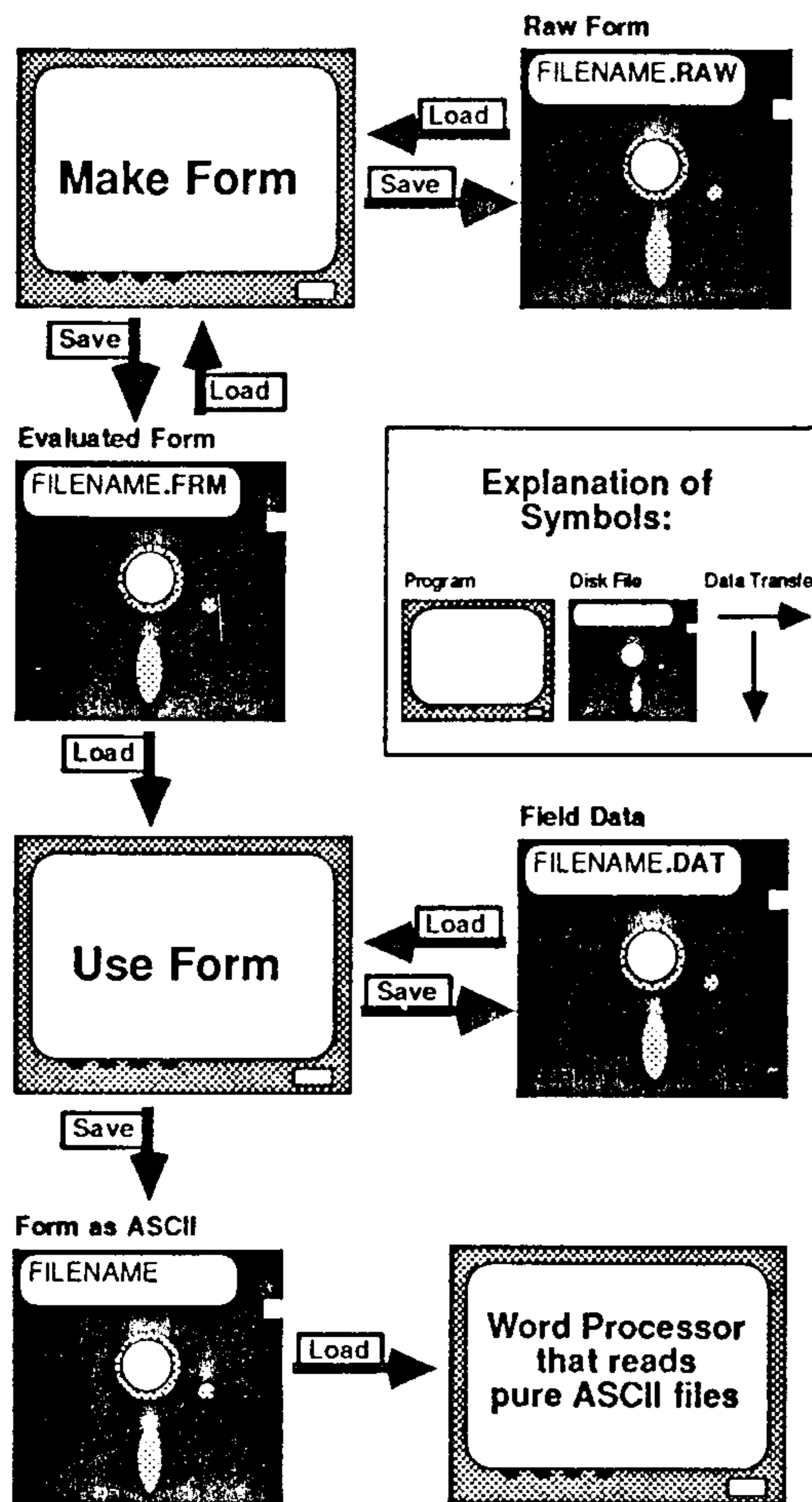
- 1) Fill Out Form
- 2) Clear Fields
- 3) Disk Routines
- 4) Printer Routines
- 5) Return to FormFlex Menu

To fill out the form, select option 1. The *Use Form* Fill Out Form option is designed specifically to enter and edit all the field data on a form. You cannot access or change any of the field names or any other text in the form—that can only be done from *Make Form*.

The program then displays your form, with all fields blank. The cursor is placed in the first field on the form, ready for

Figure 4.

FormFlex File Structure



Note: See next page for the appropriate Control Capsule for your machine.



SPECIAL HINTS

Making an IBM FormFlex Work Disk

1. Place your DOS master disk (hereafter referred to as the DOS disk) in drive A: and turn on the power to your system.

2. Enter the command **FORMAT B: /S /V**

3. The computer will ask you to place a blank disk into drive B: to be formatted. Ensure that the blank disk is in the drive and then press [Enter]. After formatting, you will be asked for a volume name. Enter the name **FORMWORK**. Then, you will be asked if you want to format another. Respond "No" to this prompt to return back to DOS.

4. Place the HCJ Volume 2 disk into drive A: and your newly formatted FORMWORK disk in drive B: (If you only have one disk drive, when you enter each of the following commands, you will be prompted to insert disks for drive A: and drive B: in turn—the HCJ disk will be the disk for A: and the FORMWORK disk will be the disk for B:)

5. Enter the following commands:

COPY A:FORMFLEX.COM B:

COPY A:MAKEFORM.CHN B:

COPY A:USEFORM.CHN B:

COPY A:COSTCOMP.FRM B:

COPY A:COSTCOMP.FLD B:

6. After the last file (COSTCOMP.FLD) has been copied, remove both disks from the system.

Label the new disk as FORMWORK. The new disk that you have created can now be used to boot your system. To run *FormFlex*, enter FORMFLEX at the A> prompt.

File Name Extensions

The IBM version uses the following file name extensions to differentiate the data files created by *FormFlex*:

A raw form	.RAW
An evaluated form	.FRM and .FLD (See below)
Field data	.DAT
Form saved as ASCII	.TXT

Two Files In One

When the IBM version saves an evaluated form, it saves the form as two separate files. One file is saved with the FRM extension and one with the FLD extension. The first file (FRM) comprises the basic layout of the form. The second file (FLD) contains all the field specifications such as the field type and field length. If you want to copy an evaluated form file from DOS, you must copy both the FRM and FLD files. To copy an evaluated form using *FormFlex*, simply load the evaluated form from *Make Form's* Disk Routines menu and then save it off to a separate disk.



SPECIAL HINTS

Making a TI FormFlex Work Disk

Format a disk giving it the name FORMWORK using the Disk Manager command module. Then use the Copy Files option to copy the following files to the newly formatted disk.

FORMFLEX
MAKEFORM
USEFORM
EDITOR_OBJ
COSTCOMP_F

File Name Extensions

The TI version uses the following file name extensions to differentiate the data files created by *FormFlex*:

A raw form	_R
An evaluated form	_F
Field data	_D
Form saved as ASCII	-none used-

Incorporating HCJWord

This Volume's TI Focus presents a word processor called *HCJWord*. This word processor is the perfect companion for *FormFlex*, because a form that is saved as ASCII from *Use Form's* Disk Routines' menu can be directly loaded, edited, and printed by *HCJWord*. Also, raw forms saved from *Make Form's* editor are compatible with *HCJWord*. Just remember to add the _R extension to the filename when loading a raw form into *HCJWord*.

CONTROL CAPSULE

FormFlex



Apple Make Form's "Edit Form" mode:

KEY	FUNCTION
Open A	Enter an Alpha field
Open U	Enter an Uppercase Alpha field
Open E	Enter an Edit Characters field
Open N	Enter a Numeric field
Open F	Enter a Formula field
Open D	Enter a decimal place for a Numeric or Formula field
TAB	Tab right 8 spaces
Open TAB	Tab left 8 spaces
RETURN	Move down one line and all the way to the left
Open I	Insert a character
Open CSR ↑	Page up
Open CSR ↓	Page down
Open \	Insert a line
Open Delete	Delete a line
Open S	Save raw form
Open L	Load raw form
Open P	Print raw form
Open C	Erase form
Escape*	Exit Edit Form
CTRL X	Exit Edit Form (UnEnhanced Ile)

Apple Use Form's "Fill Out Form" mode:

KEY	FUNCTION
RETURN	Move to next field
CSR ↓	Move to next field
CSR ↑	Move to previous field
Open CSR ↑	Move to first field
Open CSR ↓	Move to last field
Open F	Perform calculations
Open Delete	Clear current field
Open C	Clear all fields
Open P	Print form
Open O	Print fields only
Open S	Save field data
Open L	Load field data
Escape*	Exit Form
CTRL X	Exit Form (UnEnhanced Ile)

* Do NOT use the Escape key on an unEnhanced Apple Ile.



Atari Make Form's "Edit Form" mode:

KEY	FUNCTION
CTRL A	Enter an Alpha field
CTRL U	Enter an Uppercase Alpha field
CTRL E	Enter an Edit Characters field
CTRL N	Enter a Numeric field
CTRL F	Enter a Formula field
CTRL D	Enter a decimal place for a Numeric or Formula field
TAB	Tab right 5 spaces
SHIFT TAB	Tab left 5 spaces
RETURN	Move down one line and all the way to the left
CTRL O	Page up
CTRL P	Page down
CTRL L	Page left
CTRL ;	Page right
SHIFT INSERT	Insert a line
SHIFT DELETE	Delete a line
CTRL M	Load raw form
CTRL ,	Save raw form
CTRL .	Print raw form
SHIFT CLEAR	Erase form
ESC	Exit Edit Form

Atari Use Form's "Fill Out Form" mode:

KEY	FUNCTION
RETURN	Move to next field
CSR ↓	Move to next field
CSR ↑	Move to previous field
CTRL O	Move to first field
CTRL P	Move to last field
CTRL C	Perform calculations
SHIFT CLEAR	Clear current field
CTRL M	Load field data
CTRL ,	Save field data
CTRL .	Print form
CTRL F	Print fields only
ESC	Exit Form



C-64 Make Form's "Edit Form" mode:

KEY	FUNCTION
CTRL A	Enter an Alpha field
CTRL U	Enter an Uppercase Alpha field
CTRL E	Enter an Edit Characters field
CTRL N	Enter a Numeric field
CTRL F	Enter a Formula field
CTRL D	Enter a decimal place for a Numeric or Formula field
CTRL I	Tab right 5 spaces
RETURN	Move down one line and all the way to the left
f1	Page up
f7	Page down
f3	Page left
f5	Page right
f2	Insert a line
f4	Delete a line
f6	Save raw form
f8	Load raw form
CTRL P	Print raw form
SHIFT CLR	Erase form
CTRL X	Exit Edit Form

C-64 Use Form's "Fill Out Form" mode:

KEY	FUNCTION
RETURN	Move to next field
CSR ↓	Move to next field
CSR ↑	Move to previous field
f1	Move to first field
f7	Move to last field
f3	Perform calculations
SHIFT CLR	Clear current field
f5	Clear all fields
f2	Print form
f4	Print fields only
f6	Save field data
f8	Load field data
CTRL X	Exit Form



IBM Make Form's "Edit Form" mode:

KEY	FUNCTION
Ctrl A	Enter an Alpha field
Ctrl U	Enter an Uppercase Alpha field
Ctrl E	Enter an Edit Characters field
Ctrl N	Enter a Numeric field
Ctrl F	Enter a Formula field
Ctrl D	Enter a decimal place for a Numeric or Formula field
Tab	Tab right 5 spaces
Shift Tab	Tab left 5 spaces
Enter	Move down one line and all the way to the left
Pg Up	Page up
Pg Dn	Page down
Ctrl Pg Up	Move to top of form
Ctrl Pg Dn	Move to bottom of form
F1	Insert a line
F2	Delete a line
F6	Save raw form
F7	Load raw form
F8	Print raw form
F10	Erase form
Esc	Exit Edit Form

IBM Use Form's "Fill Out Form" mode:

KEY	FUNCTION
Enter	Move to next field
CSR ↓	Move to next field
CSR ↑	Move to previous field
Pg Up	Move to first field
Pg Dn	Move to last field
F1	Perform calculations
F2	Clear current field
F10	Clear all fields
Esc	Exit Form



TI Make Form's "Edit Form" mode:

KEY	FUNCTION
Ctrl A	Enter an Alpha field
Ctrl U	Enter an Uppercase Alpha field
Ctrl E	Enter an Edit Characters field
Ctrl N	Enter a Numeric field
Ctrl F	Enter a Formula field
Ctrl D	Enter a decimal place for a Numeric or Formula field
ENTER	Move down one line and all the way to the left
FCTN 1	Delete characters
FCTN 2	Insert characters
FCTN 3	Delete line
FCTN 4	Roll down
FCTN 5	Next window
FCTN 6	Roll up
FCTN 7	Tab
FCTN 8	Insert line
CTRL 1	Save raw form
CTRL 2	Load raw form
CTRL 3	Print raw form
CTRL 4	Erase form
FCTN 9	Exit Edit Form

Entering Formulas or Edit Characters:

FCTN X	Move to next line
FCTN E	Move to previous line
ENTER	Accept entry

TI Use Form's "Fill Out Form" mode:

KEY	FUNCTION
ENTER	Move to next field
FCTN E	Move to previous field if you are <i>not</i> on the first field
FCTN E	Calculate Formula fields if you <i>are</i> on the first field
FCTN X	Exit Form



Re-Weaver

Re-Weave your WordWeave files your way and print out a customized masterpiece.

So, with the help of *WordWeave*, you've created the ultimate who-dun-it, complete with a two-bit, unshaven, back-alley gumshoe. It's packed with intrigue, adventure, and romance. And after hours of experimentation on how the story should flow, you've found the perfect path (or trail). Now, how about making a final printout so that you can give a copy to your friends, or maybe even to your publisher?

Although, *WordWeave* does provide a Print option, it doesn't print the pages in the order in which the story is read. Besides that, the printout is single-spaced and only as wide as your computer's screen. Ideally, you want a printout with the line spacing and margin settings that *you* select. *Re-Weaver* to the rescue.

Re-Weaver is a trail-marker/text-formatter for use with *WordWeave* text files. With it, you can print your story using the trail that you define, and the printer formatting options that you select. Each trail that you create can be saved for later use or editing. The printer formatting options (Print Parameters) include margin settings, line spacing, lines per page, page breaks, page numbers, and selective printing of your *WordWeave* Header and/or Branch text. Also, when you save your trail, the Print Parameters are automatically saved along with it.

Note: Before you can use this program, you must have created a *WordWeave* data disk with at least one page of text on it. We also recommend that you prepare a separate formatted disk to save your trail and Print Parameters. Do *not* use the *WordWeave* disk for saving trail data. Doing so will limit the number of pages in your *WordWeave* file and possibly destroy valuable data.

The Main Menu

The program's Main Menu presents you with the following 6 options:

1. Make Trail
2. Read Trail
3. Save Trail
4. Load Trail
5. Print Trail
6. Exit

1. Make Trail

A *WordWeave* story can be read in almost any order, branching from one page to another (i.e., page 1. . . page 7. . . page 2. . .). When you choose this option, you are marking a trail through your story that determines the order in which the pages will be printed. Before choosing this option, you must place a *WordWeave* data disk into the disk drive. If you don't, an error message results, and you'll be brought back to the Main Menu.

Make Trail mode is identical to *WordWeave*'s Read mode. You can display the Branches menu, and select an option, go back to the previous page, or return to the Main Menu. There is a different keypress for each of these options. See your computer's Control Capsule for a complete list.

As you read through your story, the computer keeps track of the trail that you are blazing. Every time that you select a new branch, that decision is stored in the computer's memory. If you make a mistake, and choose the wrong branch, you can correct it: By using the option to return to the previous page, you can force the computer to forget that you were ever at the page from which you returned. This provides a very useful method for backtracking through your trail.

2. Read Trail

Here, you can test-drive the trail that you've created with the Make Trail option. Say that you load a trail that was saved a few weeks ago and you want to see how the story reads. To do this, simply choose the Read Trail option and you can "walk" down the story's trail, page by page. Again, you must insert your *WordWeave* disk into the drive before selecting this option.

In this mode, you can't decide which Branch to take—that choice was determined when the trail was created with the Make Trail option. So, to move through the story, just press the [SPACE BAR]. You can also go back to the previous page or return to the Main Menu. See your computer's Control Capsule for these keypresses.

If, while you are reading through your trail you find a mistake, there is a way to correct it. By pressing [CTRL] E from Read Trail mode, you can enter Make Trail mode and edit the trail from the current page. From here on out, you are in Make Trail mode. Any trail that was ahead of you has been

Note: The Re-Weaver program is an enhancement to WordWeave, which appeared in HCJ Volume 1. WordWeave is a specialized word processor for creating a branching text stream—be it interactive fiction or a host of multi-path text applications for such things as adventure games, user's manuals, workbooks, and tutorial materials.

forgotten by the computer. You may still, however, backtrack your trail.

3. Save Trail

Once you have found a trail that traverses well through your story, you may save it to disk. Be sure to save your trail on a disk other than your *WordWeave* data disk. When you select this option, the computer prompts you for a file name. If you stumble into this option by mistake, simply press [RETURN] or [ENTER] without entering a file name, and you will return to the Main Menu.

When entering a file name, we suggest that you use the title of your story. This way, you will not be confused as to which trail file corresponds with which story. After entering the file name, the computer saves your trail to disk, complete with Print Parameters (described later).

4. Load Trail

To load a trail that has been previously saved, select this option. The program will prompt you for a file name, and then load the selected trail file. If you do not enter a file name, you are returned to the Main Menu.

5. Print Trail

This option presents you with these three choices:

1. Change Print Parameters
2. Print it
3. Return to the Main Menu

Change Print Parameters

At the bottom of the Print Trail menu screen is a list of your Print Parameters and their defaults. This list is reproduced below:

Left margin: 5
Line spacing: 2
Page breaks: No
Header text: No

Right margin: 75
Lines per page: 55
Page numbers: Yes
Branch text: No

Choosing the Change Print Parameters option allows you to change the above parameters to your liking. The left and right margins can be set anywhere within 79 characters as long as the left margin is at least 5 characters less than the right margin. (For example, you may not have a left margin of 5 and a right margin of 9.) A left margin setting of 5 indents all text 5 spaces from the left of the page. A right margin setting of 75 keeps the characters 5 spaces from the right of the page on a 80-column printer. Line spacing may be 1 (single spaced) or 2 (double spaced). You can also choose the number of lines that you want printed per page. Lines per page can vary from 10 to 64.

There are probably places within your text, like the beginning of a new chapter, that you would like to have printed at the top of a new page. To do this, you need to do two things: First, you must use *WordWeave's* Write mode to go back into your story and add the caret character ^ (the up arrow character on the Commodore 64), every place that you want the text to begin a new page. This character is used as a control character, informing *Re-Weaver's* print routine to begin a new page. The second thing that you must do is say "yes" to the Print Parameters' page-break option. If you do not, the caret will be printed out like any other character.

If you want your pages to have page numbers printed at the bottom, say "yes" to the page numbers parameter. These numbers will be the page numbers of your trail (corresponding to the number of paper pages created)—not necessarily the page numbers of *WordWeave* screens within the trail.

When you are writing a *WordWeave* story, you can enter a Header for each page of text. If you want this Header to appear on the *Re-Weaver's* printout, say "yes" to the Header text option. Similarly, *WordWeave* allows one line of text to

CONTROL CAPSULE			
Re-Weaver			
Make Trail Mode		Read Trail Mode	
KEY	FUNCTION	KEY	FUNCTION
1	Choose Branch 1	SPACE	Go to next page
2	Choose Branch 2	ESC	Go to previous page
3	Choose Branch 3	CTRL X	Exit to Main Menu
4	Choose Branch 4	CTRL B	Toggle Branches menu
ESC	Go to previous page	CTRL E	Enter Make Trail mode
CTRL X	Exit to Main Menu		
CTRL B	Toggle Branches menu		

CONTROL CAPSULE			
Re-Weaver			
Make Trail Mode		Read Trail Mode	
KEY	FUNCTION	KEY	FUNCTION
1	Choose Branch 1	SPACE	Go to next page
2	Choose Branch 2	ESC	Go to previous page
3	Choose Branch 3	CTRL X	Exit to Main Menu
4	Choose Branch 4	CTRL B	Toggle Branches menu
ESC	Go to previous page	CTRL E	Enter Make Trail mode
CTRL X	Exit to Main Menu		
CTRL B	Toggle Branches menu		

CONTROL CAPSULE			
Re-Weaver			
Make Trail Mode		Read Trail Mode	
KEY	FUNCTION	KEY	FUNCTION
F1	Choose Branch 1	SPACE	Go to next page
F3	Choose Branch 2	←	Go to previous page
F5	Choose Branch 3	CTRL X	Exit to Main Menu
F7	Choose Branch 4	CTRL B	Toggle Branches menu
←	Go to previous page	CTRL E	Enter Make Trail mode
CTRL X	Exit to main menu		
CTRL B	Toggle Branches menu		

CONTROL CAPSULE			
Re-Weaver			
Make Trail Mode		Read Trail Mode	
KEY	FUNCTION	KEY	FUNCTION
F1	Choose Branch 1	SPACE	Go to next page
F2	Choose Branch 2	ESC	Go to previous page
F3	Choose Branch 3	F7	Exit to Main Menu
F4	Choose Branch 4	F8	Toggle Branches menu
ESC	Go to previous page	CTRL E	Enter Make Trail mode
F7	Exit to Main Menu		
F8	Toggle Branches menu		

describe a Branch in the Branch menu. If you want this line to appear on *Re-Weaver's* printout, say "yes" to the Branch Text option. The Branches that are printed are the ones that are selected in Make Trail mode.

Print It

Choosing this option requires that you already have a trail in memory. You can achieve this by loading a trail from disk, or choosing the Make Trail option from the Main Menu. Make sure that the *WordWeave* disk is in the drive before choosing this option.

This option prints your *WordWeave* story, using the trail and Print Parameters that you have selected. For best results, align the printer paper's perforation just above the print head.

The Rules

Because your computer's screen width will not always match the width specified by the Print Parameters' margin settings, *Re-Weaver* performs automatic wordwrap whenever it prints a trail. To wrap words correctly, *Re-Weaver* follows a simple set of rules. You should be aware of these rules when using *WordWeave*, so that *Re-Weaver* will correctly format your text. When reading these rules, keep in mind that there are two types of pages: There's the *WordWeave* page that is created by the *WordWeave* program, and there's the page of formatted text that is produced by *Re-Weaver's* print routine. You see, one *Re-Weaver* page may consist of several *WordWeave* pages, depending on the trail chosen and the Print Parameters used. The following figures (labeled Rules 1 through 6) illustrate each rule and their function.

Rules 1 and 2 take care of most of the word wrapping. Pay particular attention to Rule 2. Because of Rule 2, you must never let a word *break* (a word that wraps from one screen line to the next) when entering text with *WordWeave*. If you do, *Re-Weaver* will consider the broken word to be two separate words and print them as such.

CONTROL CAPSULE



Re-Weaver

Make Trail Mode

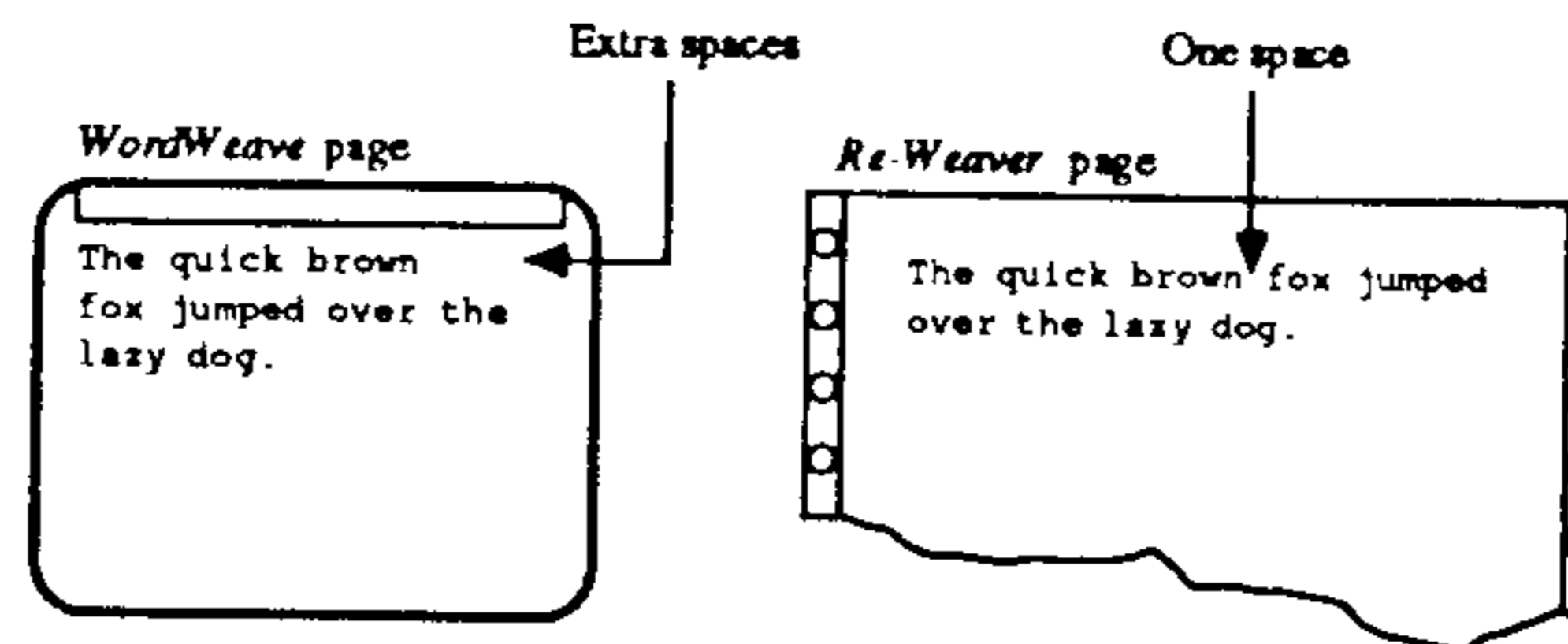
KEY	FUNCTION
1	Choose Branch 1
2	Choose Branch 2
3	Choose Branch 3
4	Choose Branch 4
FCTN 9	Go to previous page
CTRL X	Exit to Main Menu
CTRL B	Toggle Branches menu

Read Trail Mode

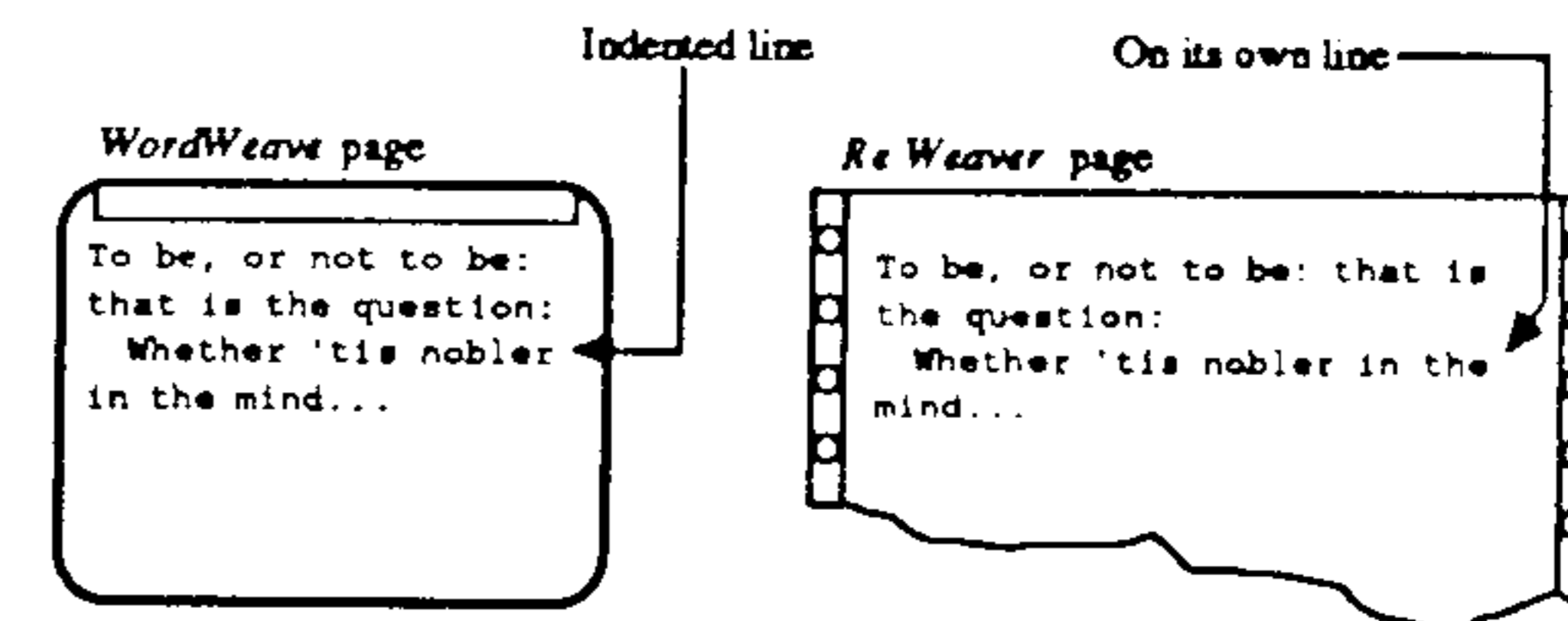
KEY	FUNCTION
SPACE	Go to next page
FCTN 9	Go to previous page
CTRL X	Exit to Main Menu
CTRL B	Toggle Branches menu
CTRL E	Enter Make Trail mode

The Rules for Text Formatting in *Re-Weaver*

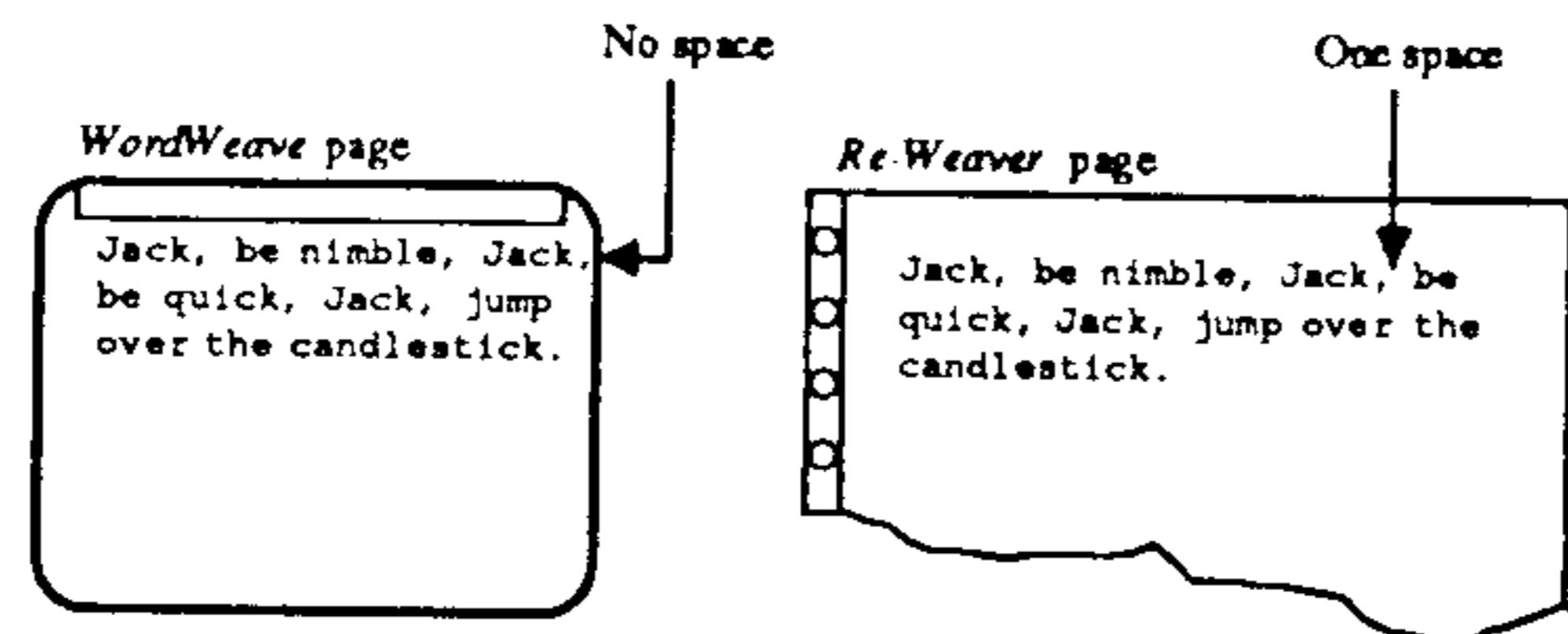
Rule 1: Spaces found at the end of *WordWeave* page screen lines are considered to be one space. This is done so that the first word on the next line can be pulled up and attached to the previous line correctly.



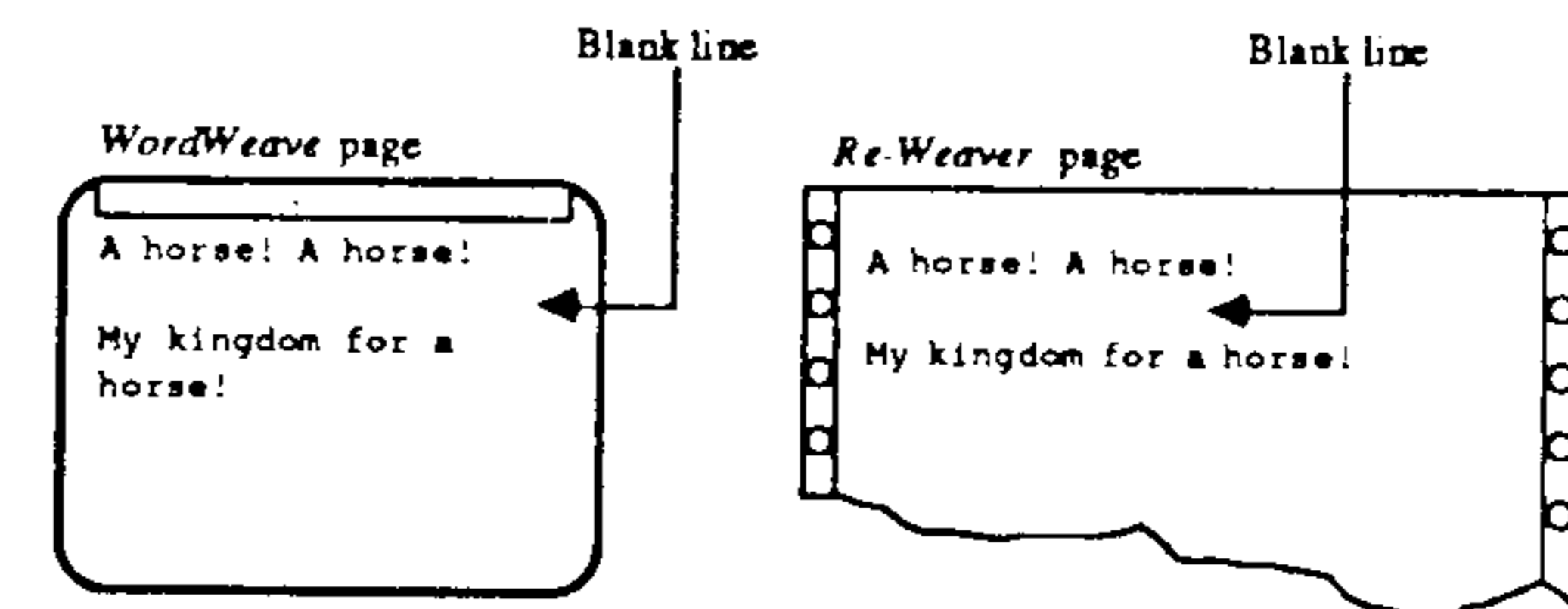
Rule 4: All indented lines of text found in a *WordWeave* page are considered to be the beginning of a paragraph. These lines will always begin on a new line within the *Re-Weaver* page.



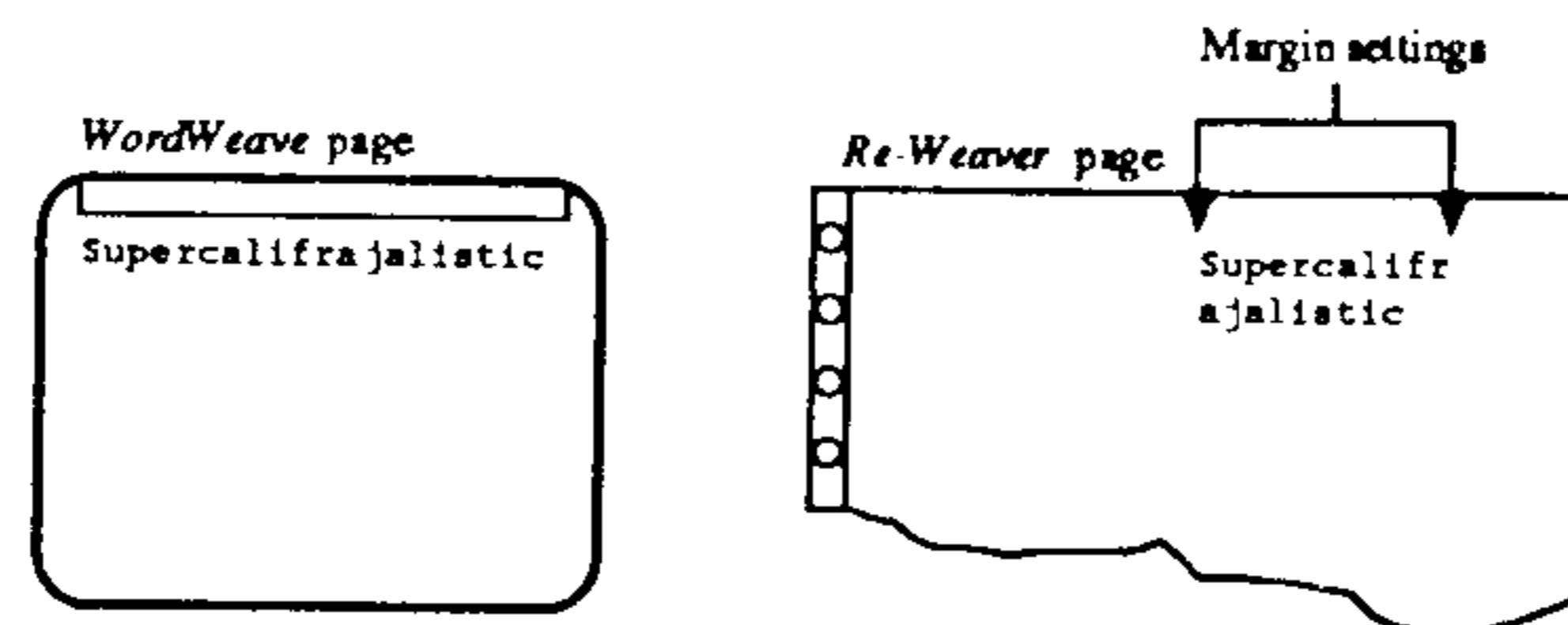
Rule 2: If there isn't a space at the end of a *WordWeave* page screen line, one is added. This is done so that if a word is pulled up from the next line, it will have a space between it and the previous word.



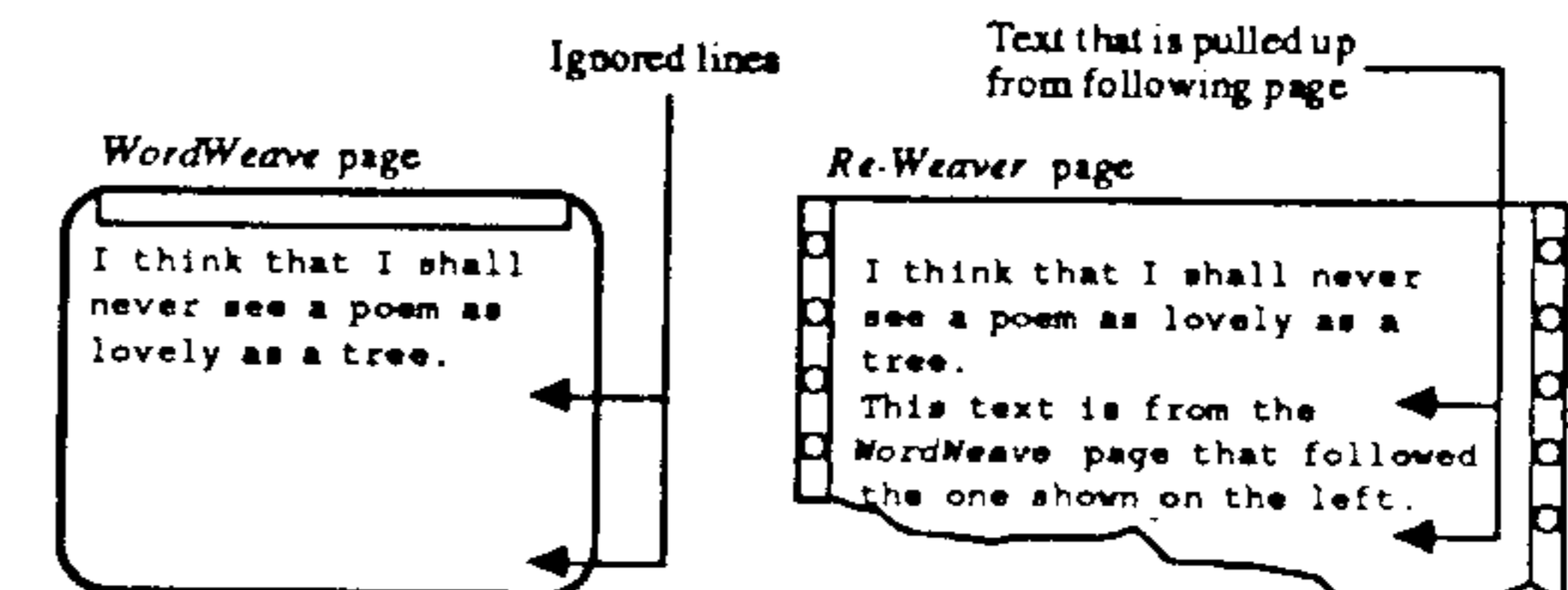
Rule 5: All blank screen lines found within the text of a *WordWeave* page are printed as blank lines on the *Re-Weaver* page.



Rule 3: All words that are too long to fit within the margin settings of a *Re-Weaver* page are broken, not hyphenated.



Rule 6: All blank screen lines found at the end of a *WordWeave* page are ignored so that the following *WordWeave* pages of text can be pulled up.



A challenging, yet very easy-to-learn educational brain-game based on the concepts of set theory.

Set theory is important in a structured approach to mathematics, and has consequently become an integral part of many computer programming languages (e.g., Pascal, C, etc.). The *Upsets* program is an educational game that offers you an opportunity to learn about and practice the basics of set theory, and then apply your understanding in either a solitaire or two-person game. If you are new to the concepts, terms, and symbols used in set theory, first see our sidebar "Getting Set-tled For Upsets"

The game is designed around a deck of cards with four different colored dots. The four colors on the cards are the elements used to describe which cards are in the set. For example, a set may comprise all cards with a blue dot and be identified by the single element BLUE. In Figure 1, this would include cards 1, 3, and 4. Another set could be all cards which have either blue dot or a red dot using the union function. The set expression would be:

$$\text{BLUE} \cup \text{RED}$$

This set would include cards 1, 3, 4, 5, and 6 in Figure 1.

Another set could be described using the intersection function to include only cards having both a blue dot and a red dot. The expression for this set would be:

$$\text{BLUE} \cap \text{RED}$$

This set would include cards 1 and 3 in Figure 1.

When the minus sign appears in a set expression, it will indicate the exclusion of one or more colors from the set being defined. An example using the difference connective is:

$$\text{BLUE} - \text{RED}$$

In this example, only card 4 would be selected from Figure 1. This means that you take the set of all cards with a blue dot, then remove any cards having a red dot.

Upsets displays up to four colors on each of the cards. You may use any or all of these four colors with any of the three connectives to form a set expression or set name. The set you describe could include all, some, or none of the cards—depending on the colors, connectives, and position of the parentheses.

Get Ready, Get Set . . .

The first menu in the program allows you to select one of three different games:

- (1) Practice
- (2) Solitaire
- (3) Challenge

When you select option 1 for a Practice session, the computer does not keep score, or limit the game length. In this mode, you are able to create set names and watch the results. This provides an excellent way to discover the logic rules that govern set names. The computer places the cards on the screen randomly from a "deck" of fifteen cards. Note that a blank card is never one of the cards in the game, because it would represent the "null" set. According to the rules of set theory, the null set is included in *all* sets and, because a blank card would *always* be selected, it is not used in the game of *Upsets*.

Option 2, Solitaire, lets you play *Upsets* by yourself. Here, you express sets that include as many cards as possible on your side of the screen (the left), while selecting as few cards as possible from the right side of the screen. You make all the plays in Solitaire, but the computer keeps score, giving you 10 points for each card included in the set on your side of the screen, and giving your non-playing "opponent" 10 points for each card that your expression selects on the right. You can win the game in two ways: (1) by selecting *all* of your cards and *fewer* than all of your opponent's cards in any one turn, or (2) by having the highest score after 10 turns.

In Option 3, Challenge, two players take turns defining the set name. One player uses the cards on the left side of the screen, while the other uses the cards on the right side. After each player takes a turn, the computer evaluates the new set, and each player receives 10 points for each card in the set.

After you select a game option, a prompt asks you to enter the players' names (one or two names, depending on the game type selected). Next, you select the number of cards you would like to work with. You can have from 3 to 6 cards displayed on each side of the screen. The more cards you select, the more complicated the game, and the greater the possible score.

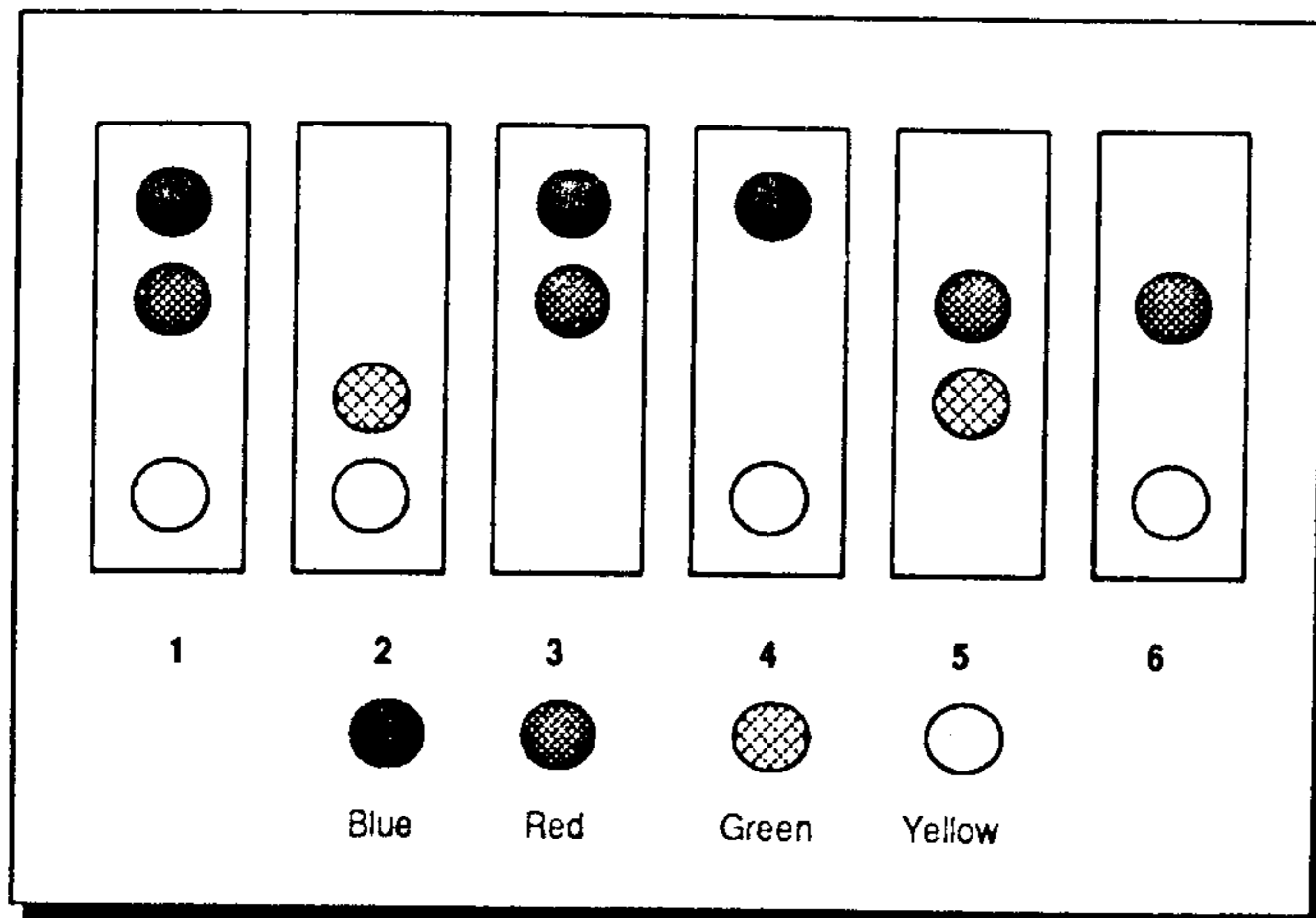
Finally, a prompt asks how many connectives (from 1 to 3) you would like to use in the set name.

Select Level Of Expression:

- 1) One Connective
- 2) Two Connectives
- 3) Three Connectives

Figure 1.

A sample set of "cards" in a game of Upsets. Note that the colors of the cards in your computer's version may be different from those indicated below.





Getting Set-tled For Upsets

Set theory is the branch of mathematics concerned with the definition and relationships between the grouping of objects, numbers . . . and well, *anything*. We say that a particular set "contains elements," very much like a box contains items. For example, a standard deck of 52 playing cards is a set of all the cards after you discard the "set" consisting of jokers and any other non-face or number cards. We can define a grouping or set that contains all the hearts. We say this set contains all cards with red hearts on them. We may do the same for cards having the red diamond, the black spade, and the black club.

We can define other sets by using "connectives" to form relationships between different sets. The basic connectives (those used in our *Upsets* program) are: **union**, **intersection**, and **difference**. Figures A, B, and C show three diagrams (called Venn diagrams) that graphically demonstrate how these functions work. The two circles, X and Y, represent two different sets. The black areas show the set created by the function.

The **union** function (see Figure A) is used to form a larger set containing elements of two smaller sets. This function is expressed by the \cup symbol and is similar to the OR function used in BASIC. Taking our deck of cards again, we can define a set containing all red cards as being the union of hearts and

diamonds, or expressed symbolically:

$$\text{red cards} = \text{hearts} \cup \text{diamonds}$$

Obviously, this function combines two of our first four sets into another set of *all* red cards.

The **intersection** function is expressed by the \cap symbol and is similar to the BASIC **AND** function (see Figure B). It creates a set by selecting elements that are contained in both sets. The fewer elements that are *shared*, the more *exclusive* the set becomes; e.g., the set of cards formed by the intersection of all aces and all red cards is a set of only two cards—the ace of hearts and the ace of diamonds.

$$\text{red aces} = (\text{red cards}) \cap (\text{aces})$$

The **difference** function represented by a minus sign allows for the exclusion of certain elements from a set (see Figure C). Using the deck of 52 cards, we can form a selection of all cards not having numbers—the set of all face cards and aces. Symbolically we write:

$$(\text{face cards} \cup \text{aces}) = (\text{all cards}) - (\text{cards with numbers})$$

These functions can be combined in set expressions, just like mathematical expressions, and parentheses are used to designate the order of evaluation.

Venn Diagrams

Figure A.

This diagram represents the union function—similar to the BASIC OR function.

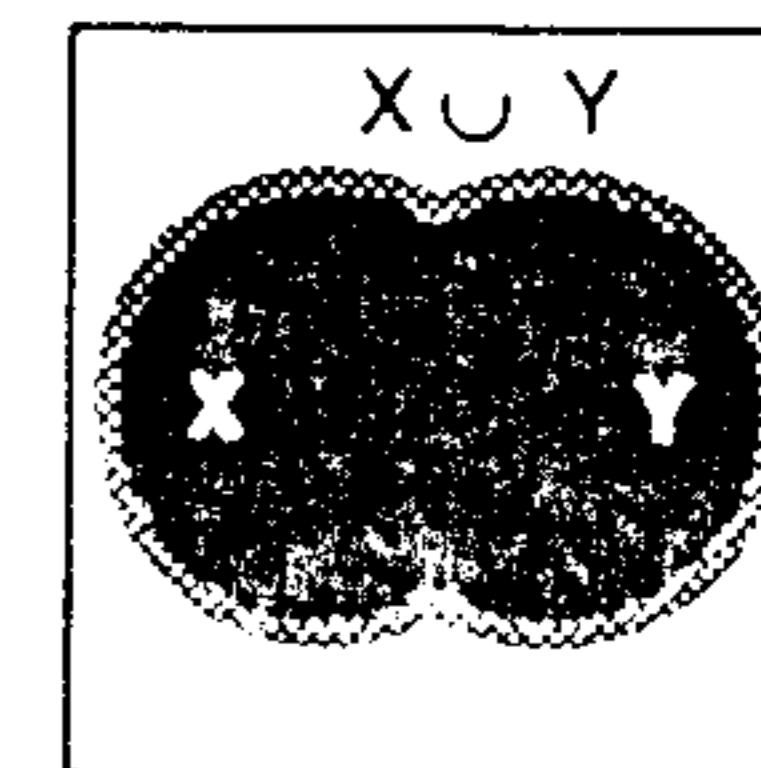


Figure B

The intersection function is like the BASIC AND statement.

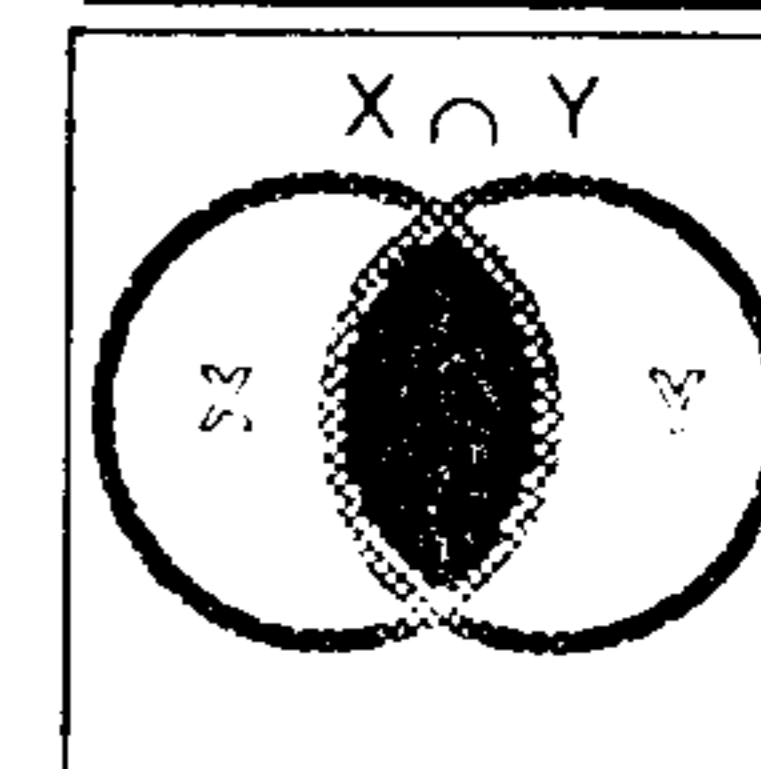
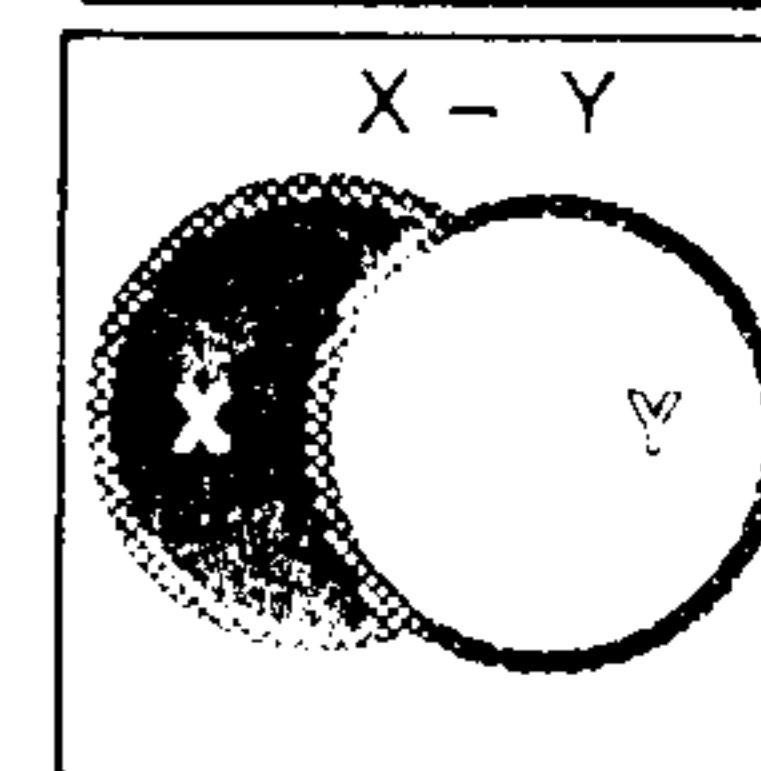


Figure C.

Here, the difference function excludes elements from set X that also occur in set Y.



This will determine the length and complexity of the set name. With only one connective, you are limited to simple set expressions like those in the examples above (e.g., **BLUE** \cap **RED**). And with only one connective you may use any two (but *only* two) of the four colors in any expression.

Two connectives allow a more complex name such as: (**BLUE** \cup **RED**) \cap **GREEN**. Selecting three connectives expands the set name to the most complex level available in *Upsets*. It also allows the switching of parentheses to change the order in which the expression is evaluated—expanding the number of strategic tools available.

The Playing Screen

After selecting all of the options described, the playing screen appears (see Figure 2). Depending upon which game you've chosen, 3 to 12 cards are displayed across the top of the screen. No two cards are ever alike. When the computer evaluates a set expression, it indicates which cards are in the designated set by placing a mark above each of the cards that satisfies the set expression.

Below the cards are 2 small boxes labeled **Round** and **Turns**. The box labeled **Round** is activated only when you play the Challenge option. In this option, you play 4 rounds. A round is over when one player has created a set that includes

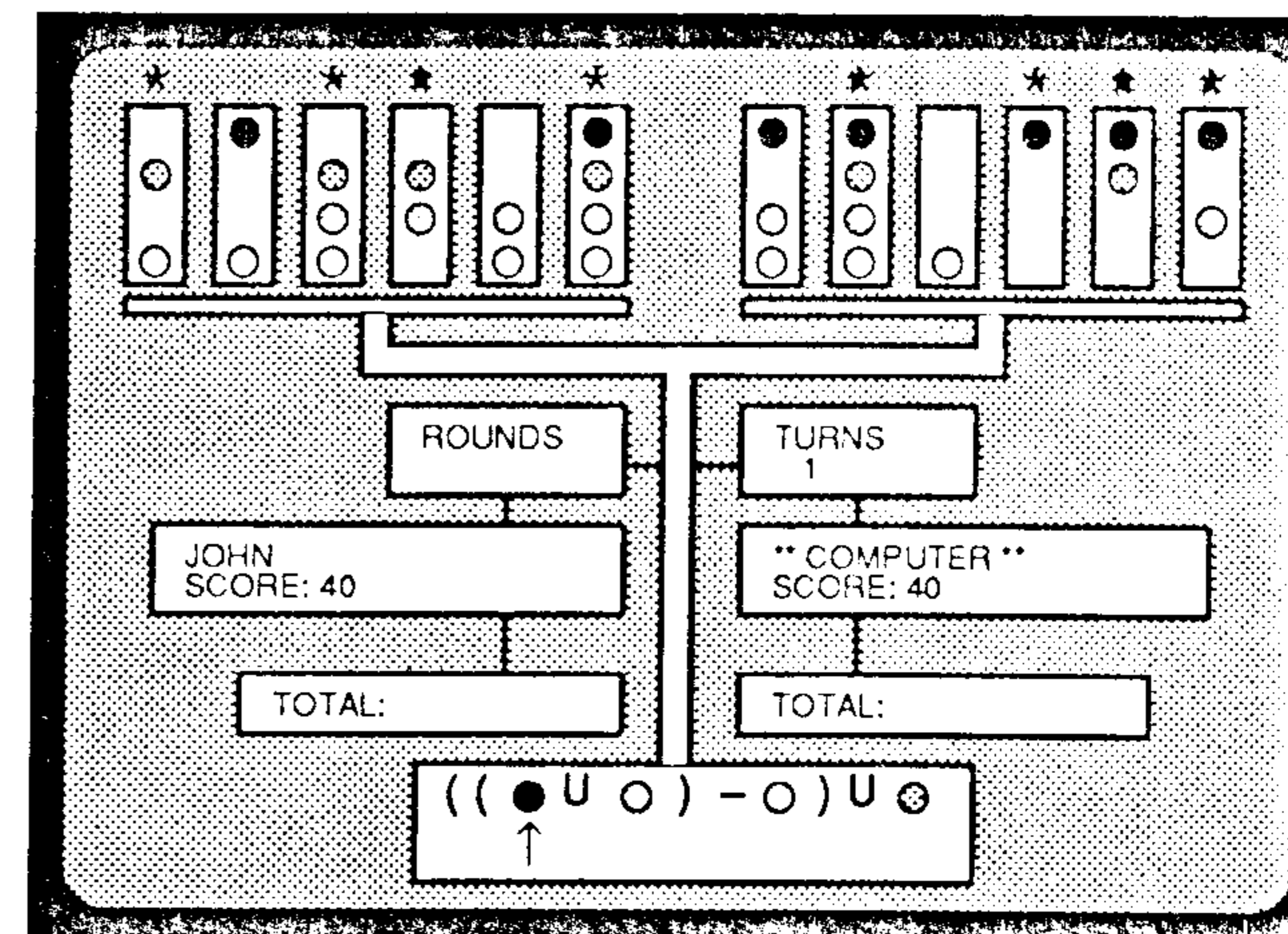
all of the cards on either one or both sides of the screen, or after 20 turns. The player with the highest total score after 4 rounds wins the game. In rounds 1 and 3, the player on the left side of the screen goes first; in rounds 2 and 4, the player on the right side goes first.

The **Turns** box indicates the current turn, and is active only during the Solitaire or Challenge options. In Solitaire, the length of a game is 10 turns. In Challenge, the length of each round is 20 turns. A turn is over after *both* players have made a change to the current set expression.

Directly below the **Round** and **Turns** boxes are the 2 **Score** boxes each labeled with the player's name. In Solitaire, the score box on the left side always holds the player's score; the score box on the right contains the score of your non-playing opponent. In a Challenge game, the name of each player appears in the box with that player's score. The score displayed here is only for the current round. At the start of each round, each **Score** is added to the associated **Total** box, and the **Score** box is reset to zero.

Figure 2.

This simulated screen depicts a solitaire game where the set expression describes 4 cards from each side.



Express Yourself

In the bottom center portion of the screen is the current set name. At the beginning of a game or round, the name is empty except for parentheses:

((. . .) . .) . .

Here, we are using periods to indicate what appears on the screen as empty character positions. Below this empty set name is a cursor. The current player moves the cursor to any position in the set name and, by pressing the [SPACE BAR], selects *one* legal connective symbol or color dot for that position in the name. To toggle through all of the legal symbols for that position, simply continue pressing the [SPACE BAR]. Once you have displayed the desired symbol, both your selection and your turn are completed by pressing [ENTER] or [RETURN]. If you move away from a position without pressing [ENTER] or [RETURN], the original symbol appearing at that location will return.

Before a set name can be evaluated by the computer, all parts of it must be filled in. It needs to contain the number of connectives you chose at the beginning of the game *and* enough colored dots to evaluate the connectives. It may take several turns before the set name is completed for evaluation. A valid name with 1 connective may look like this:

((• ∩ •) . .) . .

(Periods represent spaces, the • character represents a colored dot, and ∩ ∪ - are the connectives.)

A set name with 2 connectives may look like this:

((• ∩ •) - •) . .

A set name with all 3 connectives would look like this:

((• ∩ •) - •) ∪ •

A set name can be filled out in any order, with each player limited to setting either a single dot color or a connective during a given turn. But until the set name is completely filled out and evaluated at least once, players can't *change* any of the connectives or dots. If all 3 connectives are being used, players also have the option of changing the order of parentheses.

Order Of Evaluation

The parentheses used in a set name, as described earlier, indicate the order the connectives are to be evaluated. The computer always evaluates that part of the expression within the innermost parentheses *first*, working its way outward to the outermost connective not contained within parentheses; or, it *begins* at the left and evaluates towards the right. The default position for parentheses will always appear as in the

first example shown above, which causes connectives to be evaluated from left to right. The following examples will illustrate the effect parentheses have on a 3-connective set name:

((B ∩ R) - G) ∪ Y

(Here, the letters B, R, G, and Y represent the colors Blue, Red, Green, and Yellow.) This set would include any cards containing either (1) a yellow dot, or (2) both a blue dot and a red dot, but not a green dot.

(B ∩ R) - (G ∪ Y)

This set name is the same as the one above—*except* for the position of the parentheses. It now includes cards that contain both blue and red, but not green or yellow. Because there are only four colors, and no two cards can be identical, the set named above can include only one card.

To alter parentheses, just press P, and the parentheses change. This allows you to switch the parentheses to any of three possible settings; the set name at the bottom of the screen will be rewritten showing the new arrangement. Once you have the parentheses set to the desired arrangement, press [ENTER] or [RETURN] to fix the setting. This completes a player's turn and begins reevaluation of the set name and an updating of the score accordingly. If you decide that you don't want to change the position of the parentheses before you press [RETURN] or [ENTER], you can return the name to its original form by simply moving the cursor. This allows you to continue your turn to make a different change to the set name. You may make only *one* permanent change to the set name during your turn.

To quit, press [ESC] (or ← on the C-64 and [FCTN] 9 on the TI-99/4A). Whether you wish to simply introduce yourself to set theory, or use your talents of logic to develop effective game strategies, *Upsets* is definitely a real setup for challenging fun.

CONTROL TABLE	
<i>Upsets</i>	
KEY	FUNCTION
SPACE BAR	Change symbol
ENTER or RETURN	Accept symbol in expression
P	Alter parentheses
ESC*	Return to Main Menu
* FCTN 9 on TI-99/4A, ← on C-64	



File Verify

It's no fun when you lose an important disk file simply because you accidentally save a different file on top of it. This is, unfortunately, very easy to do. Many programs avoid this type of mistake by warning you when a file of the same name already exists on disk. If it exists, you are asked if you wish to replace the current file with the new one. Here, we offer two techniques for checking a disk for pre-existing files.

Both techniques presented here are subroutines written in Extended BASIC (see Listings 1 and 2 on this page). On your HCJ Volume 2 disk, these subroutines are saved as MERGE files under the file names VERIFY1 and VERIFY2. To load a MERGE file such as VERIFY1, enter the following: **MERGE DSK1.VERIFY1**

The first technique (see Listing 1) is the most versatile of the subroutines shown here. Not only can it check for data files (i.e., VAR 80), but it can also verify program files. To use this subroutine, set the variable **FL\$** equal to the device & filename of the file that you want to verify and **GOSUB 10000**. The subroutine returns the variable **ST**; it will be set to a 1 if the file exists, or a 0 if the file does not. For instance, to verify the existence this very subroutine on your HCJ disk, MERGE VERIFY1 into memory and execute the following from within a program: **FL\$="DSK1.VERIFY1" :: GOSUB 10000 :: PRINT ST :: END**

Assuming that your HCJ disk is in drive 1, variable **ST** should equal 1.

The way this subroutine works is simple. By **OPENING** a disk file without specifying a file name, we effectively open the disk's catalog. Now, we can read each file name in, one by one, until we find the file that we are looking for, or until we reach the end of the catalog. If the file is found, **ST** is set equal to 1. If the file is not found, **ST** is set equal to 0. Figure 1 lists the line annotations for this subroutine.

One problem with this approach is error handling. If this subroutine encounters an error, such as reading an unformatted disk, or the absence of a disk in the drive, all succeeding disk access will fail. The initial error can be caught by the **ON ERROR** command, but any following disk access will cause an error before even spinning the disk. Because the catalog is so vital to a disk's existence, you must tread carefully while it is open.

The second file-verify technique (see Listing 2) is not as versatile as the first, but it does catch disk errors without complications. This subroutine can verify all types of files except program and MERGE files. You call this subroutine the same way as the first: Set the variable **FL\$** equal to the device & filename of the file that you want to verify and **GOSUB 10000**. Again, upon returning, the subroutine returns the variable **ST** will be equal to a 1 if the file exists, or a 0 if the file does not.

This second subroutine uses a devious approach to verifying files. To verify files, it *purposely* creates I/O errors. Basically, the file being verified is open for input. If an error occurs while the file is open, we know that the file does not exist. If an error does not occur, the file must exist. Figure 2 lists the line annotations for this subroutine.

Because of its error handling abilities, our HCJ Feature program, *FormFlex*, uses the second technique for verifying files. By using this subroutine, *FormFlex* can ask a user if he wishes to replace pre-existing files when saving his form data.

Figure 1

Line Nos.	
10000-10050	Program header
10060	Open disk's catalog
10070	Get disk's name
10080	Read file name
10090	If file name is the same, set ST and exit
10100	If not end of catalog then continue reading disk, otherwise clear ST and exit
10110	Close catalog
10120	Exit subroutine

Figure 2

Line Nos.	
10000-10050	Program header
10060	Initialize ST to 0 and set error trapping
10070	Open file for input
10080	If no error occurs, set ST to 1
10090	Exit subroutine
10100	Return from error to line 10080

Listing 1

```

10000 ! FILE VERIFY1
10010 ! VERIFIES ALL TYPES OF FILES, BUT HAS NO ERROR CHECKING
10020 ! COPYRIGHT 1986
10030 ! HOME COMPUTING JOURNAL
10040 ! VERSION 2.0
10050 ! TI EXTENDED BASIC
10060 OPEN #1:SEG$(FL$,1,5),INTERNAL,RELATIVE,INPUT
10070 INPUT #1:A$,A,A,A
10080 INPUT #1:A$,A,A,A
10090 IF A$=SEG$(FL$,6,20)THEN ST=1 :: GOTO 10110
10100 IF A$<>" " THEN 10080 ELSE ST=0
10110 CLOSE #1
10120 RETURN
    
```

Listing 2

```

10000 ! FILE VERIFY2
10010 ! VERIFIES ONLY DATA FILES, BUT HAS ERROR CHECKING
10020 ! COPYRIGHT 1986
10030 ! HOME COMPUTING JOURNAL
10040 ! VERSION 2.0
10050 ! TI EXTENDED BASIC
10060 ST=0 :: ON ERROR 10090
10070 OPEN #1:FL$,INPUT :: CLOSE #1 :: ON ERROR STOP :: ST=1
10080 RETURN
10090 RETURN 10080
    
```

TI-Writer, look out!
Here comes HCJWord...

Need to jot down a few notes, write a letter to your aunt, or crank out a quick resume before a job interview? *HCJWord* may be just what you're looking for. *HCJWord* is an easy-to-use word processor that allows you to edit, save, load, and print a page of text quickly and painlessly. *HCJWord* requires Extended BASIC and 32K memory expansion. (It also helps to have a printer.)

To boot *HCJWord*, simply **OLD** the file **HCJWORD** from your HCJ Volume 2 disk and enter **RUN**. You may also select *HCJWord* from the *HCJ Director* program.

A Page Processor

HCJWord is what you might call a "page processor." You see, we specifically designed this program to process a page of text that is 60 rows by 80 columns. A page of these dimensions conveniently prints out on one 8-1/2" by 11" piece of paper. Admittedly, you should never attempt put together a 1000-page novel using this program, but for the more common job of roughing out one-page notes, *HCJWord* is the perfect choice.

presenting error messages and prompting the user for such information as file names and printer parameters.

To enter text, simply begin typing. There is no "edit" mode to enter, because you are always in it. Your cursor appears as a non-blinking black square on the screen. As you will notice, the cursor moves under the text, not over it. If while entering text your cursor reaches the edge of the screen, the edit screen will automatically shift its viewing window to display the section of the page that you are typing on. The same thing happens when you try to move the cursor off the screen in any direction—the edit screen automatically shifts its viewing window accordingly. When you have reached the end of a page line, you can either press [ENTER] or move the cursor down to get to the next line.

Text Manipulation

HCJWord provides all the standard editing features of the TI computer, and more. You can delete characters, insert characters, erase a line, insert a line, tab right, cursor in any direction, and erase the entire page of text. Also, the cursor obligingly squashes to half its normal height whenever the editor is in insert mode—a useful form of visual feedback. All these editing features are accessed via a keypress (see Control Capsule). In Figure 1, we have provided a cut-out keyboard strip that fits above the computer's keyboard. This strip allows for ready viewing of the various keystrokes that *HCJWord* has to offer. We suggest that you make a copy of Figure 1 for use as a keyboard overlay.

Saving, loading, and printing of text is also achieved through keypresses. [CTRL] 1 is for saving text, [CTRL] 2 is for loading text, and [CTRL] 3 is for printing text. Whenever you choose to save or load text, the computer prompts you for a file name. The file name must be preceded by the device (i.e., **DSK1**), or an error will occur. Similarly, whenever you choose to print your text, the computer prompts you to enter your printer parameters. To exit *HCJWord*, press [FCTN] 9. If you have made any changes to your text without saving them, *HCJWord* asks if you wish to save the text before quitting.

CONTROL CAPSULE			
HCJWord			
KEY	FUNCTION	KEY	FUNCTION
ENTER	Move down a line and left	FCTN 7	Tab
FCTN 1	Delete characters	FCTN 8	Insert line
FCTN 2	Insert characters	CTRL 1	Save text
FCTN 3	Delete line	CTRL 2	Load text
FCTN 4	Roll down	CTRL 3	Print text
FCTN 5	Next window	CTRL 4	Erase text
FCTN 6	Roll up	FCTN 9	Exit <i>HCJWord</i>

The Edit Screen

Because the TI-99/4A's video screen cannot display a whole page of text at one time, *HCJWord* uses an edit screen that works as a window. This window shows 23 rows by 28 columns of text. By using the Roll and Next function keys (see Control Capsule), you can move this window to display any section of the page that you desire.

The topmost line of the edit screen is the status line. Most of the time, this line displays the current row and column position of the cursor within the page. At other times, the status line is used for

Figure 1.

SAVE	LOAD	PRINT	ERASE							● ● HCJWord
DEL CHAR	INS CHAR	DEL LINE	ROLL ↓	NEXT →	ROLL ↑	TAB	INS LINE	EXIT		



Customizing The Program

If you desire, you may modify certain aspects of *HCJWord* to suit your personal requirements. The printer parameters, for instance, always default to the following when the program is first run: `RS232.BA=9600.DA=8`. If your printer's parameters are different, simply change the string variable `PRINTER$` in line 360 to equal the correct parameters. To make the program default to `PIO`, for example, change line 360 to read:

```
360 PRINTER$="PIO"
```

Also, once `PRINTER$` has been set correctly, there's no reason to enter the printer parameters everytime you go to print your text. To stop *HCJWord* from demanding printer parameters before printing a file, delete program lines 630 and 640. With these two lines deleted, pressing `[CTRL] 3` will print your text with no questions asked.

As for saving and loading files, you can also change the default device to something other than `DSK1`. To make the program default to `DSK2`, for example, change line 370 to read:

```
370 FILE$="DSK2."
```

How It Works

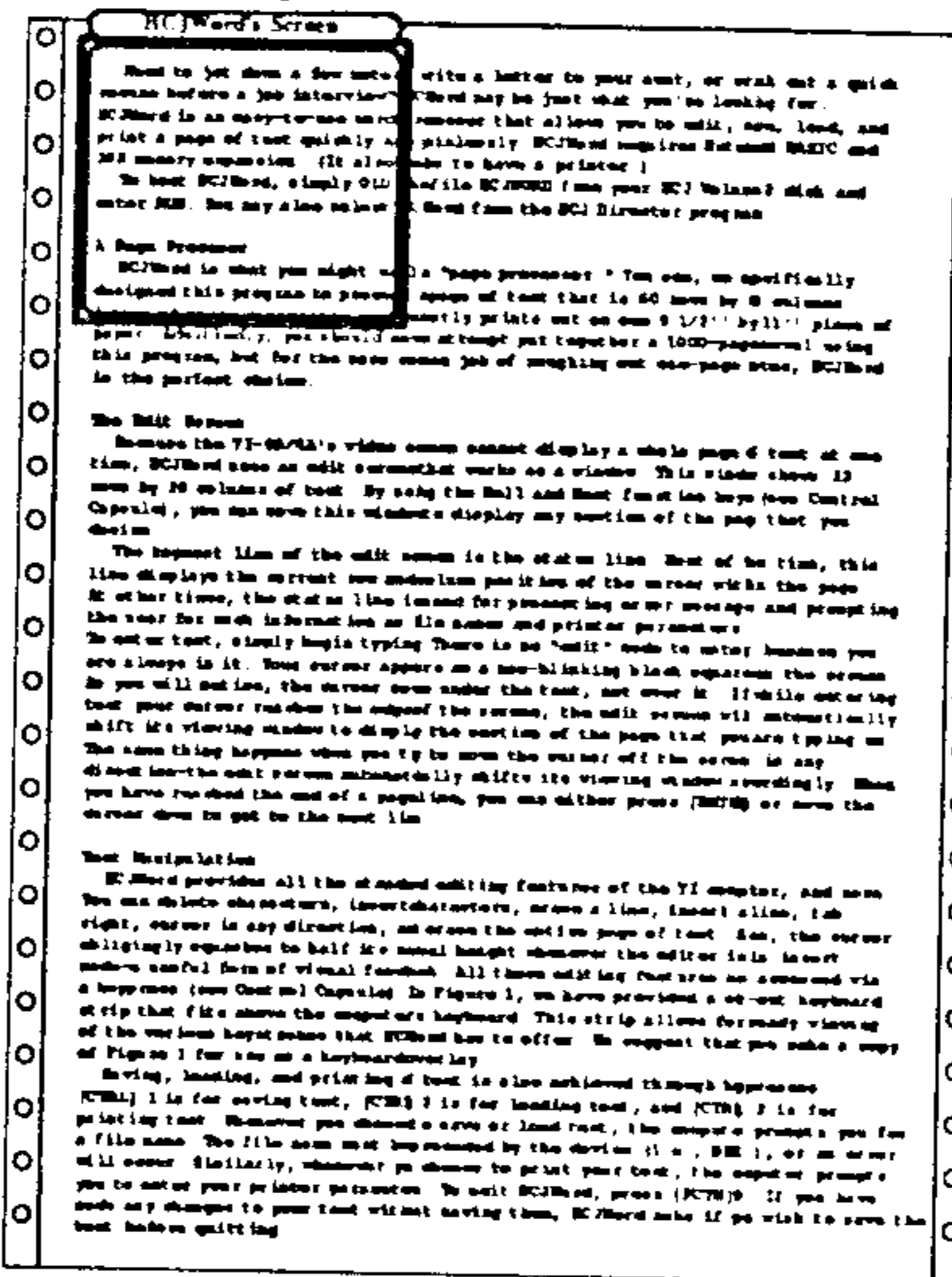
The engine of this program is contained in the file `EDITOR.OBJ`, found on your *HCJ* disk. This file contains the object code for a machine language text editor. This editor loads into the lower section of the 32K memory expansion unit (\$2000-\$3FFF). This area is also where the page of text is eventually stored. Because this area of memory expansion is unused by Extended BASIC, utilizing the editor does not limit the size of a program's BASIC code or variable space.

If you've already used the program *FormFlex*, a Feature program in this issue, you will no doubt see similarities in the *HCJWord* editor and the *FormFlex* editor. They both use the same editor kernel—`EDITOR.OBJ`.

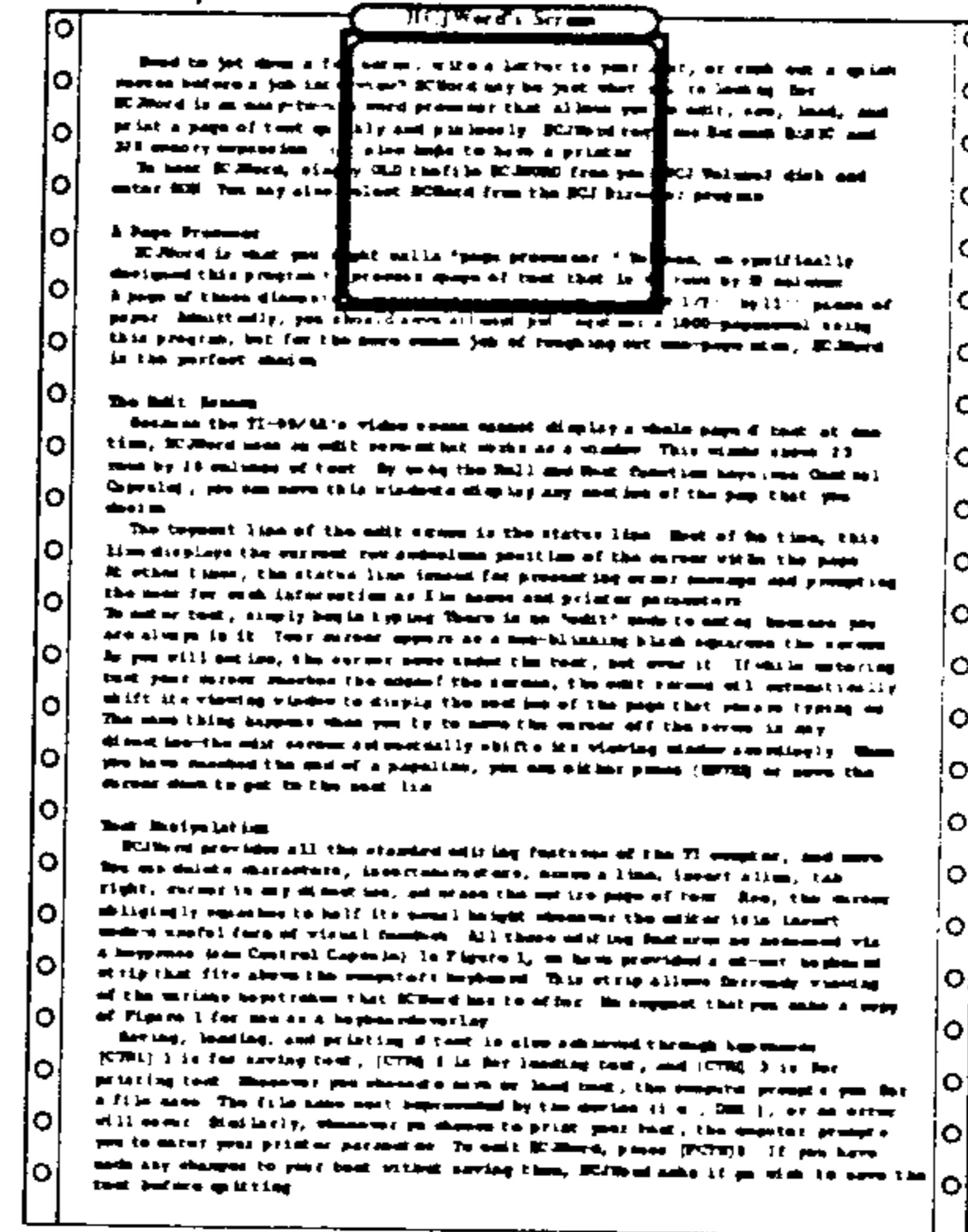
Because *HCJWord* requires the file `EDITOR.OBJ`, you must include this file on any backup of *HCJWord* that you make, or your backup will not boot properly.

HCJWord's Screen Windowing

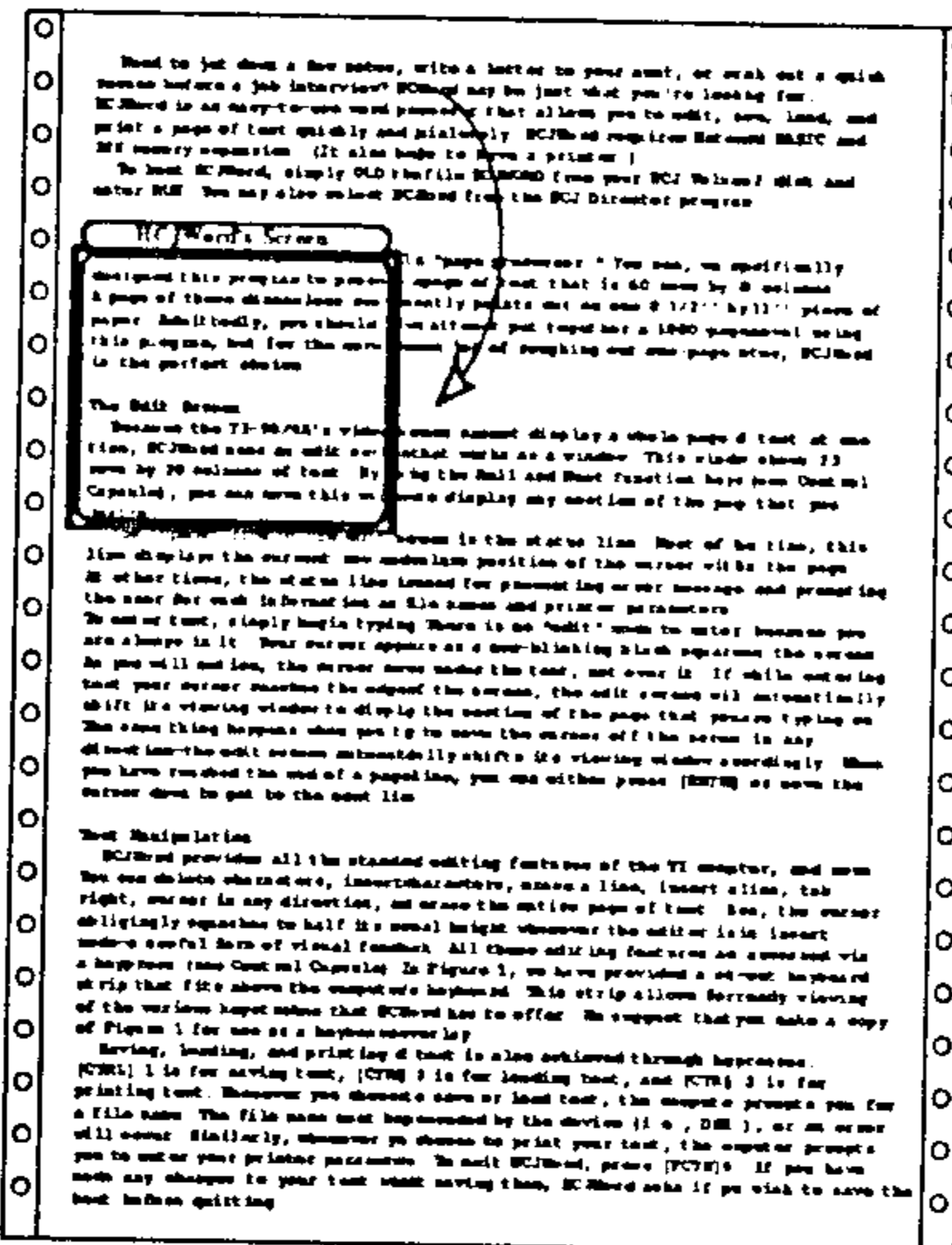
BEFORE



AFTER NEXT →



AFTER ROLL ↓



The Power of Paging

HCJWord uses the TI's screen as a window into a page of text. To enable you to view any portion of the page, *HCJWord* offers the Next and Roll functions. These functions move the screen quickly over the page of text in any specified direction.

The pictorial above shows how the Next window command (`[FCTN] 5`) moves the screen over to the right. Roll can move the window up or down on the page. To the left, we show how the Roll function (`[FCTN] 4`) is used to move downward.

This text window can move along the page by just one character at a time, through use of the cursor keys, but the Next and Roll features add speed and convenience to what might otherwise be a slow and tedious process.



IBM PC, PCjr, and Tandy 1000

Procedures For Using The IBM PC, PCjr, Or Tandy 1000

To make use of the HCJ Director menu program on your HCJ disk you need to backup your disk. Use the following procedures to produce an autoboot backup of your HCJ disk:

If you have a dual-drive system you may start with step 1, otherwise read this paragraph first:

For those of you with a single disk drive, you may still use the commands as listed below, though you will need to pay very close attention to the prompts on the screen instructing you to swap disks from time to time. The computer will tell you to place the appropriate disk in drive B:. What it means, however, is to remove the disk from drive A: and insert the disk which would have gone in drive B:. Using a single drive may mean having to swap disks quite a few times. For those who are patient though, the rewards are worth the added work. If you have further questions consult your DOS manual on the FORMAT and COPY procedures for a single-drive system.

1. Place your DOS master disk (hereafter referred to as the DOS disk) in drive A: and turn on the power to your system.

2. Enter the command **FORMAT B: /S /V**

3. The computer will ask you to place a blank disk into drive B: to be formatted. Ensure that the blank disk is in the drive and then press [Enter]. After formatting, you will be asked for a volume name. Enter **HCJOURNALn** where **n** is the Volume number. Then, you will be asked if you want to format another. Respond No to this prompt which returns you back to DOS.

4. If you wish to use a color monitor enter the command **COPY A:MODE.COM B:**

5. Enter the command **COPY A:BASIC*. B:BASIC.***

If you have an IBM compatible whose BASIC does not start with the word BASIC, then make adjustments in the command above for your version. In any case, the file on your new boot disk should always be named BASIC even if it was originally named BASICA.

6. After BASIC is copied to the new disk in drive B:, remove the DOS disk from drive A: and place the HCJ disk in drive A:

8. Enter the command **COPY A:*. B:**

9. After the last file has been copied, remove both disks from the system. Label the new disk as **HCJ ON DISK BACKUP Volume n**, where **n** is the Volume number, and place the original disk in a safe place.

10. The new disk you have created can now be used to boot your system (start from a power off condition) and will automatically bring up a menu of programs from which you may select.

You may also use the HCJ disk without backing it up if you:

1. Start from DOS 2.1 or later.

2. If you wish to run a program with a BAT, COM, or EXE extension, simply type the file name from the DOS A> prompt.

3. If you wish to run a BASIC program, you must first enter the appropriate version of BASIC, then LOAD and RUN the program.

Note: If you have an IBM PC and the program requires a color monitor, you must enable the monitor using the appropriate DOS MODE command before running the program.

Program Name	File Name	Language
HCJ Director	HCJDIR.COM*	Turbo Pascal
	AUTOEXEC.BAT	-batch file-
CodeWorks	CODEWORK.COM*	Turbo Pascal
FormFlex	FORMFLEX.COM*	Turbo Pascal
	MAKEFORM.CHN*	Turbo Pascal
	USEFORM.CHN*	Turbo Pascal
	COSTCOMP.FRM	-data file-
	COSTCOMP.FLD	-data file-
Re-Weaver	REWEAVER.BAS**	BASICA
Upsets	UPSETS**	BASICA
PC Character Editor	CHAREDIR.EXE****	Compiled BASIC
	CHAREDIR.BAS***	BASICA
	CHARSUBS.BAS***	BASICA
	IBMFONT.FNT	-data file-
	TANDFONT.FNT	-data file-
	NUKEMAN.FNT	-data file-
Nukeman	NUKEMAN.BAS***	BASICA
GET SEG	GETSEG.BAS**	BASICA

*Program requires: DOS 2.1 or later.

**Program requires: DOS 2.1 & either Cartridge BASIC on PCjr or BASICA on PC, or GW BASIC on Tandy 1000.

***Program requires: DOS 2.1 & either Cartridge BASIC on PCjr or BASICA, Color/Graphics Monitor Adapter, and Color Monitor on PC, or GW BASIC on Tandy 1000.

****Program requires: DOS 2.1 & Color/Graphics Monitor Adapter, and Color Monitor on PC.

TI-99/4A

Procedures For Loading The TI-99/4A With Extended BASIC

1. Ensure the Peripheral Box is properly connected to the console. Turn on the Peripheral Box.
2. Place the Extended Basic module securely in the machine.
3. Turn on the TI-99/4A computer.
4. Insert the HCJ disk into drive 1.
5. Strike any key to bring up the first menu, then select Extended BASIC, and The HCJ Director program will automatically RUN.
6. Select the number of the program you wish to use, then press [ENTER] and the program will load and RUN automatically.

Procedures For Loading The TI-99/4A With TI BASIC

1. Ensure the Peripheral Box is properly connected to the console. Turn on the Peripheral Box.
2. Turn on your computer and insert the HCJ disk in drive 1.
3. Strike any key to bring up the first menu, then select BASIC.
4. To load the BASIC program you wish to use, type **OLD DSK1.file name** where **file name** is the file name of the program. For example, if you wish to use FormFlex type **OLD DSK1.FORMFLEX** and press [ENTER]. Now, type **RUN** and press [ENTER].

Program Name	File Name	Language
HCJ Director	LOAD	Extended BASIC
CodeWorks	CODEWORK	BASIC
FormFlex	FORMFLEX*	Extended BASIC
	MAKEFORM*	Extended BASIC
	USEFORM*	Extended BASIC
	EDITOR_OBJ	TMS 9900 assembler
	COSTCOMP_F	-data file-
Re-Weaver	REWEAVER	Extended BASIC
Upsets	UPSETS	BASIC
	UPSETSX	Extended BASIC
HCJWord	HCJWORD*	Extended BASIC
	EDITOR_OBJ	TMS 9900 object file
File Verify	VERIFY1	Extended BASIC
	VERIFY2	Extended BASIC

*Requires 32K Memory expansion.

