SpritePOWER

- A Sprite Design Utility -

CREATED FOR THE COLECO ADAM™

TABLE OF CONTENTS

Chapter one: OFFICIAL NOTICES 1
- PRODUCT COPYRIGHT
- PUBLIC DOMAIN FILES
- DISCLAIMER
- STORAGE MEDIUM WARRANTY
- ALTERNATE COPYRIGHTS AND TRADEMARKS

Chapter two: GETTING STARTED 2
- WHAT YOU WILL NEED
- LOADING SpritePOWER
- THE SpritePOWER FILES
- USEFUL DEFINITIONS
- USING THE KEYBOARD

Chapter three: USING SpritePOWER 6
- THE PRIMARY MENU
- SCREEN COLOR
- RESET SPRITES
- VIEW SPRITES
- DRAW SPRITES
- ANIMATE SPRITES
- EXIT
- DIRECTORY OPTIONS
- CONTINUED DIRECTORY OPTIONS
- MORE DIRECTORY OPTIONS
- CAPTURING SPRITES

Chapter four: USING THE SPRITE SETS 17
- SPRITE FUNDAMENTALS
- SmartBASIC 2.0 SPRITE CONTROL
- THE HINKLE SPRITE COMMANDS
- THE SpritePOWER SPRITE ROUTINES

Chapter five: ASSORTED PROGRAMS 25
- Pix.MGR
- PUFF

SpritePOWER

CHAPTER ONE

OFFICIAL NOTICES

PRODUCT COPYRIGHT

PUBLIC DOMAIN FILES

When viewing the directory of the SpritePOWER medium (data pack or disk), you'll note that the first file (36K in length) is entitled "sp.obj". This is the object code of SpritePOWER program. This is the only program on the medium that is copyrighted by DIGITAL EXPRESS. All other files listed in said directory are hereby, if not already previously stated as such, donated into the public domain. This includes the three sets of sprites (stored both as Z80 and BASIC binary image files), the three picture files stored in SmartPAINT format, the Pix.MGR program, puff (the arcade - style game), and SpriteDemo.

DISCLAIMER

DIGITAL EXPRESS has exercised due care in the preparation of this software instruction manual and the programs which it describes. No warranty, either express or implied, is given as to the accuracy or suitability of the software or this manual for a particular purpose and DIGITAL EXPRESS shall not assume any liabilities for consequential or inconsequential damages arising as a result of using the software or interpreting the documentation.

STORAGE MEDIUM WARRANTY

DIGITAL EXPRESS warrants to the purchaser of this product that the storage medium is free from defective materials and workmanship. If the original storage medium (digital data pack or disk) fails to function properly, it will be repaired or replaced if returned to the following address:

DIGITAL EXPRESS
P.O. Box 37
Oak Hill, WV 25901

ALTERNATE COPYRIGHTS AND TRADEMARKS

The SmartBASIC interpreter is copyrighted by Lazer Microsystems, 1983. ColecoVision is a registered trademark and ADAM, SmartBASIC, and SmartWriter are trademarks of Coleco Industries.

****

SpritePOWER

CHAPTER TWO

GETTING STARTED

WHAT YOU WILL NEED

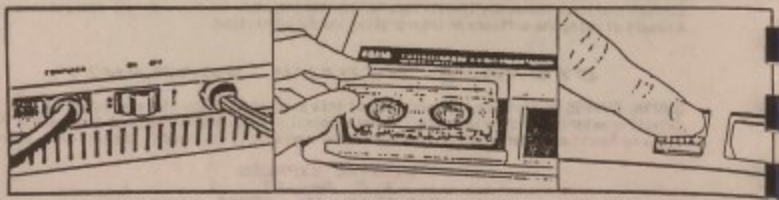
§ You will need an ADAM™ computer with a "revision 80" memory console. To verify your revision of the memory console, simply press the "R" letter key while holding down the CONTROL key in the "ELECTRONIC TYPEWRITER MODE".

§ You will need revision 79 SmartBASIC V1.0. To verify your version of the interpreter, enter PRINT PEEK (260) after loading SmartBASIC.

§ You will need a TV or monitor, preferably color.

LOADING SpritePOWER

The SpritePOWER electronic storage medium includes a number of files. The primary program has the same name as the package title, SpritePOWER. Before you can use the program, you must transfer it from the data pack or disk into ADAM's memory. This is called loading or booting the program. Once the program is loaded, it will stay in memory until you choose to exit to SmartWriter, press Computer Reset, or power down the system. Refer to the following diagram for loading SpritePOWER from a digital data pack; refer to your disk drive operator's manual for loading from a disk.



POWER SWITCH | INSERT DIGITAL DATA PACK | COMPUTER RESET BUTTON

DO NOT REMOVE DATA PACK WHILE THE DRIVE IS OPERATING!
DO NOT TURN POWER ON OR OFF WHEN A DATA PACK IS IN THE DRIVE!

HERE'S HOW TO LOAD SpritePOWER INTO ADAM's MEMORY

1. Turn ADAM on (by pressing the power switch located on the back of the ADAM printer).
2. Turn on your TV or monitor.
3. If you have disk drives, turn the power on to them.
4. Insert the SpritePOWER medium into one of the drives.
5. Close the drive door. Now, press Computer Reset.
6. Instantly ADAM will start playing a simple melody. Within a few seconds, a graphic title screen will appear as the remainder of the program loads into ADAM's memory. A few seconds later, you'll see six SmartKEYs with descriptive labels, at the bottom of the screen.

THE SpritePOWER FILES

The SpritePOWER electronic storage medium contains a number of files. The following list describes these.

THE PRIMARY PROGRAM

sp.obj: This is a powerful and user friendly machine code utility for designing your own sprite sets for use with ADAM in SmartBASIC or machine language programs. The program is enhanced with sophisticated graphics design and sound effects.

THE SPRITE SETS

The **sp.obj** program gives you three options for storing a set of sprites. You can store the file as a z80 binary image file, a BASIC binary image file (with an "H" file type), or a standard ASCII data file that can be transferred via modem, read by SmartWriter, or used with SmartBASIC (this file will have an "A" type). Since you may want to store a particular set of sprites in each of the three storage modes, it is recommended that you use a short filename suffix. You might want to use ".z" for the z80 file, ".b" for the BASIC binary image file, and ".a" for the ASCII version of the file. The SpritePOWER medium contains three sets of sprites. Each set is stored both as a z80 and a BASIC binary image file. These are:

geometry.z	geometry.b
misc01.z	misc01.b
alphabet.z	alphabet.b

THE DEMONSTRATION PROGRAMS

This medium includes two programs that demonstrate sprite usage; both of these programs require that you load your own SmartBASIC before you can use them. The program entitled SpriteDemo is very simple and replete with instructions in the REM statements. You could easily modify this program for use with your own sprite programs. The file entitled puff demonstrates sprite usage through an action packed arcade style game. You might want to play this game before using the design program just to get an idea of the exciting new dimension that sprites can add to your programs.

THE CAPTURE SPRITE FILES

SpritePOWER gives you two options for making a sprite file: you can draw them on a large grid or you can "capture" them from a high resolution graphics picture file. The picture file must first be stored in SmartPAINT format (another DIGITAL EXPRESS software package). The file Pix.MGR is included so that you may convert a picture to SmartPAINT format. You can BRUN this program after drawing on your MGR screen without disturbing the graphics or you can even load an RLE file. Three picture files are included on the medium that are already in SmartPAINT format. Each picture is stored as four 3K files with suffixes ".MRP", ".MR2", ".MR3", and ".MR4". Using these files is explained in detail later in this manual.

USEFUL DEFINITIONS

BIT IMAGE: Everything that you see on the your TV or monitor screen displayed by ADAM is generated by dint of bit image graphics. The screen is composed of 256 by 192 dots, or bits. When a dot is on (displayed), its bit value is one. When a dot is off (not displayed), its bit value is zero. Since ADAM's brain, a z80A microchip, is an 8-bit processor, every eight bits (starting at the left edge of the screen) are grouped into a specific quantity, called a byte. A collection of these bytes, whether one or 49152, is labeled a bit image.

KEYCLICK: Throughout the SpritePOWER program various sound effects and tones are used. Most of these are heard as the result of pressing certain keys. Keyclick is the technique of using arbitrary sounds to signify acknowledgeeent of keypresses.

MENU: As is the case with a restaurant's menu of its specific cuisine, a software menu is a collection of available options. SpritePOWER's primary menu consists of six options. Each option, in turn, presents its own menu of sub - options. Pressing the ESCAPE key at virtually any point within SpritePOWER will take you back to the primary menu of six options.

PIXEL: Pixel is an acronym for PIcture ELement. The term refers to a single dot on the monitor screen. This corresponds to a single bit of a bit - image.

SmartKEY: SmartKEY is the term that Coleco assigned to the six function keys on the top row of the keyboard designated by Roman numerals. You direct the SpritePOWER program by pressing certain SmartKEYs.

SPRITE: Sprites are user - defined patterns which provide for smooth animation without disturbing the background graphics. They may be positioned in multi - level overlays to create three - dimensional objects. ADAM's video chip supports 32 sprites to a set. Lower numbered sprites have a higher screen priority than higher numbered sprites. Consequently, when overlayed, higher numbered sprites will appear to be underneath lower numbered sprites.

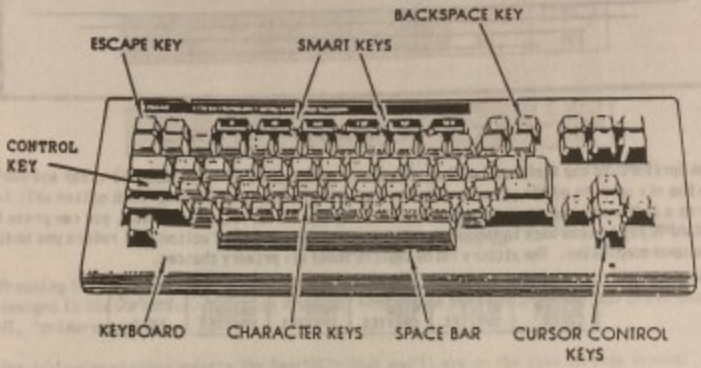
TOGGLE: Some of the SpritePOWER controls are selected by pressing a specified SmartKEY to turn the particular feature "on" or "off". This process, analogous to flipping a light switch, is referred to as "toggling" the control.

WINDOWS: Each screen within SpritePOWER can be considered as composed of three sections or windows. The largest of these sections is the graphics window; it is the upper five-sixths of the screen. Just below this is the message window; it is a light red text line used to display instructions pertinent to the SmartKEY controls. At the bottom of the screen is the SmartKEY window. This window is used to display Roman numerals corresponding to the keyboard's function keys; beneath each numeral a message is displayed describing the particular key's control.

WRAPAROUND: A grid of small squares is used by SpritePOWER to facilitate sprite design. The grid consists of 16 rows of 16 squares each. A single square represents one bit of a sprite's bit image. You can turn the bits on or off by using the arrow keys on the keyboard. A larger, solid box is used as a cursor to indicate your current position within the grid. If you move past the left edge of the grid, the cursor will go to the right edge of the grid; if you move past the bottom edge of the grid, the cursor will go to the top edge of the grid, and so on. This technique is known as wraparound.

USING THE KEYBOARD

You control the SpritePOWER program by pressing certain keys on the keyboard. The following illustration depicts the standard ADAM keyboard; you may want to refer to it in reading the key descriptions below.



BACKSPACE KEY: You may use this key or the left arrow cursor key to erase letters when entering a file's name for storing or renaming.

CHARACTER KEYS: You will use the character keys to enter file names for storing or renaming.

CONTROL KEY: While drawing sprites, the CONTROL KEY may be used to increase or decrease the current sprite number. To use the control key, you must first hold down the control key and then press the other indicated key -- in this case, in conjunction with the up or down arrow cursor keys.

CURSOR CONTROL KEYS: The cursor (or arrow) keys are used to move the cursor within the sprite design grid. They are also used to select sprite sets from the directory's file page.

ESCAPE KEY: Throughout the SpritePOWER program you can branch program execution back to the primary menu of six options by pressing ESCAPE.

SmartKEYS: These six keys are on the top row of the keyboard and are indicated by Roman numerals. They are used to control the primary functions of the SpritePOWER program.

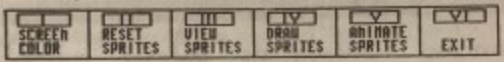
SpritePOWER

CHAPTER THREE

USING SpritePOWER

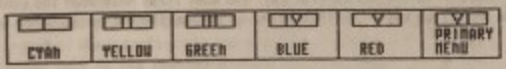
THE PRIMARY MENU

When SpritePOWER has finished loading into ADAM's memory, the title screen will clear and you'll see the six options of the primary menu displayed in the SmartKEY window. Each option will take you on a different course of action within the program. At virtually any point, you can press the ESCAPE key to come back to these six options; and, each course of action will return you to this menu upon completion. The picture below depicts these six primary choices.



SCREEN COLOR

Pressing the Roman numeral SmartKEY "I" from the primary menu, will take you to the six screen color options. The picture below shows these six choices. To select the color of your choice, just tap the SmartKEY which corresponds to your choice. When you are done, you can press SmartKEY VI, "primary menu", to go back to that menu. Pressing the ESCAPE key will also take you back to the primary menu.



RESET SPRITES

Pressing the Roman numeral SmartKEY "II" from the primary menu gives you the option to reset the sprites in the workspace to the default (start - up) shape patterns, *ie*, each one is changed back into a solid block. Use this option with care; you could accidentally undo hours of your own sprite designs.

As a precaution, you are asked if you really want to reset the current sprite designs. The first picture below depicts the SmartKEYs that are presented. Press SmartKEY VI, "YES", if you're certain that you want to reset the sprite designs. If you would rather not undo any changes that you've already implemented, just press SmartKEY V, "NO". If you select the "YES" option, ADAM will reset the sprite designs to the default block shapes. Then the message window will apprise you that the sprites have been reset, and you'll see the SmartKEY depicted in the second picture below. Just press "VI", primary menu, to go back to the first menu of six options.

VIEW SPRITES

Pressing the Roman numeral SmartKEY "III" from the primary menu will allow you to view the sprite designs in the workspace. When you are ready to continue with the program, just press SmartKEY VI, "primary menu".

The following picture depicts the SmartKEYs that you'll see on the view sprites screen. Press SmartKEY IV, page one, to see the first sixteen sprites in the set. Press SmartKEY V, page two, to view the second sixteen sprites in the set. Just above and to the left of each sprite, its number in the set is displayed. You can view the set at double magnification by first accessing that option from the DRAW SPRITES menu (which is discussed below).

DRAW SPRITES

Pressing the Roman numeral SmartKEY "IV" from the primary menu will permit you to design your own sprites, either from scratch or as minor modifications of the sprite set currently in the workspace. In addition to the normal SmartKEY commands, you also have nine CONTROL functions to choose from in designing your sprites. These are listed below, as well as, on the screen.

↑ = number up
↓ = number down

G = grid to sprite
S = sprite to grid

C = change color
M = change size

R = reverse grid
F = flip grid
I = invert grid

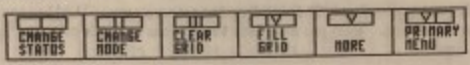
To use these nine commands, you must first hold down the CONTROL key and then depress the other indicated key. Each function changes an aspect of the design grid or of the sprite itself. Let's take a look at the DRAW screen.

The graphics window can be considered as four integrated components. At the upper left, you'll see the list of the nine control functions. Below this three DRAW parameters are indicated. These are: status (cursor up or down), mode (cursor draws or erases), and number (revealing the number of the sprite currently being displayed).

On the upper right side of the graphics window you'll see the sprite design grid. It consists of 16 rows of 16 blocks each. Each block represents one pixel in a sprite's design. The grid is eight times as large as the sprite (at normal magnification). You'll note three types of blocks within this grid. When you first start using the program, each block is hollow. Each small hollow block indicates that the pixel is off. A small solid block means that the pixel is turned on. The cursor (position indicator) is a large, flashing solid block.

Centered below the sprite design grid, the current sprite is displayed. Note that when you first start using the program, all 32 sprites are just large blocks.

At the very bottom of the screen the six SmartKEY functions are revealed. The picture below depicts these functions.



Now let's examine all these design controls and functions that you have access to. First, let's try the SmartKEY functions. SmartKEY I, change status, will toggle the cursor status. Try it. Tap the Roman numeral key and watch the status description (at the lower left side of the graphics window) change. If it was "up", it will change to "down". If it was "down", it will change to "up". The status controls the cursor effect on moving within the design grid. When the status is "up", cursor movement will not have an effect on the grid. When the status is "down", any cursor movement will change the design on the grid. THIS IS HOW YOU CREATE A DESIGN.

SmartKEY II, change mode, toggles that parameter between "draw" and "erase". If the mode was "draw", it will change to "erase". If the mode was "erase", it will change to "draw". Note that the status MUST BE "DOWN" in order for cursor movement to change the grid. Moving the cursor (while the status is "down") when the mode is "draw" will make a grid block solid. Moving the cursor (while the status is "down") when the mode is "erase" will make a grid block hollow. Note that solid blocks represent the viewable portion of a sprite and hollow blocks represent the part that is not seen.

The function of the third and fourth SmartKEYs is very simple and straightforward. SmartKEY III, clear grid, will instantly make all the blocks in the design grid hollow. SmartKEY IV, fill grid, will instantly make all the blocks in the design grid solid. Note that both of these options will clear the design that was previously displayed on the grid. Thus, you should use them with CARE.

The sixth SmartKEY, primary menu, will just return you to the program's entry menu of six primary functions. Also, pressing <ESCAPE> will branch the program back to the first menu.

The fifth SmartKEY from the DRAW menu, labeled "more", allows you to store or retrieve a sprite design from an internal program storage buffer. This permits you to exchange or merge sprites from one set to another. When you tap SmartKEY V, the graphics screen will clear except for the the sprite design grid. You are then presented with the following three options.

Pressing SmartKEY IV here, store grid, will put the grid values into this internal buffer. The grid itself will NOT change. Later you could retrieve a file from a disk or data pack and then use the SmartKEY V option, load buffer, to transfer the sprite from the earlier set to the latter. When you first start using SpritePOWER, the contents of the buffer sprite is a large block. Note that RESETing the sprite designs will NOT have an effect on the sprite design in the buffer.

These two buffer options only involve the sprite INDIRECTLY, access being accomplished via the grid -- this is a failsafe feature to prevent accidental destruction of an actual sprite in the set. When you LOAD the BUFFER, the sprite which was previously on the grid will be replaced by the contents of the buffer.

Upon tapping any of the three BUFFER function SmartKEYs you are taken back to the DRAW menu. The sixth SmartKEY here, previous menu, allows you to go back without any buffer exchange (in the event that you changed your mind after depressing the SmartKEY for "MORE" draw options).

Let's take a look at those nine CONTROL commands and cursor movement. You can move the cursor within the grid (as you may have already guessed) with any of the four arrow keys located on the lower right side of the keyboard. Note that a grid wraparound feature is employed. If you move the cursor beyond one edge of the grid it will "wraparound" to the opposite edge. This feature is employed to provide for rapid relocation of the cursor.

Holding down the <CONTROL> key and then depressing either the UP or DOWN arrow key will change the current sprite number. This will cause two changes in the graphics window. The current number (displayed at the lower left side of the graphics window) will change to reflect the new number. And, the current sprite design (NOT THE GRID DESIGN), which is centered beneath the grid, will reflect the design of that particular sprite number.

The next two CONTROL options (^G and ^S) are antithetical. Holding down the <CONTROL> key and then depressing the "G" letter key will transfer the design on the grid to the current sprite number (displayed beneath the grid). And, holding down the <CONTROL> key and then depressing the "S" letter key will transfer the current sprite's design to the grid.

Using the CONTROL + G command is how you enter a sprite into the current set. First design the shape on the gird, then use ^G put the design into the sprite set.

The CONTROL + S command allows you to edit or modify a design already in particular set. First use CONTROL plus an up or down arrow key to get the sprite that you want to edit. Then use the ^S command to transfer the sprite to the grid. Now edit the shape (using the previously discussed controls and functions). When you're finished, just tap ^G to transfer the modified shape on the grid to the actual sprite.

At this point you should note that the grid's design will remain the same until you change it. Thus, you can move anywhere throughout the program without disturbing the grid's design values. Try it. When you come back to the DRAW menu, you'll see that the design is still there.

The CONTROL + C command will change the color of the current sprite. Repeatedly using ^C will cycle through the available color choices. The color will instantly change for the sprite displayed below the design grid. This new color will remain in effect until you RESET the sprites; GETting another file from a data pack or disk will not change the sprite color values. You'll note that all sprites in this program are displayed on a black background. This feature is employed to prevent any possible confusion regarding the color of a sprite. Accordingly, the program does not allow you to change the color of a sprite to black. This is an intentional limitation of the program; in your own programs you may, of course, use black colored sprites.

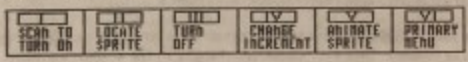
The CONTROL + M command instantly changes the magnification or size of the sprite set. The entire set may be either standard size (16 by 16 pixels) or double magnification (enlarged to 32 by 32 pixels). Once you toggle the size of the set, the new size will remain in effect until you change it. You should note that ^M is the same as pressing the <RETURN> key.

The last three CONTROL commands permit you to quickly manipulate the bits designated on the sprite design grid. CONTROL + R will reverse the grid; the previously solid blocks will change to hollow and the previously hollow blocks will change to solid. Thus this option compliments the bit image transposing set and reset bits.

The CONTROL + F command will flip the grid horizontally. This produces a mirror image of the previous design. An even number of consecutive uses of this option restores the grid to the original shape design. The CONTROL + I command inverts the grid vertically. This generates an upside-down image of the previous shape. Again, an even number of consecutive uses of the command restores the original shape design. These two commands may appear to have no effect on a design that is symmetrical.

ANIMATE SPRITES

Pressing the Roman numeral SmartKEY V from the primary menu will present the sprite animation options. These are depicted below.



At the top, center of the graphics window you'll see a small square; this is the sprite activation window. It allows you to view sprites that are not currently displayed somewhere else on the animation screen. Now let's explore the six SmartKEY animation functions.

Roman numeral SmartKEY I, scan to turn on, permits you to cycle through the sprites which are not currently displayed elsewhere on the screen. The message window instructions direct you. Use the up or down arrow key to change the sprite number (displayed inside the sprite activation window along with the sprite itself). If you want to abort the "turn on" procedure, just tap <ESCAPE>. When you find the sprite that you want to "take" for animation, just press <RETURN>. This scanning will pass over any sprites that are already displayed elsewhere on the screen. An error buzz will sound if you depress SmartKEY I and all 32 sprites are already displayed for animation.

SmartKEY II, locate sprite, is useful in finding a particular sprite when you have several of them displayed on the screen. If you select SmartKEY II when no sprites are displayed, an error buzz will sound.

You will be asked to enter the sprite number that you want to locate. The sprites are numbered 1 thru 32. You can also enter the number zero to locate the last sprite turned on (regardless of its number).

To locate the sprite an intersecting horizontal and vertical line is drawn. The sprite is just inside the southeast intersection quadrant. Since some pixels may not be turned on in the sprite's shape, it may be offset to the right from the intersection border.

SmartKEY III, turn off, allows you to remove a sprite from the animation screen. If no sprites are already turned on, an error buzz will sound when you depress the Roman numeral key.

You will be prompted to enter the number of the sprite that you want to turn off. Again, you may enter the number zero to indicate the last sprite that was turned on.

SmartKEY IV, change increment, permits you to select the pixel increment for animation. The default setting is "one". You may enter any value between one and fifty, inclusive. You may also press SmartKEY IV simply to ascertain the current value. With this option, just tap <ESCAPE> rather than entering a new increment value.

SmartKEY V, animate sprite, is the central control of this menu of options. If no sprites are already turned on, an error buzz will sound when you press the Roman numeral key.

First, you are asked to enter the sprite number to animate (move on the screen). You may also enter zero to opt for the last sprite that you turned on. Note that you MUST turn on the sprite BEFORE the program will permit you to animate it.

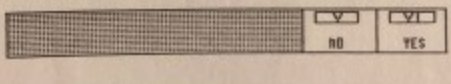
Use the four arrow keys (located at the lower right side of the keyboard) to move the particular sprite. When you have finished, press <RETURN> to go back to the menu of six animation options.

In actual use within your own programs, you may want to utilize several sprites to create one large and/or multi - layered shape which are to be animated in unison. The animation aspect of this program is not designed to demonstate that sophisticated feature. However, you could use it to place several sprites close together to view a stationary image of such an arrangement.

SmartKEY VI, primary menu, will return you to the program's first six options. Tapping <ESCAPE> will also accomplish this. You should note that SpritePOWER does not store the locations to which you moved the sprites. Thus, every time that you select the ANIMATE SPRITES option from the primary menu, you will start out with no sprites turned on for animation.

EXIT

Pressing the Roman numeral SmartKEY VI from the primary menu of six options allows you to leave the program. You are asked if you really want to reset your ADAM with the following two choices. If you select "NO", you'll be returned to the primary menu. Choosing "YES" will reset your computer just as if you had pulled the computer reset switch. If there is a medium in one of the drives, it will be booted. If not, your ADAM will jump to SmartWriter.

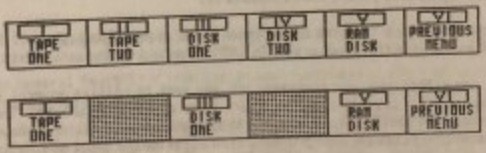


DIRECTORY OPTIONS

Pressing the STORE/GET word processing key from the primary menu will present you with a variety of file access choices. Unlike the other options from the primary menu, this one has more than one menu of sub - options. The first of these sub - menus presents three options. These are depicted below.

SmartKEY VI will take you back to the primary menu. SmartKEY V will read the medium (data pack or disk) in the specified drive; be CERTAIN that you have a storage medium in the specified drive BEFORE pressing SmartKEY V. At first this specified drive will be the one that SpritePOWER was booted from. You can change the current drive with the SmartKEY IV option, select new drive.

When you press SmartKEY IV, ADAM will scan each storage drive to ascertain whether or not it contains a medium. Thus, you should insert any alternate media in the drives BEFORE pressing SmartKEY IV, select new drive. If you have a storage medium in all four drives, you'll see the information in the first illustration below in the SmartKEY window. Only drives which contained media when you chose "select new drive" will be displayed. For instance, if only the first tape and disk drives contained a storage medium, you'd see the information depicted in the second picture below.

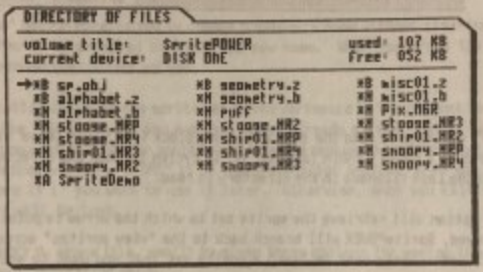


You'll note that SmartKEY V is labeled ramdisk. Upon completing the SpritePOWER program (sp.obj), 13K of user memory was still free (one of the benefits of programming in machine code). This extra RAM has been dedicated to serve you as a ramdisk. It acts as an ultra - fast storage drive allowing you to store and retrieve sprite files without accessing the standard auxiliary devices. When you EXIT the program these files will be lost. Thus, you should use this ramdisk as a temporary workspace. Then when you are ready to complete your session with the program, just GET the files individually from the ramdisk that you want to keep and STORE them permanently on a disk or data pack.

To select a new drive, just tap the SmartKEY which corresponds to the drive that you want to access. When you do, SpritePOWER will take you back to the first menu of three directory options. Then simply depress the Roman numeral "V" key, read medium.

CONTINUED DIRECTORY OPTIONS

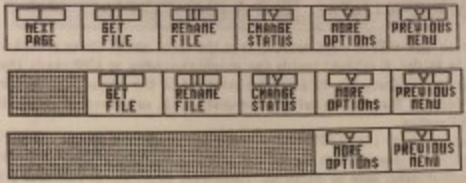
When you do press SmartKEY V, read medium, from the first set of three directory options, you'll see a blank file folder displayed inside the graphics window as ADAM reads the selected device. When this task is finished, the file names contained in the medium's directory will be written onto the file folder. The picture below shows what you'll see revealed within the file folder for the SpritePOWER medium.



In the top part of the folder, the current device is shown (disk one in the picture above). And, the volume's title is also shown. To the right of this data, you'll see how many blocks of the storage medium are USED (for file storage) and how many blocks are FREE (available for file storage). Below this, on the file folder, all the file names are shown (in three columns). Just to the left of each file name, you'll see the file's type displayed (as with SmartBASIC). This will most likely be an "A" (for an ASCII file: BASIC program, SmartWriter document, a BASIC text file, etc.), an "H" (for a binary image BASIC file, a SmartWriter document, a binary converted BASIC program, etc.), or a "B" (for a z80 program -- z80 programs acually have an "ASCII 2" filetype). Just to the left of the file type, an asterisk may be displayed if the file is LOCKed.

You'll see a red arrow pointing to the right. You can position this arrow using any of the four cursor arrow keys to select a file by pointing to it. A full wraparound feature is employed for moving this pointer arrow -- this permits fast pointer movement. Also, pressing the HOME key will return the pointer to the upper left starting position. If the medium doesn't contain any standard files, a message stating so will be displayed vice any file names.

Depending on certain conditions you'll see one of three possible sets of SmartKEY options beneath the file folder. If the medium that ADAM just read had a two (or more) block directory with file names in the extra directory blocks, you'll be presented with the set of options depicted in the first picture below. If you accessed a standard one block directory (with files stored on the medium), you'll see the set of options depicted in the second picture below. If the medium was blank (did not contain any standard files), you'll be given the set of two options depicted in the third picture below.



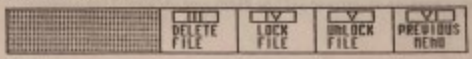
The "next page" option will read the next directory block (for two or three block directories). This option works in a round-robin fashion, *ie*, it cycles through the directory blocks starting over again when the last filename in the directory is read.

The "get file" option will retrieve the sprite set to which the arrow is pointing. When the file has been retrieved, SpritePOWER will branch back to the "view sprites" screen (to show you the set). You will most likely want to "get" a sprite set that you want to modify or to continue developing.

The "rename file" option allows you to change the name of any file stored on the current medium. You do NOT need to enter the file type when renaming a file; the program will give the new filename the same filetype as the previous filename. As is the case with all of the file access options, be certain that the arrow is pointing to the filename that you want to access BEFORE you select a SmartKEY function.

With this option, SmartKEY III, the file name will be highlighted on the file folder after you press the Roman numeral function key. Just type in the new filename and press the RETURN key to have ADAM change the name. You can press BACKSPACE or the left arrow key to erase typed letters in the new filename. If you press ESCAPE while entering the new name, the program will go back to the PREVIOUS menu rather than to the primary menu, which is the normal response to an ESCAPE keypress.

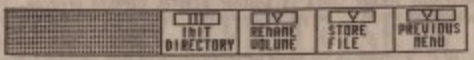
Pressing SmartKEY IV, "change status" presents another set of file access choices. These are revealed in the illustration below.



When you press a SmartKEY from this menu, the specified action will be performed on the filename which is highlighted on the file folder. DELETEing a file erases it from the directory; use this option with EXTREME CAUTION! LOCKing a file protects it from deletion. When a file is first stored, its status is "UNLOCKed". All the files that come on the SpritePOWER medium are LOCKed (protected from DELETE) and incapable of being UNLOCKed -- this a precautionary measure.

MORE DIRECTORY OPTIONS

The "more options" selection gives you three additional directory controls. The illustration below shows these.

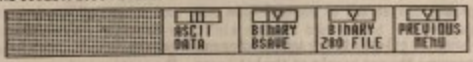


Use the "init directory" option with EXTREME CAUTION; this will clear all the files stored on a medium. Note that the SpritePOWER medium is protected from INITializing. When you select this option, you'll be prompted to enter the new volume name and the block size for the directory, ie, one (standard), two or three blocks.

The "rename volume" option permits you to change a volume's name without clearing the files from the directory. You'll be prompted to enter the new name. When you press the RETURN key, SpritePOWER will rename the current volume.

The "store file" option will save the sprite set in the workspace to the current medium. You'll be prompted to type the file name. If the name you enter already exists, you'll be asked to enter another name. If you want to transfer sprite sets from one medium to another, you can "GET" a file from the source medium and then "STORE" it on the destination medium. When you design a sprite set, you MUST store it if you want to use it later. Otherwise, when you EXIT the SpritePOWER program, your work will be lost.

When you tap SmartKEY V, store file, you'll be given three options for saving the sprites. These are depicted in the illustration below.



SmartKEY III, ascii data, permits you to store the sprite values of the set as a standard text file with line numbers. The information will be entered in DATA statements (16 elements per program line number). You can GET this file with SmartWriter or LOAD it from SmartBASIC -- the file can even be transferred via the ADAMlink modem software. You'll be asked to enter the starting line number (1 to 63000, inclusive) and the line number increment (1 to 10, inclusive). When you view the ASCII sprite file from one of the aforementioned programs, you'll note that leading 0's are included. This might be considered as an intentional bug in the program; but, it provided for a standard length for each of these ASCII files. Note that the SpritePOWER program (sp.obj) will NOT allow you to GET an ASCII data file.

SmartKEY IV, binary bsave, allows you to store the sprite values as a BASIC "M" file. The default BASIC BLOAD address is 29696; but, you can use your own address parameter. The SpritePOWER program DOES permit you to GET a sprite file stored as a BASIC binary image file (type "M").

SmartKEY V, binary z80 file, allows you to store the sprite values as a z80 binary image file. This file is intended for use with machine code programs. The SpritePOWER program DOES permit you to GET a sprite file stored as a z80 binary image file. When you store files on the ramdisk, you will most likely want to employ this option -- thus allowing room for eleven files on the ramdisk.

CAPTURING SPRITES

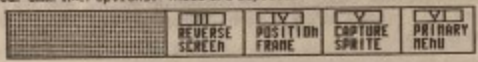
In addition to drawing sprites on the design grid, SpritePOWER also gives you the option to "capture" sprites from high resolution graphics pictures. First, though, the picture must be converted to SmartPAINT format (this is another DIGITAL EXPRESS program, featured on our ShowOFF I package).

The program entitled Pix.MGR is included on this medium so that you may convert your own HGR pictures or RLE picture files to SmartPAINT format. You must first boot your own SmartBASIC before using this program. If you are already in HGR graphics mode when you BRUN the program, it will NOT clear your screen.

There are also scores of public domain SmartPAINT picture files available. Three of these are included on the SpritePOWER medium.

SpritePOWER allows you to GET three types of files: BASIC binary image sprite files, z80 binary image sprite files, and SmartPAINT picture files. To GET the latter, just point to any one of the four component files of the picture (designated by the suffix ".HRP", ".HR2", ".HR3", or ".HR4"). When you tap the GET SmartKEY, ADAM will retrieve the picture (it does NOT matter which one of the component files you point to).

Only the bit image of the file will be retrieved; since a sprite can only be assigned one color there is no reason to get the picture colors. When the picture is loaded into memory, you'll be presented with four SmartKEY options. These are depicted in the illustration below.



The picture will have two colors: black and the current screen color. The portions of the picture that are black represent the set pixels of the bit image. The portions of the picture that are NOT black (the current screen color) represent the unset (or reset, not viewable) pixels of the bit image. You can reverse the bit image (set to reset, and vice versa) by tapping SmartKEY II, reverse screen.

At the upper left of the screen you'll note a small white square. This is the sprite capture frame. To move this frame, just press SmartKEY IV, position frame. The on-screen instructions direct you to move the frame with the arrow keys (at the lower right side of the keyboard). When you have finished relocating the frame, just press the <RETURN> key.

Now, to capture the bit image within the frame, just tap SmartKEY V, capture sprite. The graphics window will clear. Three items of interest are then revealed. At the left top of the screen, you'll see the captured bit image. At the center top of the screen, you'll see the current sprite design. And, at the right top of the screen, the current sprite number is shown. You can change the number with either the up or down arrow key. To transfer that captured image to the current sprite, just press <RETURN>. When you are finished with that particular image, press <ESCAPE> to go back to the menu of four SmartKEY capture options.

You may capture as many sprites as you want from the hi-res picture (note only 32 sprites to a set, though). To leave the capture mode, just depress SmartKEY VI, primary menu. When you do, the picture will be lost. You may then want to choose the DRAW option from the primary menu to edit a captured sprite.

SpritePOWER

CHAPTER FOUR

USING THE SPRITE SETS

SPRITE FUNDAMENTALS

SCREEN DEPTH

Sprites are graphics patterns which provide for smooth animation by not destroying background graphics or text. ADAM's video chip supports 32 sprites on a single screen. They can be used in 32 column TEXT mode, graphics mode 2 (GR, HGR, and HGR2), and multi-color mode (not accessed by SmartBASIC). Sprites are not available in 40 column TEXT mode.

The video chip gives each sprite a screen priority based on its sequence in the bit image table. The first sprite has a higher priority than the second one, and so on. The thirty second sprite has the lowest priority. The priorities are important because each sprite occupies a different level of depth on the video screen.

There are 34 planes of depth to the screen. The most distant is the default background which comprises the 256-by-256-pixel video screen. This default background is employed solely for screen color. Hackers will be recognize that the VDP register 7 controls the color.

Atop this plane of depth is the graphics screen which is 256 pixels across by 192 pixels down. The graphics screen can be used to display any type of bit image design: alphabetic characters, shapes, trig graphics, and so on. In 32 column TEXT mode, color is assigned to each set of eight characters. In graphics mode 2 (GR, HGR, and HGR2) one set and one reset color can be assigned to each byte of the bit image (every eight pixels starting from the upper left corner of the graphics screen). Set bits of this screen are commonly referred to as the foreground and reset bits are usually called the background.

The upper 32 planes of depth are reserved for the sprites. Thus, all sprites appear to be on top of the graphics screen (text and other bit image graphics). It is these planes of depth that permit you to move sprites without disturbing the graphics screen. Each sprite occupies its own plane designated by the sprite number. The 32nd sprite is in the plane just above the graphics screen. The 31st sprite is in the plane just above the one designated for the 32nd, and so on. The first sprite is in the plane which is most distant from the graphics screen (closer to you). The arrangement is like this:

```
plane 34:  default background
plane 33:  graphics screen
plane 32:  sprite # 32
plane 31:  sprite # 31
plane 30:  sprite # 30
.........
plane 3:   sprite # 3
plane 2:   sprite # 2
plane 1:   sprite # 1
YOU, the computer user
```

This dimensional information can be used by the programmer to create 3D shapes by positioning sprites according to their screen priority. Also, you should design your sprites with these facts in mind. Most likely, you will want to use the first sprite for your most important design.

SPRITE SIZE

There are two basic sizes of sprites: 8-by-8 pixel and 16-by-16 pixel (because of the greater design resolution, SpritePOWER only works with the larger size). Each size may also be enlarged to double magnification. But, this enlargement option changes the entire set; magnification can NOT be set for individual sprites.

POSITIONING SPRITES

Sprites can be positioned at any graphics screen pixel location. Thus, they can have a vertical position of "0" thru "191", inclusive. And, they can have a horizontal position of "0" thru "255", inclusive. When animating sprites, you'll note that a "fade off" feature is employed by the video chip for the bottom vertical coordinates and the far right horizontal coordinates.

There is one major limitation in animating multiple sprites. The video chip will only permit four sprites to be viewable at any given vertical coordinate (horizontal plane). When more than four sprites are positioned at the same vertical location, the lower priority (higher numbered) sprites will vanish partially or entirely. As a programmer, you'll need to ensure that this limitation does not have an adverse effect on your creations.

In addition to the standard 192 possible vertical positions, ADAM's sprites can be located at two special vertical coordinates. A vertical position of "200" turns a sprite off. A vertical position of "208" turns that sprite off plus all the higher numbered (lower priority) sprites, as well. For instance, suppose you have all 32 sprites displayed on the screen. If you set the vertical coordinate of the second sprite to "208", all the sprites numbered from the second to the thirty second will instantly disappear.

SPRITE AND GRAPHICS COLOR

Each sprite may be assigned ONE color for the SET pixels. The RESET pixels will default to the next lower priority plane's color giving the sprite the appearance of being on top of background graphics. Any of the video chip's 15 colors may be employed. The color code values are different than those used by either GR COLOR or HGR HCOLOR. Rather, the true internal color code values are employed. These are described below.

0 = transparent	8 = medium red
1 = black	9 = light red
2 = medium green	10 = dark yellow
3 = light green	11 = light yellow
4 = dark blue	12 = dark green
5 = medium blue	13 = magenta
6 = dark red	14 = gray
7 = cyan/aqua	15 = white

The transparent option assumes the color of the default background (discussed above). Medium red is more of a venetian red hue which may be used a brown. These 16 color code values are ADAM's master color codes. Here are some tips on using these values to add visual appeal to your own BASIC programs.

SmartBASIC V1.0 screen colors:

Let's start with the version of BASIC that came with your computer, SmartBASIC V1.0. Each of the four video modes that it supports (32 colum TEXT, GR, HGR, and HGR2) uses individual setup routines. You can POKE values into these interpreter routines to set screen color. The addresses are:

17059 = TEXT default background
17115 = TEXT NORMAL fonts
17126 = TEXT INVERSE fonts

18607 = GR default background
18633 = GR graphics window
18711 = GR TEXT fonts

25431 = HGR default background
25471 = HGR graphics window
25568 = HGR TEXT fonts

Changing the default background colors is very simple. Just select your color preference from the master color code table (listed at the bottom of the previous page, 18) and POKE it into the appropriate address. Then just enter the corresponding video mode command. Suppose you want to change the TEXT default background to dark blue. Just do this ...

```
POKE 17059, 4: TEXT    (RETURN)
```

Changing the other screen color values is a little more complex. Each of these addresses requires two color values per byte. The SET pixel color is stored in the high nibble (multiplied by 16) of the color byte and the RESET pixel color is stored in the low nibble of the color byte. Suppose you want the NORMAL TEXT fonts to be black on a cyan background and you want the INVERSE TEXT fonts to be white on a dark red background. Here's what you do ...

```
POKE 17115, (1*16) + 7
POKE 17126, (15*16) + 6
TEXT
```

When changing the BASIC screen colors, you should keep a few facts in mind. You must enter the corresponding video mode command (TEXT, GR, HGR, or HGR2) in order to implement the color change. The HGR2 POKEs are the same as those for HGR except that no TEXT will be displayed. FLASH, in the TEXT mode, is the result of alternating between NORMAL and INVERSE colors. In TEXT mode you should NOT use the same color for SET and UNSET pixels -- the fonts will NOT be viewable. And, you may want to PEEK the nine screen color addresses BEFORE POKEing in new values.

SmartBASIC V1.0 graphics color tables:

The value of the current GR COLOR is at address 16776. The value of the current HGR HCOLOR is at address 16777. You can also use the HPLOT command to erase points (IPLOT) by POKEing a 128 into address 16777 instead of using the HCOLOR command.

You can correct the COLOR, HCOLOR, and SCRN commands so that they use the master color code values. This way, you only have to learn one set of color values. Here's how ...

To correct HCOLOR:

```
FOR x = 0 TO 15: POKE 18765 + x, x: NEXT
```

To correct COLOR and SCRN:

```
FOR x = 0 TO 15: POKE 18781 + x, x: NEXT
```

SmartBASIC V2.0 screen colors:

Here are the screen color control addresses for the Coleco's public domain version of the interpreter:

17184 = TEXT default background
17240 = TEXT NORMAL fonts
17251 = TEXT INVERSE fonts

16783 = HGR graphics window
24784 = GR graphics windwo

24695 = GR and HGR default background
24847 = GR TEXT and HGR TEXT fonts

SmartBASIC V2.0 graphics color tables:

The value of the current GR COLOR is at address 17111. The value of the current HGR HCOLOR is at address 16776. You can POKE a 128 into address 16776 to erase points (3PLOT) using the HPLOT command.

You can correct the COLOR, HCOLOR, and SCRN commands so that they use the master color code values. You'll only need to learn one set of color values. Here's how ...

To correct HCOLOR:
FOR x = 0 TO 15: POKE 25360 + x, x: NEXT

To correct COLOR and SCRN:
FOR x = 0 TO 15: POKE 25378 + x, x: NEXT

SmartBASIC 2.0 SPRITE CONTROL

One of the more powerful enhancements that this latter version offers over the original is its built-in sprite command. BASIC 2.0 cleverly uses the DRAW command to position sprites. Address 16788 is used as a flag byte. When its value (determined by PEEKing) is zero, the DRAW command can be used for normal hi-res shapes. When its value is NOT zero (1 to 255, inclusive), the DRAW command is designated for sprite animation.

BASIC 2.0 comes with two default sprites: a sailboat and a Star Trek Enterprise. This small bit image table occupies 64 bytes starting at address 192. You can setup your own bit image table at any unused are of RAM. Addresses 16786 and 16787 are the low and high byte pointers to the starting address of the bit image table.

SmartBASIC 2.0 sprite bugs:

For some unknown reason, 2.0 sets the default vertical position of all sprites to "208". This means that you have to make the sprites appear on the screen in numerical order: you can't simply turn on the twelfth sprite, for example. A simple POKE correction allows you to turn on the sprites any sequence that you deem fit. It is ...

POKE 17229, 200: TEXT

Also, 2.0 will not let you set the vertical coordinate of sprites to either of the two special positions (for turning off sprites either individually or collectively). This makes it near impossible to remove a sprite once it's drawn. Here's how to correct this designer oversight ...

POKE 11943, 208 (in STDMEM)
POKE 12454, 208 (in EXTMEM)

Sprites are drawn in the current HCOLOR. Thus, you'll most likely want to use the HCOLOR correction (described above) before using sprites with 2.0.

Using SpritePOWER sprites with BASIC 2.0:

When you create your BASIC 2.0 sprite program, you'll need to setup the aforementioned addresses, set LOMEM, and BLOAD a SpritePOWER sprite set stored as a BASIC binary image file. Then, to move the sprites, just use the DRAW command. Consider this simple example.

```
100 LOMEM: 30270
110 POKE 17229, 200: TEXT
120 POKE 11943, 208
130 FOR x = 0 TO 15
140 POKE 25360 + x, x: NEXT x
150 PRINT " one moment please ..."
160 PRINT CHR$(4);"BLOAD geometry.b, A29696"
200 REM move the sprite
210 POKE 17240, 241: TEXT
220 HCOLOR = 7
230 FOR x = 0 TO 255
240 DRAW 1 at x, 80: NEXT x
```

Changing sprite size:

SmartBASIC 2.0 includes a short routine that allows you to change the size of sprites displayed on the screen (normal or double magnification). Here's how to use it ...

for instant sprite enlargement,
POKE 17339, 227: CALL 17338

for standard size sprites,
POKE 17339, 226: CALL 17338

THE HINKLE SPRITE COMMANDS

In their detailed guide to SmartBASIC V1.0, The Hacker's Guide to ADAM: Volume Two, Peter and Ben Hinkle reveal how to set up four NEW sprite commands. These include: SETUP, DEFINE, SPDRAW, and BUMP. The SETUP command allows you to set the size and magnification factor for the sprite set. DEFINE allows you to create a bit image of the sprite designs. SPDRAW is used in similar fashion to the DRAW command to position sprites on the screen. BUMP is used to determine sprite collisions. The sprite's color is assigned with the current HCOLOR value.

To use a SpritePOWER sprite set with these commands, you'll need to store the file with the ASCII DATA option. Then you can LOAD the file and READ the bit image values (in the DATA statements) into the DEFINE command as numeric variables.

THE SpritePOWER SPRITE ROUTINES

In addition to the above mentioned sprite control options, the SpritePOWER medium also comes with a program entitled "SpriteDemo". The program is LISTed on pages 23 and 24 of this manual so that you may easily study how it works.

It uses three machine code routines for sprite control that we've donated into the public domain. The routines use a combined total of 110 bytes. In our BASIC example, they occupy addresses 27600 thru 27709, inclusive. These routines can be used with SmartBASIC 1.0, SmartBASIC 2.0 and any z80 machine code program (hackers will be interested to know that the routines are not address specific). In fact, these same routines are even used in the SpritePOWER program (sp.obj) itself.

The primary routine occupies addresses 27600 thru 27641, inclusive. This is the sprite setup routine. You must CALL 27600 after using a BASIC video mode command, ie, TEXT, GR, HGR, or HGR2. This routine performs four functions: it transfers the bit image data to the video chip, it sets up the VDP register pointers, it turns off any sprites that were already displayed, and it sets the magnification of the sprite set. Any time that you want to turn off all the sprites, just CALL 27624. You can change the size by POKEing into address 27636 (195 = double size, and 194 = standard size). Then, CALL 27635 to implement the size change.

If you're a hacker, you'll note that the NMI VDP interrupt is disabled with this routine. This is intentional. The interrupt routine at address 102 ($66) is too long (uses too much computer time -- in microseconds). Although the routine enables FLASHing fonts, it can cause bit image distortions on the video chip when sprites are used. This is particularly evident in the SmartBASIC V1.0 TEXT mode. Thus, the routine disables FLASH (until the TEXT command is used), but it eliminates any video distortion.

The next routine occupies addresses 27642 thru 27680, inclusive. This one is used to move the sprites and change sprite colors. POKE the vertical coordinate into address 27643. POKE the horizontal coordinate into address 27644. POKE the sprite number (1 thru 32, inclusive) into address 27646. POKE the sprite color into address 27647. Then CALL 27642. If you plan on moving the sprite often, you may want to POKE numeric variable values into these addresses rather than absolute numbers.

The third routine occupies addresses 27681 thru 27709, inclusive. This routine is used to locate sprites. To use it, just POKE the sprite number (1 thru 32, inclusive) into address 27682 and then CALL 27681. The four sprite attribute elements will be at address 54272. PEEK(54272) will reveal that sprite's vertical position. PEEK(54273) will reveal that sprite's horizontal coordinate. Address 54274 contains the sprite number (the number that you POKEd into addres 27682). And, PEEK(54275) will reveal the sprite's current color value.

The program demonstrates the use of these routines. If this is your first exposure to sprites, all this information may not be entirely lucid on the first reading. Don't worry. Try the program. Experiment. You can not do any permanent damage to ADAM with the program.

In your own BASIC programs, you will need to POKE the DATA values for the machine code routines into the proper addresses just as the program does. Be sure to enter each value correctly. Even one incorrect value could cause your system to lock up (you'd have to reboot SmartBASIC) when you CALL a routine's starting address.

Welcome to the exciting new dimension of sprite usage ...

LIST of SpriteDemo ...

```
 10 REM sprite usage demo
 20 LOMEM :30720
100 REM sprite setup routine
110 DATA 33,0,116,17,0,56,1,0,4,205,26,253
120 DATA 1,62,5,205,32,253,1,7,6,205,32,253
130 DATA 62,200,17,128,0,33,0,31,205,38,253
140 DATA 1,194,1,205,32,253,201
150 FOR x = 27600 TO 27641: READ mc: POKE x, mc: NEXT
151 REM CALL 27600 to setup sprites after TEXT, HGR, HGR2, or GR
152 REM CALL 27624 to turn off all sprites
153 REM POKE 27636,195 for double size, or
154 REM POKE 27636,194 for standard size; then
155 REM then CALL 27635
200 REM sprite movement or color change routine
210 DATA 33,000,000,17,0,0
220 DATA 34,0,212,237,83,2,212,123,61,135,135,50,2,212
230 DATA 79,6,0,33,0,31,9,93,84,33,0,212,1,4,0,205,26,253,201
240 FOR x = 27642 TO 27680: READ mc: POKE x, mc: NEXT
241 REM POKE 27643, vertical position
242 REM POKE 27644, horizontal position
243 REM POKE 27646, sprite number
244 REM POKE 27647, sprite color
245 REM then CALL 27642 to effect changes
300 REM find sprite position routine
310 DATA 62,0
320 DATA 245,61,135,135,79,6,0,33,0,31,9,93,84
330 DATA 33,0,212,1,4,0,205,29,253,241,50,2,212,201
340 FOR x = 27681 TO 27709: READ mc: POKE x, mc: NEXT
341 REM POKE 27682, sprite number
342 REM then, CALL 27681 to get attribute data
343 REM after this, PEEK(54272)=vertical position
344 REM PEEK(54273)=horizontal position
345 REM PEEK(54274)=sprite number
346 REM PEEK(54275)=sprite color
400 REM let's try the routines
1000 POKE 17115, 241: TEXT: PRINT " one moment please ..."
1010 REM get the BSAVEd file into RAM
1020 PRINT CHR$(4); "bload misc01.b": HOME
2000 REM setup sprites
2010 CALL 27600
3000 REM turn on #1
3010 POKE 27643, 80: REM vertical position
3020 POKE 27644, 120: REM horizontal position
3030 POKE 27646, 1: REM sprite number
3040 POKE 27647, 11: REM sprite color
3050 CALL 27646: REM transfer the 4 parameters
3100 REM leave it there for three seconds
3110 FOR x = 1 TO 3*750: NEXT x
```

SpriteDemo LIST continued ...

```
4000 REM move it around
4100 ch = 112: cv = 64: rs = 50
4110 FOR y = 1 TO 3: REM number of circles
4120 pi = ATN(1)*4: radian = pi/180
4130 FOR point = 2*pi TO 0 STEP -radian*6
4140 ht = rs*SIN(point): vt = rs*COS(point)
4200 FOR z = 0 TO 3
4210 POKE 27643, cv+vt+z*18: POKE 27644, ch+ht
4220 POKE 27646, 1+z*2: POKE 27647, 7+z
4300 CALL 27642: NEXT z: NEXT point: NEXT y
4400 REM turn them off
4410 CALL 27624
5000 REM move sprite with joystick
5010 vt = 80: ht = 128: co = 3: nu = 12
5100 HOME: PRINT " use joystick to move"
5110 PRINT: PRINT " press <escape> to end"
5200 POKE 27643, vt: POKE 27644, ht
5210 POKE 27646, nu: POKE 27647, co
5220 CALL 27642
5300 js = PDL(5): IF PEEK(64885) = 27 THEN  END
5310 IF js = 1 THEN  vt = vt-2
5320 IF js = 4 THEN  vt = vt+2
5330 IF js = 8 THEN  ht = ht-2
5340 IF js = 2 THEN  ht = ht+2
5400 IF vt > 192 THEN  vt = 0
5410 IF vt < 0 THEN  vt = 192
5420 IF ht > 255 THEN  ht = 0
5430 IF ht < 0 THEN  ht = 255
5500 GOTO 5200
```

SpritePOWER

CHAPTER FIVE

ASSORTED PROGRAMS

Pix.MGR

The program entitled "Pix.MGR" allows you to store and retrieve hi-res pictures. It is a public domain donation from DIGITAL EXPRESS. And, it is designed for use with SmartBASIC V1.0. If you don't already have ShowOFF I or PowerPAINT by DIGITAL EXPRESS, you'll find this program useful for storing the hi-res picture files in SmartPAINT format (for use with SpritePOWER in the "capture" mode).

To use it, just boot your own SmartBASIC V1.0 and enter BRUN Pix.MGR. It automatically determines if you are already in HGR mode. If you are, it won't clear the screen. This way you can BRUN the program after drawing a nice HGR picture without disturbing your graphics.

You can store and retrieve screens plus some other options. The LOAD feature allows you to LOAD a standardized RLE (Run Length Encoded) picture, a SmartPAINT picture, or a 10K picture file previously stored with the program. Due to the RLE compatibility, you can set the background and HPLOT colors. With the SAVE feature, you can store the picture in SmartPAINT format or as a single 10K file. With the SmartPAINT option, you can also clear the area that SmartPAINT uses as a title bar.

The program also allows you to change colors already on the screen. You can change all of one HPLOT color to another color of your choice. And, you can change the total background color without disturbing the HPLOT colors. Unlike our commercial graphics programs, these color changes are accomplished primarily from BASIC which makes the process a little slow.

PUFF

PUFF is a fast action, arcade style game (written in BASIC) which uses sprites for animation. The game is played in simultaneous opposition; with the one player option you compete against ADAM. Most user input is accepted through the game controller (keypad, joystick, and triggers). You can tap the <ESCAPE> key at any point to restart the program. Hi-res graphics, sound effects, and simple melodies enhance the program. The first time you play it, you should select option #3 (from the keypad) to view the pages of instructions. At the end of each page, just tap any keyboard key to continue.

This program provides a good illustration of sprite usage, reveals some BASIC programming tricks, and can offer many hours of enjoyment FOR THE WHOLE FAMILY.