SuperSprite™

OWNER'S MANUAL

SuperSprite

by

Synetix

15050 N.E. 95th St.
Redmond, Wa 98052

Preface

This manual is organized into four sub-manuals, each covering certain aspects of the SuperSprite or its supporting software.

The first portion of the manual is the Installation and Technical Reference section. Please read this section carefully before and during installation of the SuperSprite.

This portion of the manual contains:

- Precautions to observe before installing the SuperSprite
- Directions on installation of the SuperSprite.
- Directions on testing the SuperSprite.
- Troubleshooting tips.
- Advanced programming information.

The second portion of the manual describes the StarSprite I system. You should be sitting at your computer when you read this document. It will lead you through the Ampersprite language, and the rest of the utilities that comprise the StarSprite I system.

The third portion of the manual concerns the Echo][Software. This document is a tutorial on making your SuperSprite speak. The information necessary for advanced programming applications with the speech chip is also contained in this portion of the manual.

The last portion of the manual describes Echo Words, a vocabulary of over 700 natural sounding words for the Echo][portion of the SuperSprite. Both a tutorial, and technical information for the advanced programmer are contained in this section.

THIS PAGE LEFT INTENTIONALLY BLANK

TABLE OF CONTENTS

THIS PAGE LEFT INTENTIONALLY BLANK

1.0 BEFORE YOU BEGIN

1.1 Introducing The SuperSprite

Congratulations, and thank you for buying the SuperSprite. We feel that it is the best product of its kind on the market, and we are about to prove it to you!

The SuperSprite is a peripheral card for arcade style games, educational applications, and many other uses. It simultaneously synchronizes:

- conventional Apple video (for example, graphics)
- animated sprite graphics (sprites are large programmable characters that move independently on the screen)
- sound effects
- synthesized speech (using the Echo II speech synthesis circuitry and software)

The SuperSprite comes with a new language called Ampersprite that transforms standard Applesoft into a language that lets you program sound effects and musical chords with simple commands. Using Ampersprite, you can:

- design sprites of various sizes, shapes, and colors
- sequence their movement
- program sound

1.2 What This Manual Covers

So you know where to find information in this manual, read this brief summary of each section:

- **Section 1** introduces SuperSprite and this manual, including warranty information and a quick "get started" procedure for experienced Apple users.
- In **Section 2**, everyone should read about unpacking the SuperSprite. For novices, the section includes an introduction to Apple basics (Sections 2.3, 2.4, and 2.5).
- **Section 3** discusses SuperSprite features, including graphics, sound, and speech.
- **Section 4** tells how to install the SuperSprite. Be sure to follow the handling precautions detailed there.
- After you have installed the SuperSprite, perform the checkout test covered in **Section 5**.
- In the unlikely event that you have trouble, **Section 6** helps you pinpoint the cause and correct the problem.
- **Section 7** is a reference guide for those interested in the advanced programming of the SuperSprite.

1.3 A Quick Procedure To Get Started

Whether you are a novice or a seasoned Apple user, we have made this manual convenient for you. If you are a new Apple user, we recommend that you work through the entire manual. However, experienced users can follow this quick procedure to get started:

1. Unpack the SuperSprite.
2. Complete and mail the warranty card.
3. Read Section 3 to learn all the SuperSprite's features.
4. Read the installation directions in Section 4.3.
5. Run the checkout test in Section 5.

1.4 Warranty Information

Please fill out your warranty card and send it in **immediately.** The warranty registers you for:

- software updates
- manufacturer support
- documentation updates
- additional product information

2.0 GETTING STARTED

2.1 What Is Included With The SuperSprite

Your SuperSprite Installation Kit should include these items:

- the SuperSprite printed circuit board
- a monitor cable
- a speaker
- this SuperSprite Installation and Technical Reference Manual
- StarSprite I Manual
- StarSprite I system disk, double-sided
- StarSprite I Demonstration disk/Checkout Software (double-sided)
- Echo][Software/Echo][Words (double-sided)
- warranty card

Please check your installation kit carefully for all these items. If anything is missing, please call Synetix Customer Service at (800) 426-7412.

2.2 What You Will Need

To use the SuperSprite you will need:

- an Apple][, Apple][Plus, or Apple //e (Revision B motherboard) computer with 48K minimum memory. **If you have an Apple //e and do not know whether it is Revision A or B,** please contact your Apple dealer.

  Note: Some of the Echo II software requires 64K of memory to work properly.

- one or more disk drives, with controller cards set up to read 16-sector "floppy" diskettes

- Disk Operating System (DOS) 3.3

- a 75-ohm composite video monitor and connecting cable, or an ordinary home television set with a radio-frequency (RF) modulator

  Note: If you are using a television set, your Apple dealer can provide you with an RF modulator and show you how to connect it. Should you decide to buy an RF modulator, experience has shown that more expensive modulators produce a better picture.

- Applesoft in RAM or ROM

- an external amplifier, such as a stereo (this is optional)

2.3 Setting Up Your Apple Computer

If you have just unpacked your Apple computer, here is how to set it up to use the SuperSprite.

1. Connect one end of the computer's power cord to the power connector at the left rear of the computer, and plug the other end into a grounded (3-prong) power outlet.

2. Install the disk drive(s) by following the instructions in Chapter 2 of the Apple Disk Operating System (DOS) Manual that came with the disk drives. Or follow the instructions provided by the manufacturer.

After completing these installation steps, you are ready to use the SuperSprite.

2.4 Handling Diskettes

With one or more disk drives, your Apple can store and retrieve information on 5 1/4-inch "floppy" diskettes made of flexible plastic. This flexibility distinguishes them from harder "rigid" disks. Each diskette is a mass-storage medium that can hold 143 kilobytes (143K) of information.

Diskettes are a very convenient way of storing information, but they have to be handled with respect. Each diskette comes sealed in its own black jacket. However, they still are sensitive to heat, dust, smoke, scratches, fingerprints, and magnetic fields.

When handling disks, follow these precautions:

- Never take a diskette out of its jacket.
- Never bend it (much less fold, spindle, staple, or otherwise mutilate!).
- Never touch its surface through any of the holes in the jacket.
- Once the label is attached to the diskette, use only a felt-tip pen to write on the diskette label--and do not press too hard!

2.5 Making Backup Copies

It is important to make backup copies of your StarSprite I system disk, demonstration disk, and Natural Speech/Checkout Test software disk. Be sure to copy all six sides. If you do not know how to make backup copies, see your Apple Disk Operating System (DOS) Manual for instructions.

3.0 SUPERSPRITE FEATURES

3.1 Graphics Capabilities

The SuperSprite uses the Texas Instruments TMS9918A Video Display Processor (VDP), which lets you create, display, and manipulate sprites of various sizes, shapes, and colors. You can program up to 32 different sprites and coordinate their movement into fast and smooth animation.

You can choose sixteen standard colors (including black and transparent) for sprites or patterns. Additional colors may be created by mixing two or more standard colors.

Sprites are far easier to define and manipulate than conventional high-resolution graphics images. They move independently of the background. With single commands, you can change these sprite attributes:

- color
- horizontal (X) position on the screen
- vertical (Y) position on the screen
- resolution (number of pixels)
- magnification (1:1 or 2:1)

3.2 Sound Capabilities

The SuperSprite uses the General Instruments AY-3-8912 Programmable Sound Generator to produce two types of sound: tones and white noise. You can modify the sound to produce a variety of sound effects and musical tones.

The sound generator may be programmed to activate the sound effects with a minimal amount of processor time. This makes the effects happen smoothly and quickly.

Furthermore, with the programmable envelope generator that is part of the sound generator, you can change the amplitude of the sound wave over time. This results in effects such as piano-like tones and explosions that fade in volume.

To create and modify sounds, choose any of 16 programmable filters, which are similar to a filter on a stereo. The sound generator has three types of filters:

- The **high-pass filters** eliminate tones below a given frequency, producing brighter sound quality.
- The **low-pass filters** eliminate tones above a given frequency, producing a softer tone.
- The **band-pass filters** eliminate tones above and below a given frequency range, producing a pure tone centered around the given frequency.

3.3 Speech Capabilities

On the SuperSprite, the Texas Instruments TMS5220 speech chip uses the Echo II Speech Synthesizer to generate two basic forms of speech:

- natural sounding, or digitized, speech (often called fixed speech)
- robotic, or phoneme-generated, speech

Natural or fixed speech may be generated from a vocabulary of more than 700 words, which are supplied with the SuperSprite. This type of speech sounds almost like a human voice.

Custom-encoded vocabularies may be ordered to suit your individual needs.

Phoneme-generated speech sounds more robotic, but has the flexibility of being able to say virtually anything. A text-to-speech program supplied with the SuperSprite converts the text printed by an ordinary Applesoft PRINT statement to speech (you can even list your programs out loud!).

4.0 INSTALLING THE SUPERSPRITE

4.1 Handling Precautions

This section tells how to install your SuperSprite. Before you do, please read the following precautions carefully. You must understand them before you handle the SuperSprite.

The SuperSprite is sensitive to static electricity. To ensure a long service life, observe these precautions:

1. Before handling the SuperSprite, always ground yourself by touching the power supply case of your Apple or a large metal object. This is especially important in carpeted areas.

2. When the SuperSprite is out of your computer, always wrap it in its protective anti-static envelope.

Warning: Always turn the computer's power off before you insert or remove your SuperSprite from your Apple. Failure to do so voids the warranty and could severely damage your SuperSprite, your Apple, or both.

4.2 How to Install the SuperSprite

Please refer to Figure 1 for the locations of the cable jacks and adjustment controls.

Note: When connecting monitor cables to the SuperSprite, support the circuit side of the board with the palm of one hand. Then firmly insert the cables with a twisting motion into the cable jacks. Never rock the cable back and forth into the cable jack, or you may damage the SuperSprite.

Before you can install the SuperSprite, you must move your television or monitor off to one side of your Apple. Then remove the Apple's top cover for access to the motherboard. Now you are ready to install the SuperSprite.

1. Connect the monitor and speaker cables to the SuperSprite:

   a. Connect one end of the video monitor cable to the SuperSprite **video in** jack.

   b. If you are using an external amplifier, such as a stereo, connect the pre-amp output cable to the **pre-amp output** connector on the SuperSprite.

   c. Bring the speaker cable through the rear panel, and connect it to the SuperSprite **speaker** jack.

   d. Connect the monitor or television to the SuperSprite:

      If you are using a monitor, connect the Apple Monitor cable to the monitor jack on the SuperSprite.

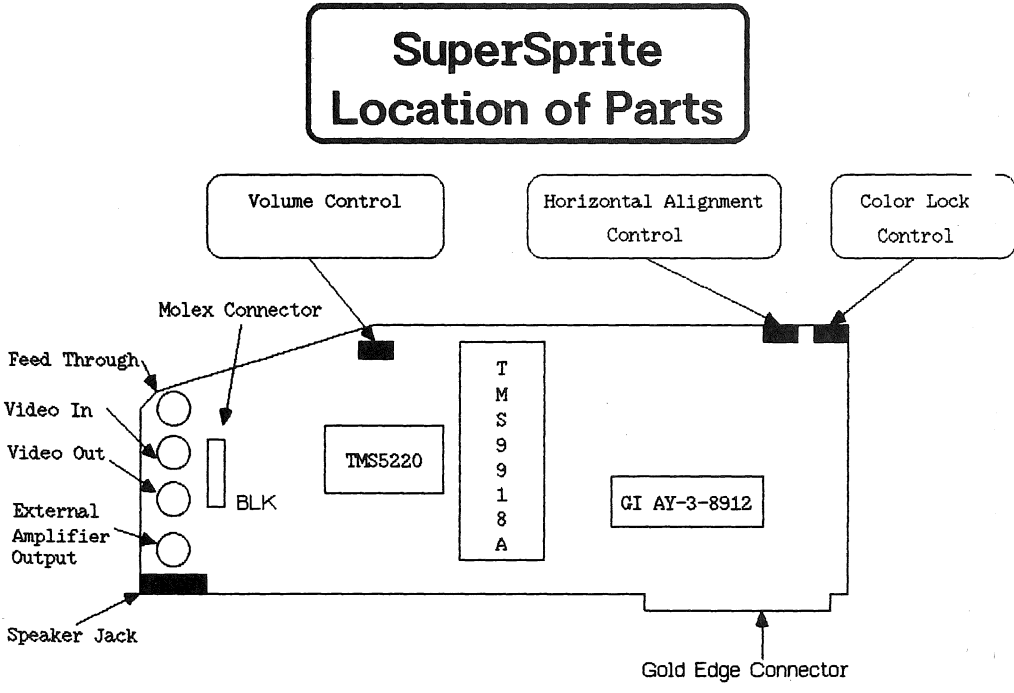


Figure 1. SuperSprite Location of Parts

If you are using a television with an RF modulator, connect the RF modulator molex connector to the SuperSprite so that the black wire is closest to the bottom (connector edge) of the board.

Warning: Make sure that the RF modulator molex connector is properly connected with the black wire toward the **BLK** marking on the SuperSprite board. Connecting the RF modulator incorrectly may damage the board and the RF modulator.

2. Now install the SuperSprite in your Apple:

   a. Guide the monitor cables behind the SuperSprite and out the back of the Apple.

   b. Carefully insert the SuperSprite into slot 7, which is the right-most slot on the Apple motherboard. To fully insert the board, you may have to gently rock the board as you press it into the slot.

   Note: Be careful not to bend or break any protruding components along the top edge of the SuperSprite.

3. Connect the monitor cables to your Apple's rear panel and your monitor:

   a. Plug the cable from the SuperSprite **video out** jack into the video in jack of your monitor.

   b. Plug the cable from the SuperSprite **video in** jack into the video out jack of your Apple.

Note: Make sure that all the cables are securely connected in the proper location at both ends before turning the power on.

4. Before turning the power to your Apple on, please make certain that you have replaced the cover, and have secured it in position. If you should need to remove the cover from your Apple, make absolutely certain that the power is off. This ensures that incidental contact between the SuperSprite and the cover of the Apple will not cause any problems.

5.0 CHECKOUT TEST

5.1 Starting The Test

Once you have all the necessary equipment installed and connected properly, you are ready to start the Checkout Test. Follow this procedure carefully. Then, if your Apple is not equipped with the Autostart ROM feature, you will have to do a few extra steps.

To start the Checkout Test:

1. Turn on your video monitor.

2. Open the door to the main disk drive. (The main drive is drive 1, in the highest-numbered peripheral slot containing a disk controller card. By convention, this is slot 6.)

3. Carefully insert your Natural Speech/Checkout Test software disk in the disk drive:

   a. Hold the diskette with your thumb on the label--the label reading CHECKOUT DISK should be facing up.

   b. Push it gently into the disk drive as far as it will go.

4. Gently close the disk drive door.

5. Turn on your Apple computer by pressing the power switch. This rocker switch is located on the back of the computer, near the left side.

If your Apple is equipped with the Autostart ROM feature, the computer will beep, the red light labeled IN USE on the front of the disk drive will come on, and the disk drive will start whirring and chattering. (Don't be concerned by these noises -- they are perfectly normal.) After 5 to 10 seconds, the IN USE light will go off, and the noises will subside. You are ready to continue to Section 5.2.

WARNING: IF THE LIGHT DOES NOT COME ON AND THE APPLE DOES NOT BEEP, TURN OFF THE APPLE IMMEDIATELY!!

If your Apple does NOT have the Autostart ROM feature, it simply beeps, then displays an asterisk (*) near the bottom of the screen, followed by a blinking white box called the cursor. To finish starting up the Checkout Test, you must:

1. Press the RESET key.

2. Type the number of the peripheral slot to which the main disk drive is connected (usually 6).

3. Type the combination keystroke, CTRL-P. (Hold down the CTRL key while typing the letter P.)

4. Press RETURN.

The disk drive should then start up, as described above. Now you are ready to continue to Section 5.2.

WARNING: IF THE LIGHT DOES NOT COME ON AND THE APPLE DOES NOT BEEP, TURN OFF THE APPLE IMMEDIATELY!!

5.2 The Checkout Test Menu

If all has gone well, your screen should look like this:

```
PLEASE SELECT A TEST:

1. Overlay Test
2. Horizontal Alignment Test
3. Video RAM WR/RD Verification Test
4. Interrupt Test
5. Sound Test
6. Speech Test
7. Exit Test Program

DURING ANY TEST PRESS SPACE BAR TO CONTINUE

ENTER A NUMBER:
```

This is a "menu" screen from which you may select what you want to do. The first six items on this menu are tests. The last selection ends the test program.

Select each menu item in the order listed. To select a test, type the number of the test you want to do, then press the RETURN key on the Apple. When you finish a particular test, press the space bar to continue. You'll automatically return to the menu screen so you can select another test.

Note: This section mentions several adjustment **controls**, which are small adjustment screws on the SuperSprite. For the exact location of each control, please refer to Figure 1. We recommend using a small plastic screwdriver for all adjustments.

5.3 Test 1: Overlay Test

Before beginning this test, adjust the tint and color controls on your television or monitor to their middle positions.

This test asks you to fine-tune the color bar pattern which appears on your screen:

Adjust the tint, color, and brightness (sometimes called black level) controls on your television or monitor until the fourth color bar from the left is dark green and the brightness is comfortable for you.

If the colors are accurate and the picture is locked, press the space bar to continue with the test menu.

If you cannot properly adjust the color, or if the picture drifts to the right or left across the screen, adjust the **color lock** control (which is the second adjustment control from the back of the board) until the picture stops moving and the fourth color bar is as close to dark green as possible.

Press the space bar to return to the test menu.

5.4 Test 2: Horizontal Alignment Test

This test asks you to adjust the **horizontal alignment** control which is the control closest to the back of the board. Usually, this will not be necessary. However, if your picture is not centered on the screen, follow this procedure:

1. Slowly adjust the horizontal alignment control until the screen is centered horizontally. During this test, the picture normally will jump around slightly on the screen, and the speaker will buzz.

2. When the picture is centered, press the space bar to return to the test menu.

5.5 Test 3: Video RAM WR/RD Verification Test

This test checks all video RAM for integrity. After you select Test 3:

- The display will flash blue and yellow (all video RAM is filled with test patterns).

- Several random sprites will appear on the display.

If the screen displays the message, "ALL VIDEO RAM OK," press the space bar to return to the test menu.

However, if the test detects a suspect chip on the SuperSprite, the display will tell you to:

1. Turn off the power to the Apple.

2. Reinstall the SuperSprite.

3. Repeat this test.

4. If the problem continues, call Synetix Customer Service at (800) 426-7412.

5.6 Test 4: Interrupt Test

The SuperSprite generates IRQ interrupts (maskable interrupt requests) at a rate of 60 Hz. These interrupts correspond to the end of a video frame. Many programs will not use this feature. But if you have one that does, you can run this test to make sure the SuperSprite is generating the interrupts as expected.

The sound tone and changes in the backdrop color are all handled on an interrupt-driven basis. Therefore, during the test, four sprites move rapidly on the screen as a tone sounds in successively lower pitches and the backdrop color changes.

However, if the interrupts are not functioning properly, the tone may not sound or change or the backdrop color may remain transparent. You cannot fix this problem. Instead, call Synetix Customer Service at (800) 426-7412.

Press the space bar to return to the test menu.

5.7 Test 5: Sound Test

The Sound Test checks for proper operation of the noise and tone generators, the 16 programmable filters, and the envelope generator. You can adjust the sound only during the tone test--all other adjustments must be done at the factory.

1. Noise Test. During the noise test, listen for a hissing sound.

2. Tone Test. A tone (decreasing in pitch) should sound each time you see the words, "Tone Generator A," "Tone Generator B," and "Tone Generator C," on the screen. As needed, slowly adjust the **volume level** control, which is the control closest to the keyboard.

3. Filter Test. A tone should sound each time a number appears on the screen. If you do not hear a tone for any given filter number, check the volume control of your television or monitor, then try this test again.

4. Envelope Test. During this test, listen for a decaying tone.

If any test does not perform properly, please call Synetix Customer Service at (800) 426-7412.

Press the space bar to return to the test menu.

5.8 Test 6: Speech Test

During this test the computer will talk to you and tell you what to do.

If you do not hear the computer's voice, check the volume level control on your SuperSprite, and repeat this test. If this problem persists, call Synetix Customer Service at (800) 426-7412.

Press the space bar to return to the test menu.

5.9 End of Test

When you select choice 7 on the test menu, the "]" prompt should appear. This prompt signals the end of the Checkout Test.

6.0 HAVING TROUBLE?

If you have a problem getting the SuperSprite to work with your Apple, use this table to figure out what to do. If any problem persists, check your Apple.

PROBLEM	PROBABLE CAUSE	WHAT TO DO
Power indicator does not light, or Apple does not beep when you turn on the power.	Molex connector to RF modulator upside down.	1. Power down immediately!! 2. Check the RF modulator connection, and correct if needed. 3. Remove the SuperSprite, and check the installation instructions in Section 4.0.
Horizontal drift	SuperSprite is not locked to Apple video.	1. Press CTRL and RESET at the same time. 2. Adjust the color lock control slowly and carefully until the fourth color bar from the left is dark green.
Apple video is not centered on the screen	The horizontal alignment control is out of adjustment.	1. Run the test software. 2. Select the Horizontal Alignment Test, and follow the instructions in Section 5.2.
The speaker buzzes.	Speaker cable is not firmly connected.	Reconnect the speaker cable.

PROBLEM	PROBABLE CAUSE	WHAT TO DO
You hear no sound.	The volume is too low.	Increase the volume control on your monitor or television.
You see no video.	The cables are not firmly connected, or they are connected to the wrong cable jack.	1. Check the cable connections between the Super-Sprite and the Apple; correct if needed. 2. Increase the brightness and contrast controls on your television or monitor. 3. If you still have no video, power down immediately!

7.0 TECHNICAL REFERENCE GUIDE

7.1 SuperSprite Slot Position

The SuperSprite can only work in the Apple motherboard's slot 7. This is because the composite sync (pin 19) and 3.579 MHz color reference (pin 35) appear only in slot 7 of the Apple. These two signals are necessary for proper operation of the video circuitry.

7.2 Addressing

Please refer to the following table for addressing in slot 7.

DECIMAL	HEX	CHIP ADDRESSED	FUNCTION
49407	$C0FF	Sound	REGISTER WRITE OR
49406	$C0FE	Sound	DATA READ
49405	$C0FD	Sound	DATA WRITE
49404	$C0FC	Sound	DATA WRITE
49403	$C0FB	unused	
49402	$C0FA	unused	
49401	$C0F9	unused	
49400	$C0F8	unused	
49399	$C0F7	VDP	WRITE ONLY/FRAME RESET
49398	$C0F6	Video Switch	MIX VDP/EXTERNAL VIDEO
49397	$C0F5	Video Switch	APPLE ONLY OUT
49396	$C0F4	Video Switch	APPLE VIDEO IN ON
49395	$C0F3	Video Switch	APPLE VIDEO IN OFF
49394	$C0F2	Speech	DATA WRITE/STATUS READ
49393	$C0F1	VDP	REGISTER WRITE
49392	$C0F0	VDP	VRAM READ/WRITE

Table 1. Slot 7 Addressing

7.3 Interrupts

The TMS9918A (VDP) generates interrupts at a rate of 60Hz. These interrupts correspond with the end of each video frame. The interrupt line on the VDP is tied to the IRQ line of the Apple bus, which allows you to take advantage of maskable interrupts in your programs.

You should use interrupts for these reasons:

- Interrupts happen on a regular basis. This allows certain programming events to be timed, such as musical tone generation, sprite collision checking, or keyboard polling.

- Interrupts are generated at the end of a frame. This is also when the status register of the VDP is read. For more information on the status register, please see the Texas Instrument TMS9918 manual.

To enable interrupts, follow these steps:

1. Enable VDP interrupts by setting bit 5 of register 1. For example:

```
ENABLE9   LDA  OLDREG1    ;KEEP A COPY OF REGISTERS IN APPLE RAM
          ORA  #$20       ;SET BIT 5 IF NOT ALREADY SET
          LDY  #$01       ;REGISTER 1
          STA  VREG       ;STORE DATA
          STY  VREG       ;STORE REGISTER NUMBER
          RTS
```

2. Allocate the interrupt in the Apple. Under DOS 3.3, this is done by setting the interrupt vector at location $3F4.

```
ALLOC     LDA  #<IRQHAND ;LSB OF YOUR INTERRUPT HANDLER ADDRESS
          STA  IRQVECT   ;IRQ VECTOR AT $3F4
          LDA  #>IRQHAND ;MSB OF YOUR INTERRUPT HANDLER ADDRESS
          STA  IRQVECT+1
```

3. Set the "enable interrupts" flag on the 6502.

```
ENABLE6   CLI             ;ENABLE INTERRUPTS ON 6502
```

To write an interrupt handler, it is important that you follow these steps:

1. Preserve the contents of the X and Y registers.

2. Read the VDP register location (acknowledge interrupt).

3. Perform interrupt processing.

4. Restore the contents of the X and Y registers.

5. Restore the contents of the accumulator from location $45.

6. Return.

```
IRQHAND   TXA           ;PRESERVE THE X REGISTER
          PHA           ;ON THE STACK
          TYA           ;PRESERVE THE Y REGISTER
          PHA           ;ON THE STACK
          LDA  VREG     ;READ THE VDP REGISTER
          .
          .      some user processing
          .
          PLA           ;RESTORE THE
          TAY           ;Y REGISTER FROM THE STACK
          PLA           ;RESTORE THE
          TAX           ;X REGISTER FROM THE STACK
          LDA  $45      ;RESTORE THE ACCUMULATOR
          RTI
```

Please observe the following precaution: Interrupts must be masked on the 6502 before any reads or writes to video RAM or VDP registers. Otherwise, interrupts may interfere with the synchronization between the 6502 and the VDP.

Note: It is unnecessary to disable interrupts on the VDP.

```
TALKVDP     SEI           ;MASK INTERRUPTS
            .
            .    interaction with video RAM or registers
            .
            CLI           ;ENABLE INTERRUPTS
```

7.4 Graphics Circuitry Description

The Texas Instruments TMS9918A Video Display Processor (VDP) is one of three important chips in the SuperSprite circuitry. As a slave central processing unit, it is capable of generating color composite graphics independent of the host processor. It has 16K bytes of RAM dedicated solely to video generation. The video RAM appears to be static to the Apple since the VDP handles all refresh operations in the video RAM.

7.5 Sound Circuitry Description

Sounds are generated by the General Instruments AY-3-8912 Programmable Sound Generator (PSG), which is the second of the three SuperSprite circuitry chips. It uses three separately addressable tone generators, each capable of producing tones in the 15 Hz to 63 KHz range with a 12-bit tone control resolution.

The amplitude (volume) or envelope shape for each tone generator is independently controllable. There is also a tone source. The three tones and noise source can be selected in any combination using the register 7 mixer. The volume of each tone channel is software selectable, using one of 16 programmable filters. You may also select one of ten envelopes.

To select filters, follow these steps:

1. Add 64 to the contents of register 7. This is required only the first time.

   Note: If you don't add 64, the default is filter 15 selected.

2. Store a filter number into register 14. Refer to the Table 2 for the contents of register 14.

FILTER NUMBER	FILTER TYPE	CORNER FREQUENCY
0	High Pass	400 Hz
1	Low Pass	400 Hz
2	Band Pass	400 Hz
3	No Filter	400 Hz
4	High Pass	800 Hz
5	Low Pass	800 Hz
6	Band Pass	800 Hz
7	No Filter	800 Hz
8	High Pass	1600 Hz
9	Low Pass	1600 Hz
10	Band Pass	1600 Hz
11	No Filter	1600 Hz
12	High Pass	3200 Hz
13	Low Pass	3200 Hz
14	Band Pass	3200 Hz
15	No Filter	3200 Hz

Table 2. Register 14 Contents

To program the sound generator, follow these steps:

1. Store the desired register number (1 through 15) in $C0FF (49407).

2. Store data to that register in $C0FD (49405). The data is then automatically routed to the preselected register.

   Note: This step may be repeated for continued access to the same register.

Note: Data is stored to these locations by using the POKE command from Applesoft, or any of the store commands from machine language.

For example, to produce a tone you would store one of the following registers in $C0FF (49407), and store its contents in $C0FD (49405).

REGISTER	CONTENTS	RESULT
0	100	Fine-tune the tone
7	62	Tone A only (no filters)
8	15	Maximum volume on Tone A

The registers on the sound generator function as follows:

REGISTER	FUNCTION
0	Fine tune for channel A frequency
1	Coarse tune for channel A frequency
2	Fine tune for channel B frequency
3	Coarse tune for channel B frequency
4	Fine tune for channel C frequency
5	Coarse tune for channel C frequency
6	Sound period for noise generator
7	Mixer (0-63 disables filtering)
8	Volume for channel A
9	Volume for channel B
10	Volume for channel C
11	Envelope period fine tune
12	Envelope period coarse tune
13	Envelope shape
14	Filter number

Table 3. Sound Registers

The way tone period (pitch) is specified for a given channel is by storing data to a register pair (for example, registers 1 and 2). These registers correspond to the fine-tune and coarse-tune settings for the PSG.

To calculate the proper values for a particular register to produce a given pitch, use the formula:

Tone Period = clock frequency / 16 X desired pitch

so, on the Apple, with a clock speed of 1.023 MHz:

Tone Period = 1,022,727 / 16 X desired pitch

= 63,920.438 / desired pitch

So if you want to calculate the tone period for the A above middle C, which has a pitch of 440 Hz, you would simply perform the calculation:

Tone Period = 63,920.438 / 440

= 145

This value is less than 255, and is specified by storing it in the fine-tune register, and storing a zero in the coarse-tune register. As the desire pitch becomes lower, the coarse-tune register will have to be utilized. The values for the two registers may be calculated with the BASIC program:

```
10 TEXT : HOME : INPUT "WHAT IS THE DESIRED PITCH (IN HZ)? ";P
20 TP = INT(63920.438 / P)
30 CT = TP - INT(TP/255) * 255
40 FT = TP - CT * 255
50 PRINT "THE COARSE-TUNE TO PRODUCE A PITCH OF ";P;" IS: ";CT
60 PRINT "THE FINE-TUNE IS: ";FT
70 PRINT "THE OVERALL TONE PERIOD IS: ";TP
80 PRINT : INPUT "CALCULATE ANOTHER SET OF VALUES (Y/N)? ";A$
90 IF A$ = "Y" OR A$ = "y" THEN 10
100 END
```

The mixer register (register 7) determines whether tone, noise, or both are enabled on a given channel. In addition, the mixer register determines whether or not filtering will be enabled. The following table gives the tone enable/disable settings:

Register 7 Value	Channel Enabled
0	A,B,C
1	B,C
2	A,C
3	C
4	A,B
5	B
6	A
7	none

Table 4. Mixer Register Settings

To obtain the correct value for the mixer register, determine which channels are to produce tones, which are to produce noise, and whether or not filtering is desired. Use the following formula to determine the correct value for the mixer:

Mixer Value = Tone Setting + Noise Setting X 8

If you intend to use filtering, simply add 64 to the value calculated above.

So, if you wish to produce an unfiltered tone on channel A only, the tone setting is 6 and the noise setting is 7. Plugging these values into the above formula yields 62 (if you want filtering enabled, use 126).

The volume (amplitude) control registers (8 through 10) accept values between 0 and 15 inclusive. To specify full volume, store a 15 in the appropriate register, and to shut off a channel, store a 0 in the appropriate register.

A value of 16 specifies that volume will be controlled by the envelope generator.

Registers 11 and 12 are the envelope period control registers. Like the tone period registers, the envelope period control registers are paired. You may calculate envelope period with the following formula:

Envelope Frequency = Clock Frequency / 256 X Envelope Period

so,

Envelope Period = 3995.0273 / Envelope Frequency

(using the 1.023 MHz clock on the Apple)

For example, if you wished to produce a tone with a 300 Hz envelope period, you would specify an envelope period of 13.

Register 13 specifies the envelope shape. Only the least significant four bits of the register are recognized, and the following table briefly explains the various settings.

Decimal Value	Continue	Attack	Alternate	Hold
0	no	no	xx	xx
4	no	yes	xx	xx
8	yes	no	no	no
9	yes	no	no	yes
10	yes	no	yes	no
11	yes	no	yes	yes
14	yes	yes	yes	no
15	yes	yes	yes	yes

Table 5. Programmable Envelopes

When **hold** is set on, it limits the envelope to one cycle, holding the last count of the envelope counter (either high or low).

When **alternate** is set on, the envelope counter reverses count direction after each cycle.

When **attack** is set on, the envelope counter counts up, and when it is set off it counts down.

When **continue** is set on, the cycle pattern is as defined by the hold setting, otherwise the generator will reset to zero after one cycle, and hold at that count.

7.6 Speech Circuitry Description

The Texas Instruments TMS 5220 speech processor is the third of the three SuperSprite circuitry chips. It uses the Echo][circuitry, which models the human vocal tract using Linear Predictive Coding (LPC). Instead of storing the actual speech signal, only those parameters needed to describe each speech sound are stored.

The publications listed below may be of use to advanced programmers who are concerned with the timing constraints and theory of operation of the various hardware devices used on the SuperSprite.

Applesoft Reference Manual; Apple Computer, Inc., 1982, Cupertino, Ca.

Apple][Owner's Manual; Apple Computer, Inc., 1982, Cupertino, Ca.

Apple //e Owner's Manual; Apple Computer, Inc., 1982, Cupertino, Ca.

Leventhal, Lance: 6502 Assembly Language Programming; Osborne/McGraw-Hill, 1979, Berkeley, Ca.

TMS5220A Voice Synthesis Processor Data Manual; Texas Instruments, Inc., 1982, Houston, Tx.

TMS9918A/9928A/9929A Video Display Processors; Texas Instruments, Inc., 1982, Houston, Tx.

Programmable Sound Generator Data Manual; Microelectronics Division, General Instrument Corporation, 1981, Hicksville, NY

THIS PAGE LEFT INTENTIONALLY BLANK

StarSprite I

by

Don Fudge

Manual by: Don Fudge

ISBN: 930182-41-3 (book)
ISBN: 930182-42-1 (package)

THIS PAGE LEFT INTENTIONALLY BLANK

1. Purpose and Features of StarSprite I

The purpose of the StarSprite system is to provide software support for the SuperSprite board that is easy enough for computer novices, yet comprehensive enough for the expert.

User-friendliness has been the keyword all the way through the software creation project. The software has been written so anyone, from a child new to computers to a graphics expert purchasing SuperSprite and StarSprite to upgrade his or her Apple graphics and sound, will be able to relate to it, use it, and enjoy it immediately.

I sometimes refer to the entire set of StarSprite software as the StarSprite system. It includes StarSprite I (for beginners), StarSprite II (for intermediate programmers), and StarSprite III (for more advanced programmers).

Even though StarSprite I will work for beginners, it's also a must for **all** users of the SuperSprite board, regardless of past computer experience, since it has all the basic utilities, interface software, and routines to "get you going", as well as the Ampersprite language -- an enhancement to Applesoft that gives Applesoft programs the capability of accessing the various aspects of the SuperSprite.

You should be able to see exactly why SuperSprite and StarSprite I come together as one package: they are the two essential ingredients in the just-beginning revolution in Apple sound and graphics.

The speed, smoothness, and ease of use of sprite graphics is what microcomputer animation is all about.

2. Adjustment and Alignment Program

Included in your SuperSprite package is a Checkout Disk. This disk contains all the programs necessary to adjust your SuperSprite properly. Complete information on this disk is contained in the Installation and Technical Reference Manual.

Warning: Do not attempt to make **any** adjustments other than those described in the Installation and Technical Reference Manual.

For your convenience, there is a short alignment and adjustment program included on the StarSprite I demonstration diskette.

To run this program, boot the StarSprite Demonstration disk, and the Alignment/Adjustment program will immediately be loaded in.

Note: Boot is the computer term for reading in the operating system from disk. Don't let this term intimidate you -- all you have to do to boot a disk is place it in drive 1 and turn the power to the Apple on.

If your SuperSprite has not been adjusted, or if it needs some fine tuning, proceed as follows:

- Press the space bar to signify that you want to do some adjusting now (pressing the RETURN key indicates that you do not want to perform any adjustments).

- Read the entire screen. Your board must be in slot 7, and your computer must have the top off with no monitor or drives in the way, so you can reach the SuperSprite's adjustments.

- If you have not yet hooked up your SuperSprite, please do that according to the instructions in the Installation and Technical Reference manual before proceeding.

- The remainder of the adjustment program is self-prompting, and will lead you through a simple test procedure. If you want to perform a more extensive test, please refer to the instructions in the Installation and Technical Reference manual.

3. Speech on the SuperSprite

Echo II hardware, software, and documentation is included in this system. The documentation is included in the Echo II manual (contained in the SuperSprite package).

The Echo II software is on Side 2 of the Demonstration Disk, and the Echo II Natural Speech Vocabulary is on the reverse side of the Checkout Disk.

To try out the Echo II, simply boot the Demonstration Disk (Side 2) as described above, and listen. Many features of Echo II speech synthesis are demonstrated.

You'll find that making your SuperSprite board talk is simple and enjoyable, whether with Textalker, Speakeasy or Natural Speech.

4. An Introduction

A demonstration of the StarSprite I system is contained on side 1 of the Demonstration Disk.

You'll see video display processor (VDP) sprite graphics and sprite animation. You'll hear 3-tone-harmony tunes and sound effects -- the sound will happen concurrently with the graphics **but it will not slow down the graphics a bit.** You'll see a pastoral scene painted with our Sprite Painting program and saved as a VDP graphics scene. You'll see VDP graphics and Apple graphics together on the same screen.

You will see animation in which the entire screen contains large moving objects at the same time a song is being played -- and all this will be happening in Basic with the help of the Ampersprite language.

For further examples of this kind, run option 6 from the main program menu of the program disk (Side A) and choose option 1 in the submenu, which takes you to a Graphics Tutorial Using Ampersprite.

The demo disk also contains a game, StarSprite Arcade, which demonstrates machine language programs using SuperSprite. The animation is a bit more (for the speed and smoothness involved) than Apple machine language could handle.

Increasing the step value (how far the shape moves at a time) a bit would keep the animation perfectly smooth but get the animation moving so fast you'd see all sprites as a blur of speed. And yet, even with huge delay loops, and consequently slow animation, the sprites move perfectly smoothly and with no flicker or jerk.

If you've done much programming, you'll realize the significance of all this: the SuperSprite/StarSprite combination gives you a great deal of power, and avoids the usual flicker, jerk, and slowness problems inherent in normal Apple graphics.

5. Sample Games

Before trying any of the games in the StarSprite system, it is important that you have either game paddles, a joystick, or some other game I/O device plugged into the Apple.

Several other programs in StarSprite I have this requirement, and will be so documented.

5.1. StarSprite Arcade

From the main program disk, side A, use option 1 to play a game and then choose (1) StarSprite Arcade from the submenu. In the game hit 1 to play game as it is.

Read the command screen and begin playing.

Pressing game button 1 will give you double magnification.

Typing any number from 0-9 will change the color attribute of the sprite on the video plane whose number you hit. The color number will go increase, so if you type 3 and video plane 3's sprite is color 5 (light blue), it will change to 6 (dark red).

Typing "3" again will change the color to 7, then 8, and so on until 15 (white) changes to 0 (transparent) and you cycle through the colors again (see color number chart on the command card).

If you would like to alter the game's characteristics, you may do so by using option (2) alter game characteristics by answering questions. You will be able to make the game considerably different, and you can even save the new version with option 4 of the game menu.

If you create your own sprite tables and path tables with some of the utilities in the StarSprite I system, you will be able to use them in the game with option 3. You need no programming experience to use these utilities.

5.2. StarSprite Maze

Note: Before selecting StarSprite Maze, make certain game paddles, a joystick, or some other game I/O device is plugged into your Apple.

From the main program disk, side A, use option 1 to play a game and then choose (2) StarSprite Maze from the submenu.

In the game type 1 to play game as it is.

Even if you select the keyboard option, you must have a game I/O device hooked up.

If you'd prefer not to have to hit the 0 button to eliminate aliens you're on top of, or if you would like to change any other game characteristic use (2) Alter Game Characteristics by answering questions.

Pressing <Return> accepts the default value for any modifiable option.

If you create your own sprite and maze and direction tables using StarSprite I utilities, you'll be able to include them in your games using option 3. And no programming experience whatever is necessary.

5.3. StarSprite Aliens & Asteroids

From the main program disk, side A, use option 1 to play a game and then choose (3) StarSprite Aliens & Asteroids from the sub-menu. In the game, type 1 to play game as it is.

Read the command screen to learn how to tell the Apple what you want.

If you'd like to change game characteristics, use (2) Alter Game Characteristics by Answering Questions.

Press <Return> to accept default values. The eight different explosion shapes (sprites) referred to in the questions are the eight-sprite explosion sequence. An "explosion" causes these sprites to be displayed consecutively on the screen. **You** get to choose the colors of each sprite in the sequence.

The reason the "calculating" message flashes on the screen for a few seconds is that a new random number table gets created each time you play this game, so that no two games will ever be alike. These random numbers control the directions and starting places of each alien or asteroid.

You can put a background scene, in normal Apple graphics, behind the game's screen while you play, using the main game menu option 3. SCENE6502 is the name of a sample scene on the program disk.

If you create your own sprite tables or step tables (for alien/asteroid speeds) using StarSprite I utilities, you will be able to include them in your games using option 3. And no programming experience whatever is required.

6. Paint With Sprites

Note: You **must** have game paddles, a joystick, or some other game I/O device installed for Sprite Painting. We have found the Koala Pad (Koala Technologies, Inc.) to work particularly well with the Sprite Painting program.

From the main menu choose option 2 for Paint With Sprites and when the program loads read the command page.

The color palette that fills the right side of the screen has 16 squares, each one representing a VDP (Video display processor) color. The first two colors are invisible since 0 is transparent and 1 is black.

Choose option 1 in the main menu to Paint With Brushes.

Position the cursor over one of the colors with paddles or joystick and press button 0.

Next, move over to the sprite brushes and choose one by positioning the cursor over it and pressing button 0.

Now press <ESC> to switch to the painting screen.

Move the paddles or joystick and press button 0 to paint one brush's worth, or hold down button 0 and move paddles or joystick at varying rates of speed.

You can really get effects that resemble painting if you move slowly and carefully and use a "dot-cluster" brush.

An example of sprite painting is contained on the StarSprite I demonstration disk in the "butterfly" demo.

Once you've finished with a brush or color, press ESC to return to the palette and choose new colors or brushes.

You may mix colors by painting close to or overlapping previously painted areas.

If you would like to label your picture with text, select option 5 of the program menu (which you get to by hitting Q for quit in either the painting or the palette routine).

Read the command page, and use Return as a toggle to get you in and out of **word mode.** Word mode causes each letter to be to the right of the previous one unless you're at the far right and get "wraparound".

Wraparound is an effect that happens when you attempt to type past the right margin of the screen. The text wraps around to the next line (beginning on the left).

Use ESC to switch between upper and lower case. Hit button 0 when you have finished labelling the painting.

Saving or loading pictures (VDP type) and sprite saving are also available in the sprite painting program.

7. Sprite Making and Scene Inspection

From the main program disk, side B, choose option 3 for Sprite Making/Editing & Scene Examination.

In the submenu choose option 2 for Single Sprite Editor.

In the program choose option 5 to Load In Sprite Generator Table.

Specify the ALIENS for the sprite table name. There's no need for any disk switching.

When asked if you want all the sprites in the table at once, type Y for YES.

There are only 56 sprites in the table but specify 64 for the number anyway. It's okay to always give 64 for this, as unfilled sprites are displayed as blank.

Notice that the first six lines of sprites are animation sequences. In each sequence an alien shape goes through a series of subtle changes, finally cycling back to the first shape again. You too can easily create such sequences -- it takes no programming experience. Press the Space Bar.

Use option 6 to Place A Sprite From Table On Grid.

Specify 0 as the sprite number (the grid you saw went from 0 to 63 for a total of 64 sprites). Sprite 0 will be drawn on a large grid. Next to this grid the sprite will be displayed in its actual size as you edit it.

Remember that you can double this sprite's dimensions and quadruple its area with one short Ampersprite command (&RM1) at any time -- the sprite will then change from 16 X 16 pixels to 32 X 32.

Press the Space Bar.

Now to change the sprite. This is called **sprite editing.** Select option 7 to Edit Sprite On Grid.

You will see a blinking cursor. Move it around with Apple's editing keys: I, J, K, and M. Use the ESC key to see your command options.

Plot on a grid square with the P key and erase a filled-in grid square with the E key.

Notice how your changes also affect the actual-size sprite.

Press Q to quit once your editing is completed.

Choose to save current sprite in memory. Specify 63 as the sprite number. A "calculating!" notice will appear on the screen. Your new sprite is being saved in Apple memory at the end of your sprite table. If you were to use 4 to Save Sprite Table To Disk, this new sprite would be a permanent sprite 63 in your ALIENS file. Let's **not** do that now.

Note: Unless you save the sprite in memory, your work will not be saved to disk.

Select option 8 to Rotate Sprite On Grid.

Choose (3) 180 degrees.

Read the procedure reminder -- then hit Space Bar. The alien should now appear upside down. It can be saved in this position too. You may to try other rotations, or you may exercise options 9 or A if you like.

Option A will give you a mirror image of your sprite. Option 9 will inverse your sprite (replace black squares with white and white squares with black). Press ESC to Quit.

From the main menu choose (3) Sprite Making/Editing & Scene Examination and in the submenu choose 5 for Applesoft Scene Inspector.

The purpose of this program is to allow you to inspect any PAINT MASTER SCENE UTILITY* scenes you may have created. There's one on this disk called SCENE6502. When you're asked for the "combination hplot scene and color fill file's name" specify SCENE6502 and examine the scene. Hit Space Bar and then press N for No when you're asked if you'd like to see another one.

The scene was a sample of an Apple graphics background scene on the screen at the same time as VDP (video display processor) patterns, sprites and text. This is called video overlaying, and multiplies the potentials of your Apple computer tremendously.

* an Avant-Garde scene-creation utility available alone, or as part of StarSprite II.

8. Maze Creation

From the main disk menu of side A choose (4) Making/Editing Files Used By Games

In the submenu choose (1) Maze Creation Using Game Paddles.

In the program choose (2) Load & See Old Maze.

Specify the maze name table NAMEM -- the final M stands for maze game. There's no need for disk switching.

You'll see the pattern scene (VDP being imitated by Applesoft hplots is the method used here for display purposes) used in the StarSprite Maze game, including an upper left corner gap used for scorekeeping.

Press the Space Bar and choose (1) Create Maze.

Answer no by use of the letter N when asked about putting SCORE:00000 onto your maze.

Answer Y for yes when asked about "POKEing frame characters in".

Read command page. Press any key to begin.

Notice the blinking dot -- that's your **cursor.** The coordinates are displayed at the bottom of the screen.

If you would rather be seeing a full screen with no text, press the Space Bar.

Now move the cursor with the paddles.

Press button 0 when you have it at your first maze line's starting place.

Now move the cursor again -- but this time to your line's endpoint -- (the beginning and ending points must not be the same) and press button 1.

Now move the cursor to your next maze line's starting point (this may be the last line's endpoint).

All lines must be at least as long as the distance between two separate dots.

Press button 0, move the cursor, and press button 1 again.

Lines must be either horizontal or vertical -- diagonals are not allowed.

If you type S you will be able to save your maze (if you change your mind, you may terminate the save with CTRL-C). But for this exercise, use ESC to exit maze creation.

A full tutorial on maze creation and using the maze in the Star-Sprite Maze game is contained in later in this manual.

9. Character Typing

From the main disk menu, side B, choose (5) Typing Programs.

Choose (1) Type Using StarSprite's Text mode.

In the program study the commands. If you need to see the commands again while typing, use CTRL-A for aid.

Use CTRL Q to quit and return to menu.

Use CTRL X to erase and restart.

The <Return> key will function as a carriage return. The left arrow will act as backspace. If you use CTRL B (background color) and CTRL T (text color) you can change the text and background colors.

The CTRL B or T is typed first, then 0-9 or A-F. Consider the A-F to be colors 10-15.

From the main disk menu, side B, choose (5) typing programs.

Select (2) Type Using StarSprite's Graphics Mode in the submenu.

In the program study the commands. They're the same as those of the text mode typing program. Try out the program the same way you did above. The only difference is that you're in graphics mode, not text mode, and your characters fit 32 to a line, not 40.

The character speed is 5 characters per second but using Ampersprite you will be able to attain far greater speeds.

The same pattern table character set is used in the graphics mode typing program as was used in the text mode typing program.

10. Ampersprite Tutorials

10.1. Graphics Tutorial Using Ampersprite

From the main disk menu choose option 6 for Tutorial Ampersprite Programs.

In the submenu choose (1) Ampersprite Tutorial For Graphics.

In the program, follow all examples, reading the text and watching Ampersprite command results. At the end, hit CTRL Reset and examine program lines to your heart's content.

10.2. Sound Effects Tutorial Using Ampersprite

From the main disk menu choose option 6 for Tutorial Ampersprite Programs.

In the submenu choose (2) Sound Effects Tutorial Using Ampersprite.

In the Program, read the two introductory text pages and when you see the (0-19) prompt, press <Return> for a tutorial. Once you're returned to the (0-19) prompt, try out the various sound effects. If you would like to hit CTRL Reset and examine the program in detail, feel free to do so.

10.3. Ampersprite in an Educational Application

From the main disk menu choose 6 for Tutorial Ampersprite Programs.

In the submenu choose option 3 for an example of the use of Ampersprite in an educational application.

In the program watch the simple, clean animation and screen full of sprites demonstrate a simple use of Ampersprite to support good educational graphics. Read the comments at the end and the tutorial on Ampersprite use. Then either hit CTRL Reset and list various program lines, or return to the main disk menu.

11. Ampersprite - Introduction

11.1. Sprites and Sprite Tables

A **sprite** is a programmable object. Sizes vary from 8 X 8 to 32 X 32 hi-res units. The factors involved are size and magnification. Study this table:

PIXEL SIZE	8X8 SPRITES	SIZE	MAGNIFICATION	TOTAL SPRITE SIZE
1X1	1	SIZE 0	MAGNIFICATION 0	8X8
1X1	4	SIZE 1	MAGNIFICATION 0	16X16
2X2	1	SIZE 0	MAGNIFICATION 1	16X16
2X2	4	SIZE 1	MAGNIFICATION 1	32X32

When magnification 0 is used, each sprite bit is mapped onto 1 X 1 pixels on the screen. When magnification 1 is used, each sprite bit is mapped onto 2 X 2 screen pixels.

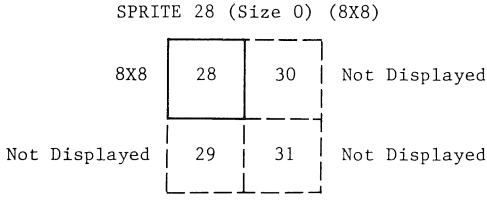
When size 0 is used, 8 consecutive bytes are used from the sprite table to define the sprite shape on the screen, starting at the sprite number given. When size 1 is used, 32 consecutive bytes are used from the sprite table to define the sprite shape on the screen. This creates a 16 X 16 sprite at magnification 0 and a 32 X 32 sprite at magnification 1.

When you create a sprite table you're allowed up to 64 sprites (0-63) before it's all filled up. Each of these sprites is a 16 X 16 sprite requiring 32 bytes. Each sprite of the 16 X 16 type is a composite of four 8 X 8 sprites. Here's the way a 16 X 16 sprite gets screen mapped:

SPRITE 28 (Size 1) (16X16)

8X8	28	30	8X8
8X8	29	31	8X8

The above is what size 1 is all about: putting together four 8 X 8 sprites to form a 16 X 16 sprite. All size 1 sprites take up four numbers in a sprite table. If the above sprite 28 was used but you were in size 0, only the solid portion of the screen map diagram below would be displayed on the screen:



Suppose your sprite object (16 X 16) was a ball. Only one quarter of a ball would show up if you used size 0 when displaying your 16 X 16 sprite on the screen. Using magnification 1 would simply give a 16 X 16 quarter of a ball instead of an 8 X 8 quarter of a ball. So every time you use 16 X 16 sprites be sure size 1 is in effect. In Ampersprite here's how to deal with size and magnification:

```
&RZ0   (gives size 0)
&RZ1   (gives size 1)
&RM0   (gives magnification 0)
&RM1   (gives magnification 1)
```

The R stands for VDP REGISTER. What this boils down to is a signal bit that's either on or off: hence the 1 or 0.

The reason people who know graphics want to use sprites rather than vector shapes or block shapes is that they're much faster, smoother, and easier to use.

You would need to have sets of seven preshifted shapes with conventional Apple block shapes, but with sprites, you will need only one sprite in memory.

In order to move a sprite in Applesoft, you might use code that looked like:

```
10   FOR Q=0 TO 224: &AX3,Q: NEXT
```

The Ampersprite command &AX simply means change the X coordinate on a sprite.

The 3 in line 10 signals the SuperSprite to change the horizontal location of the sprite on video plane 3.

The Q gives the current X coordinate value.

When locating a sprite, the upper left corner will appear at the specified coordinates. This means a sprite at X = 0 will be on the screen, but a 32 X 32 sprite with an X coordinate over 224 will be partially off the screen to the right.

When moving a sprite into the display you're allowed to use -31 to 0 to effect "bleeding in" of the sprite from the top edge of the backdrop.

Values of 207-191 allow bleed-in from the bottom of the screen.

Note: The value 208 given as a Y coordinate is a signal to the VDP to cease sprite processing. This allows you to blank out a whole section of sprites currently being used.)

For horizontal bleed-in from the right edge of the screen, use values from 255 downwards.

For left bleed-in a special bit in the color byte of the sprite attribute table is set.

The forth byte of any sprite attribute table entry is the color byte. The first four bits of this byte get the sprite color number (0-15). But the most significant bit gets turned on when you want to bleed-in a sprite from the left edge of the screen.

This means adding the number 128 to your color number. So, if your color was 6 (red), you would specify 134 as your color number until your sprite was fully within the normal screen boundaries -- then you'd change your color number to 6.

The way this EC bit (early clock) works is to shift your sprite's horizontal screen position left 32 pixels. So when you give a 0 X coordinate with color 134 your sprite will be off the screen. Once you get to a 32 X coordinate you must change the color to 6 and the X coordinate back to zero to be fully within the backdrop.

A sprite attribute table is a table of attributes up to 128 bytes long -- enough to give four attribute values for each of 32 sprites on 32 different sprite planes. Each sprite gets four attributes in the following format: Y (vertical), X (horizontal), sprite number, color. Here's a sample 3-sprite attribute table:

VIDEO PLANE #0				VIDEO PLANE #1				VIDEO PLANE #2			
Y	X	Sprite Number	Color	Y	X	Sprite Number	Color	Y	X	Sprite Number	Color
43	29	88	8	9	81	36	3	0	0	0	0

For this example, we are specifying 16 X 16 sprites (size 1, magnification 0 or 1), and therefore expect **sprite numbers** to include the three following 8 X 8 sprite numbers as well and thereby be divisible by 4.

When using attribute tables, you needn't do any more than use a few Ampersprite commands to get the table in memory, if it is short. If it's long, use the &L (load) command.

Sprite pattern tables (or sprite tables) are composed of up to 2048 bytes, which allows for 256 8 X 8 sprites or 64 16 X 16 sprites. Shorter tables are allowed (you need only have as many sprites in the table as you intend to use).

You may have no more than four sprites on one horizontal scan line. If you use more, the sprite of lowest priority will be obscured on that line.

12. Patterns and Pattern Tables

The VDP pattern plane (often called the background plane) has a lower display priority than sprites, but higher than the backdrop or external video (or Apple graphics or text).

Note: Priority refers to which plane gets precedence on the display screen--lower numbers have higher priority.

Three tables are used to generate pattern planes on-screen: the name table, the color table, and the pattern generator table.

There are four VDP modes that govern what is displayed on the pattern plane:

- Graphics I
- Graphics II
- Text
- Multicolor mode (lo-res graphics)

The multicolor mode is similar to Apple lo-res except it is less convenient to use.

See the section on the multicolor mode for more information on it.

To get into multicolor mode from Ampersprite, use the command &RU.

Multicolor mode uses no color table. It uses a pattern table that defines tile colors, and a name table to point to these color-defining pattern table entries. (To see what it looks like LOAD Sprite Painting and change &RG2 in line 90 to &RU. Then paint a Sprite Painting (main menu option 2), and once the screen is fairly full quit and examine the screen.)

How do you tell the location and color of a pattern?

```
Pattern table = WHAT SHAPE (which dots on the screen are on)
Color table   = WHAT COLOR (what color the dots are)
Name table    = WHERE ON SCREEN IS WHAT PATTERN
```

A pattern is an 8 X 8 dot tile. It represents eight bytes of data, mapped one below the next, sequentially.

Pattern tables are bit-mapped when they are displayed.

You would expect a 255 ($FF) byte to be a dash and a 0 byte to be nothing, thereby allowing lower priority video (backdrop or Apple graphics) to show through. This is not the case. Bits being on or off signify which of two possible colors will be displayed on the screen.

In Graphics I mode, the color table corresponds to the pattern table as follows: the color table has 32 entries, each one byte long. Each entry defines two colors. You may have seen machine language hexadecimal bytes before. For example, $F6. Each digit, in the color table, defines a color. So each of the 32 entries defines two colors, so in the above example colors $F (15) and 6 are stipulated. The on bits of the corresponding pattern table bytes would be color 15 and the off bits would be color 6.

The first entry in a Graphics I mode color table defines the colors for patterns 0-7. The next entry is for patterns 8-15, etc. This goes up to the last entry in the color table (the 32nd) which defines colors for patterns 248-255. This takes care of all pattern colors, since there can only be 256 patterns in Graphics I mode.

The screen-mapping is done via the name table. A name table has 768 entries, called pattern positions. Each of these 768 entries must contain a number from 0 to 255, representing which of the 8-byte patterns to display.

In other words, a pattern table is like a library of little shapes on 8 X 8 dot tiles. When you want to display one of these shape-tiles on the screen, you need to NAME the one you want.

Assume position number 5 has a 3 in it. Number 5 is the 6th position since we must count number 0 as being first.

Number 0 is at the upper left hand corner of the screen. Position 5 is five tiles (8 X 8) to the right of that. If the name table has a 3 in position 5 that means that the fourth (since pattern 0 is first) set of 8 pattern bytes in the pattern table will define what is displayed at position 5 on the screen.

The color of the pattern at position 5 is determined by the two digits in the first color table entry, since the color table entries each cover eight consecutive patterns.

Again, on bits correspond to the color represented by the first digit of the color table entry and off bits correspond to the color represented by the second digit.

Let's look at a sample diagram and sample tables that will help clarify all this:

PATTERN TABLE (IN HEXADECIMAL)

PATTERN NUMBER								
(0)	70	88	98	A8	C8	88	70	00
(1)	20	60	20	20	20	20	70	00
(2)	70	88	08	30	40	80	F8	00
(3)	F8	08	10	30	08	88	70	00
(4)	10	30	50	90	F8	10	10	00
(5)	F8	80	F0	08	08	88	70	00

Etc. up to pattern number 255

COLOR TABLE (IN HEXADECIMAL)

(48)	98	18	16	04	89	22	16
8B	3A	FF	01	00	06	66	78
94	B8	C1	D8	64	62	86	04
99	88	42	01	11	19	47	AC

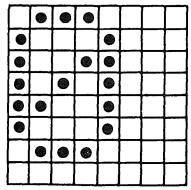
That's the complete table

NAME TABLE (IN HEXADECIMAL)

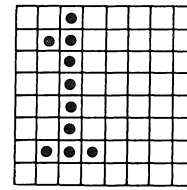
88	B1	64	11	(00)	(03)	(02)	DD
F1	(02)	7E	E2	62	19	92	87

Etc., up to entry #767

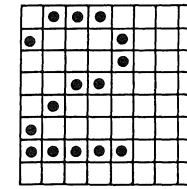
PATTERN #0



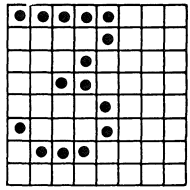
PATTERN #1



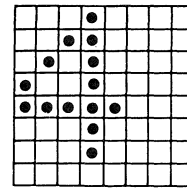
PATTERN #2



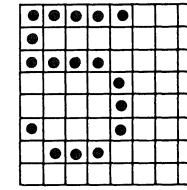
PATTERN #3



PATTERN #4



PATTERN#5



(If you would like to understand how bit-mapped pattern bytes are determined, study the values, in decimal and hexadecimal, for each of the eight bit positions below:)

$80	$40	$20	$10	$08	$04	$02	$01
128	64	32	16	8	4	2	1

(The first byte of pattern 0 gets a value determined by adding up ON bit positions. $40 and $20 and $10 add up to $70. Or 64 and 32 and 16 add up to 112, which also translates to $70.)

You have seen that the first eight patterns use only the first entry in the color table in Graphics I mode. That means that in the above example not only the six patterns given but the next two as well would use $48 as their color-determining data. So on bits would be $4 or dark blue, and off bits would be $8 or medium red.

Can you predict what the upper left corner of the display screen would look like using the above tables?

(?) (?) (?) (?) 0 3 2 (?)

(?) 2 (?) (?) (?) (?) (?) (?)

The above diagram is all the pattern predictions that could be made. Remember that the 0, the 3, and both 2's will have blue text on red background.

Color predictions can easily be made for any of the above question-marked areas of the screen (meaning the pattern shape is a yet unknown, since the pattern table given only goes up to 5).

We know not only that patterns 0-7 will use $48 for color data. We also know that patterns 8-15 will use $98 and 16-23 will use $18, etc. So the fourth name table entry, $11 (decimal 17), can be predicted to have black ($1) text on a medium red ($8) background. The rest are just as easy to predict; as I've said -- the color table is complete as shown.

Graphics II mode is a bit different. The pattern table is 6144 ($1800) bytes long and so is the color table. The two of these alone take up $3000 bytes -- 3/4 of the VDP RAM on the SuperSprite. The card has 16K, a 6144 byte table is 6K.

Note: Sprites and backgrounds are available in either Graphics mode (and multicolor mode as well).

If you create pattern tables that have 5 X 7 text characters, these tables are usable in both graphics and text mode of your VDP.

When using text mode there is room on the screen for 40 X 24 characters and characters are on 8 X 6 tiles, not 8 X 8 tiles, so you must make sure that the least significant three bits of your pattern bytes are zero.

Using text in graphics mode uses 8 X 8 tiles (again using 7 X 5 characters, as in text, mode) but you have the option of creating 7 X 6 or 7 X 7 characters, if you desire -- but they'll be unusable in text mode, and dual use pattern tables are normally the best way to go.

The reason Graphics II mode is so memory intensive is that scenes and detailed graphics take more space to store. In a complex picture, the patterns in every tile position on the video display will differ from every other tile position pattern. This means that there will need to be 768 different patterns.

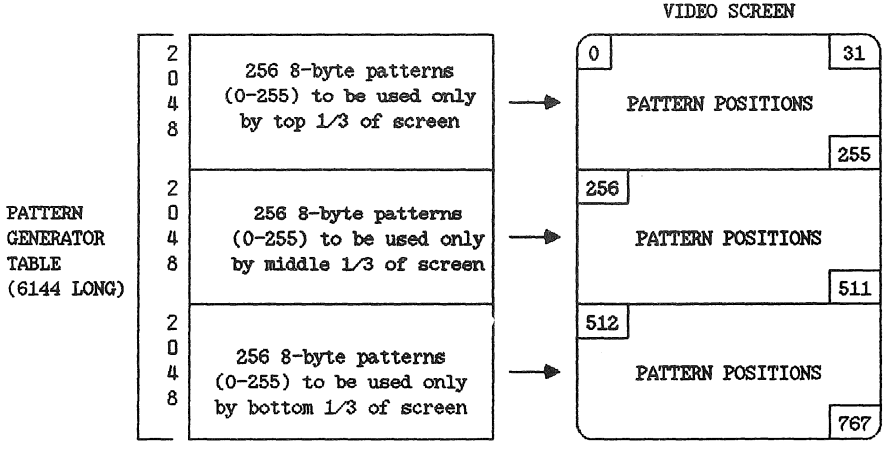
This is not possible in Graphics I mode because the data (representing pattern numbers) which goes into each pattern position in the name table must be 0-255.

Here is how things are set up in Graphics II mode:

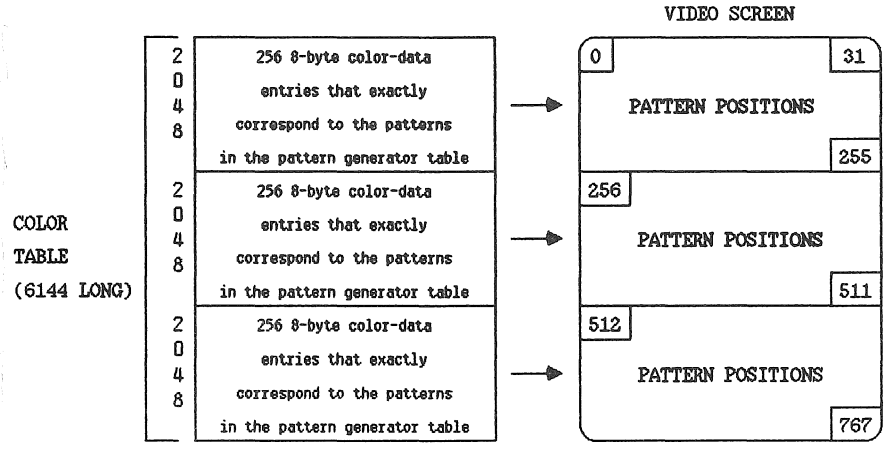
In Graphics II mode the name table is partitioned into three segments, each 256 bytes long (just like Graphics I).

The first segment (256 X 8 = 2048 long) is mapped to the top third of the screen only. The second segment is for the middle third of the screen and the third segment is for the bottom third of the screen.

These three pattern tables are adjacent in memory and therefore form one long (6144 long) pattern table, as follows:

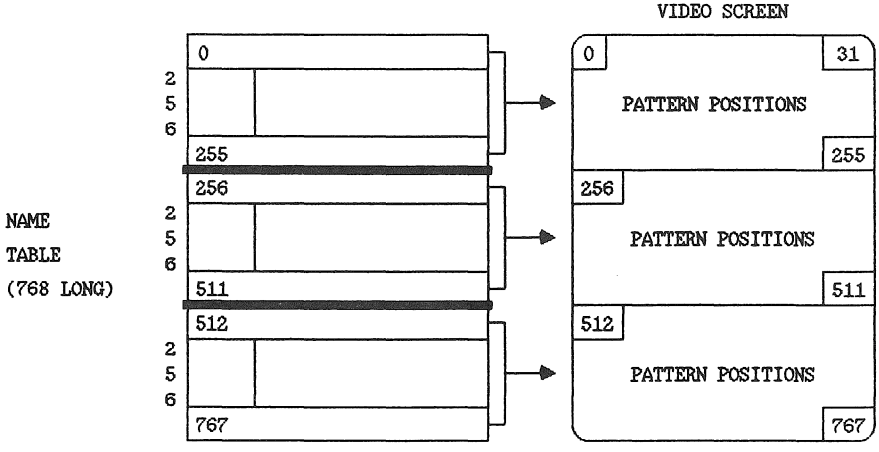


There is a corresponding color table, also 6144 long:



If you understand the above diagrams, you will see that the 5777th byte of the pattern table has its colors defined by the two digits of the data in the 5777th byte of the color table. If $48 is found in the color table in the 5777th byte and the 5777th pattern table byte is $81 (129), then it means that if that particular pattern byte is used on the screen, it will have a blue dot on each end and a red dash in the middle.

Note: Only if the name table points to a pattern is it ever displayed on the screen. Only if the name table has a "210" in its lower 1/3 will we know that the 5777th bytes of both color and pattern tables are being used. Examine a name table in Graphics II mode; it, like pattern and color tables, is divided into thirds:



In this example, we are examining the use of the 5777th byte of the pattern and color tables.

This byte is seen (along with the other seven bytes in that pattern) only if the name table points to it.

In order to name the pattern, we perform the following calculation:

Pattern name = Byte in Pattern Table ÷ 8

So:

Pattern name = 5776 ÷ 8 = 722

The 722nd pattern would be in the bottom third of the pattern table so subtract 512 to get the actual pattern name of 210.

Notice that I used 5776 since the 5777th byte in the table is byte 5776 because byte 0 is the first one. To check, notice that if you start the top third of the pattern table with byte 0, the middle third with byte 2048 and the bottom third with byte 4096, then 4096 + (8 times 210 = 1680) = 5776.

So, in the actual name table, unless the pattern number 210 ($D2) is found in the third section or page of the table, then bytes 5776-5783 (the 5777th-5784th bytes in the table), which make up pattern 210, will not be displayed on the screen.

It is impossible for patterns from sections to be displayed on a different third of the screen.

In actual usage, if you want a blank screen just have pattern 0 be eight 0's in the first, second, and third sections of the pattern table, and then put 256 zeros in a row for each section of the name table -- 768 zeroes in all.

Notice that it took only 24 pattern bytes of data to define an entire screen full of video display.

If you want the middle section to be all solid instead of blank (black), you may have the eight bytes of the first (0) pattern of the middle section of the pattern table all be 255 ($FF).

There is another way, however. All the above speculations are based upon the assumption that you've put something like 241 ($F1) into the first eight color data bytes (0 patterns will therefore correspond with them) of each of the three sections of the color table. $F1 means white text for on bits and black background for off bits since $F = 15 = white and 1 = black.

In color bytes the rule is always that the first hex digit determines the color of the on bits and the second hex digit determines the color of the off bits.

The remainder (2040 bytes) of the bytes of both pattern and color tables can be anything at all and it will have no effect on the video display if the name table points only to pattern 0 in all pattern positions (all 256) in all three sections of the name table. **Only the color and pattern data that are being pointed to by the name table bytes have any influence on the display.**

The other way to get the top and bottom sections of the screen to appear black and the middle white is to leave the pattern data bytes of pattern 0 alone and instead concentrate on the color table data corresponding to pattern 0. Let's change the $F1 to $FF or $0F or $1F or anything else that ends in $F ($F = 15 = white) in the first eight bytes of the middle section of the color table (again, this corresponds to pattern 0).

If you'd prefer a red band have the last digit be 6 or 8 or 9; for green try 2 or 3 or 12 ($C).

Suppose I need a screen full of the digit 3. The hexadecimal digits for a pattern that looks like a "3" are: F8 08 10 30 08 88 70 00.

To display this pattern in each tile position on the screen, the following steps are necessary:

- Make this set of bytes pattern 0 in the first pattern of each section of the pattern table.
- Use $F1 for all corresponding color table data.
- Have all 768 bytes of the name table be $0.

Or use it as pattern 3 (the fourth pattern) and put the above data as bytes 24-31 in each of the three sections of the pattern table, and fill the name table with three and the color table with 241's ($F1).

The following example line of Applesoft Basic demonstrates the use of Ampersprite to accomplish the goals discussed above.

```
10   POKE 7, 241: POKE 8,0: POKE 9,0: CALL 852: REM THIS ASSUMES
     THAT COLOR TABLE IS AT $2000 AND IS $1800 LONG
```

The above line (choose your own line number, of course) will fill all 6144 bytes of the color table with 241. If you want some other type of data to end up in the color table, merely substitute the new color data number for 241 and then POKE that into 7 above, where I've POKEd 241.

If you only wish to deal with byte 5776 of the pattern table, use the following Ampersprite command:

```
10   &P 5776,0
```

The P stands for pattern, the 5776 gives the table's byte number (or offset past starting address of table) and the 0 is the data to insert. To take care of the entire pattern which begins at byte 5776 you could do this:

```
10   FOR A = 5776 TO 5783: &P A,0: NEXT
```

You will normally be using pattern numbers, not byte numbers, so the following scheme makes more sense:

```
10   FOR A = 4096 + (210*8) TO 4096 + (210 + 8) + 7: &PA, 0: NEXT
```

Or:

```
10   B = 4096 + (210*8): FOR A = B TO B + 7: &PA,0: NEXT
```

Since the color table has a one to one correspondence to the pattern table in Graphics II mode, it can be handled in the same way:

```
10   B = 4096 + (210*8): FOR A = B TO B + 7: & CA,8: NEXT
```

In the above example, on bits get 0 for data and off bits get 8 (red). The C stands for color. Below we'll update the pattern table with new pattern data for pattern 210 of the third section, and we'll use 8 for the color data -- but we'll put these two routines together:

```
10   B = 4096 + (210*8): FOR A = B TO B + 7: &PA,0: &CA,8: NEXT
```

The above line of Basic will only change the screen in places where the position you are at is pointed to by a name table entry of 210, and it must be in the third section. If you want all three of the pattern table and color table sections to have their pattern 210 updated, use the following algorithm:

```
10   FOR B = 210*8 TO 4096 + (210*8) STEP 2048
20   FOR A = B TO B + 7: &PA,0: &CA,8: NEXT : NEXT
```

Often the pattern and color table data are static, while the name table gets updated to change the screen display.

Suppose the 18th, 19th, and 20th patterns in the pattern table (17-19) are letters A, B, and C. And suppose you need the word "CAB" to be displayed on the screen at name table positions 260-262. The commands are:

```
10   &N 260,19 : &N 261,17 : &N 262,18
```

Here is a routine for getting words, in string form, onto the screen:

```
500  A$ = "STARSPRITE FOREVER!": PSN=262: GOSUB 510: GOTO 520:REM
     POSITIONS 262-281 ARE USED HERE
510  L = LEN (A$): FOR A = 1 TO LEN: C$=MID$ (A$,A,1): CH = ASC
     (C$) -32: &N PSN + A-1, CH: NEXT : RETURN
515  REM PSN = SCREEN POSITION TO PRINT ON AND CH = CHARACTER TO
     PRINT (GIVEN BY PATTERN NUMBER)
```

In the above example, a pattern table CHAR is used. You'll find it on side A of the main program disk.

For an actual example of the usage of these commands, load program called Ampersprite Basic. Examine line 0 to see how this file is loaded and lines 2000-2199 to see how pattern tables and name tables can be loaded into VDP RAM.

In CHAR, ASCII numbers 32-127 have been given pattern table numbers 0-95 for convenience, which explains the minus 32 in line 510.

To put a colored background behind each character of "StarSprite Forever!":

```
530  FOR A = 33 TO 58: Z = 2048 + A *8: FOR B = 0 TO 7: &C Z+B,6:
     NEXT : NEXT : FOR B = 2048 TO 2048 + 1*8 + 7: &C B,6 : NEXT
```

The pattern numbers of the 26 letters of the alphabet, in this Graphics II mode example, are 33 to 58. Space and "!" are numbers 0 and 1.

Notice that we're past name table position 255 but not as high as position 512, so the tiles we are using are located in the middle section of the screen and tables.

Also notice that since each segment of the table is 2048 bytes long, adding 2048 to the starting address of the color or pattern tables locates the position of the start of pattern 0 of the middle section.

Also notice that multiplying any pattern number by 8 gives us the number of bytes of offset (from the start of that section of the pattern or color table) to address the table bytes for a given pattern (or the colors that correspond to it) in the table.

In line 530 the 2048 gives us the section offset and A*8 gives us the pattern offset to locate the correct pattern.

Notice that the variable "A" will always represent the pattern number we are now dealing with.

The FOR-NEXT loop indexed by "B" refers to the eight bytes each pattern contains.
&C Z+B,6 puts the color red (6) into all eight bytes of each alphabetic character pattern (#33-58).

The second FOR-NEXT loop indexed by "B" puts red (6) into patterns 0 and 1, the space and "!" patterns.

All we've really done is change color table data behind all the letters, not only of "StarSprite Forever!" but also of the alphabet, space, and "!". If we had already inversed (background added) letters available as patterns in our pattern table, all we would have had to do is "&N" those into place with code similar to line 510, with the minus 32 changed to whatever is appropriate to have ASCII codes lead us to correctly corresponding pattern table numbers.

The TMS9918A VDP also has a Text mode. The Text mode also accesses pattern tables, but only uses the most significant 6 bits of every byte, which allows a 40 column display in text mode, as compared to 32 column display in either graphics mode.

Turn back a few pages and study the patterns that create the numerals 0 through 5. Notice how the characters are on an 8 X 8 grid but are 7 X 5 in size. In graphics mode this allows three spaces between characters horizontally and one space vertically.

In text mode, because only six horizontal bits are used, there is only one bit space between characters in all directions.

To enter text mode, use &RT after Ampersprite is initialized.

To enter Graphics I mode, use &RG1.

To enter Graphics II mode, use &RG2.

To enter multicolor mode, use &RU.

Remember to initialize the Ampersprite. See lines 30 and 80-154 in the Ampersprite Basic program (side A) for an example.

The "&R" commands set the various VDP registers. R means register.

Text mode uses only 3/4 of every pattern for display.

Another difference is a 960 (0-959) position name table (40*24=960). See programs TYPE.TX and TYPE.GR to see how graphics and text characters differ. Ampersprite wasn't used in these programs, to allow you to learn Basic interfacing with the SuperSprite without Ampersprite.

No sprites are allowed in text mode, but you may alter text color:

```
& RX 15,6
```

In the above example X stands for text, the 15 gives text mode text color, the 6 gives text mode background color.

Note: The second parameter of the & RX command specifies the backdrop color for all four modes.

The screen is **not** divided into three sections in text mode so this mode is similar to Graphics I mode in pattern table requirements (except for the 6 of 8 bits per byte usage).

In text mode, update the name table with "&N" commands.

13. Colors and Color Tables

The numbers governing the color generated by the SuperSprite are:

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 = Light blue	13 = Magenta
6 = Dark red	14 = Grey
7 = Cyan	15 = White

The use of any color except transparent (0) in patterns will prevent Apple video from showing through, so make sure your backdrop, controlled by "&RX" command, specifies transparent backdrop, for example: "&RX15,0".

If you are not mixing Apple and VDP graphics, other backdrop colors are fine.

On sprite video planes, the entire plane is transparent except for the part of the plane containing the sprite. A sprite can be covered up or hidden by only one method: having a higher-priority sprite cross its path (the lower the video plane number the higher the priority).

A normal pattern table based scene (not those created in the Sprite Painting program) is transparent everywhere one of the color table's bytes that corresponds to pattern table bytes has a 0 in one of its hex digits; e.g. $03 or $E0, so long as both on and off bits appear in the pattern and the name table points to the particular patterns in question.

A pattern not pointed to is not displayed.

Color tables, in the multicolor or text modes, do not exist. The "&RX" command takes care of text color and text background color in text mode. The "pattern table" takes care of color in the multicolor mode -- this table tells the VDP not what various patterns look like but what color tiles go where.

There are two types of color tables: Graphics I and Graphics II. (Sprite colors use no tables -- they are determined by the fourth byte in sprite attribute so sprite colors are for sprites while color tables are for patterns only.

Graphics I mode color tables contain 32 bytes. The first byte corresponds to patterns 0-7. The second to patterns 8-15. The last (32nd) to patterns 248-255.

Each color byte is a number between 0-255 (inclusive). The hexadecimal notation of this data is easier for illustrative purposes. Each hexadecimal digit may represent a color, and all hex numbers (8-bit) use two digits (when you see $7, $07 is understood).

The first hex digit of a color byte corresponds to the on bits of the pattern and the second hex digit to the off bits of the pattern. A number like $48 means that the on bits are color 4 and the off bits are color 8.

The limitations of Graphics I mode are obvious when you try to imagine creating a complex multicolor scene with it -- you are generally better off using this mode for text that is being used in the graphics mode (32 X 24 characters/screen).

The Graphics II mode uses a different approach. In this mode there are 6144 pattern table bytes and 6144 corresponding color table bytes. Each byte in the pattern table has a corresponding byte in the color table.

We have looked at &RX#,#, which controls text and backdrop colors.

In order to control the colors that pertain to the background patterns and sprites, several new commands are required.

First, we must specify where color table is located in VDP RAM; The command for this is: &RC#. This command will be discussed in the next chapter.

The &AC #,# command tells sprites what color to be.

The &C #,# allows you to place color data directly into the color table from Applesoft.

```
10   B = 4096 + (210*8): FOR A = B TO B + 7: &C A,8: NEXT
```

The above line changes the color data to $08 for pattern 210 in the bottom section of the pattern table.

```
10   POKE 7,241 : POKE 8,0: POKE 9,0: CALL 852: REM THIS ASSUMES
     COLOR TABLE IS AT $2000 AND IS $1800 LONG
```

The above routine causes the color table (all 6144 bytes of it) to be filled with the number POKEd into location 7, in this case, 241 ($F1).

The Sprite Painting program makes use of the pattern and color tables in a different manner than we have been discussing.

Both are the inverse of what you would expect after dealing with normal pattern and color tables.

The merge routine takes the sprites (which you move around the screen and paint with) and merges them into the current pattern table and color table. The screen this painting program starts out with has all zeroes in both the color and pattern tables.

Normally a one-dot addition to a black (0) byte would cause the new value to be 1, 2, 4, 8, 16, 32, 64, or 128.

Instead a one dot addition causes the new pattern table byte to be 254, 253, 251, 247, 239, 223, 191, or 127, (the inverse of the screen's normal bytes -- dashes with notches in them instead of simple dots). The same inverse situation happens to the color table as well.

Normally, when a dot is added to the pattern table, the corresponding color byte would be $F0 (240) in the color table, which means 15 ($F) is for on bits (15 is white) and 0 is for off bits (0 is transparent).

However, when the merge program is working you get the inverse of this: $0F is placed in the color table rather than $F0. This means that the on-bits will be transparent and the off bits will be white.

Reversing the figure-ground format on either the color or the pattern table byte would have caused an inverse result on the screen. But reversing the format of both results in the sprites being copied accurately into both the color and pattern tables.

14. VDP Registers

The TMS9918A video display processor has eight write-only registers and a status register.

Technical Note: You can write to any of the registers with the "&RR#,#" command. The first number is the register number to update, and the second number is the data to write. Do not use this command to help load registers initially -- there are specific commands for that. Use it for registers 2-7 later, if necessary, but check lines 30-65 in the Ampersprite Basic program on the main program disk (Side A), or see below, for the ramifications of register manipulation.

Since Ampersprite keeps track or the contents of the various registers, you will have to want to update a the Ampersprite locations corresponding to the register changed so Ampersprite commands do not get confused about where your tables are.

The "&RR" command should be avoided unless you are only using parts (such as the sprite attribute section of the language) of the Ampersprite language or are handling all the above updating necessary.

There are 14 different Ampersprite commands for dealing with the VDP registers. The order in which the various commands should appear (near the beginning of the program) is:

1. CALL 912
2. &I
3. &RX (text color, backdrop color)
4. &RG (1 or 2)
5. &RE (0 or 1)
6. &RD (0 or 1)
7. &RZ (0 or 1)
8. &RZ (0 or 1)
9. &RS (0 or 1)
10. &RA (address or sprite attribute table)
11. &RP (address of pattern table)
12. &RC (address of color table)
13. &RN (address of name table)
14. CALL 831 (clears VDP RAM)
15. &L tables into VDP RAM

A **register** is a place where things are stored. The VDP registers control the operation of the VDP and allocate where information is stored in VDP RAM. The reason you want to write to VDP registers is so that the VDP knows where sprite, name, pattern, color, and attribute tables are stored in VDP RAM, and so the various colors, sizes, modes, magnifications can be specified to the VDP and stored until new video characteristics are desired.

At the level of this utility, you will have no reason to read the status register (which needs to be done in machine language). But you may need to read VDP RAM data, to see how something has changed or to see if a table has loaded correctly.

```
10    POKE 31,8: POKE 30,0: POKE 249,0: POKE 251,9: POKE 252,0:
      CALL 804
```

The above routine is the reverse of the "&L" command. It has five parameters that are respectively Apple memory high byte, Apple memory low byte, counter, VDP RAM high byte, VDP RAM low byte.

With "&L" you are copying bytes from Apple RAM into VDP RAM. With the above CALL 804 routine, which is a VDP READ routine, you are copying bytes from VDP RAM into Apple RAM.

The above example, reads from $900-$9FF in VDP RAM and copies it into $800-$8FF of Apple memory. The above routine copies one page ($100) of memory. If 0 is placed in the counter is interpreted as $100. Any other value (1-255) put into the counter will be taken at face value.

Here are the functions of write-only registers 0-7:

Register Number

0. Enable/disable Graphics II mode

1. Specify RAM type, enable/disable display, interrupts enabled or disabled, text mode enabled or disabled, multicolor mode enabled or disabled, size 0 or 1 specified, magnification 0 or 1 used.

2. Name table base address stored

3. Color table base address stored

4. Pattern table base address stored

5. Sprite attribute table base address stored

6. Sprite table base address stored

7. Text color stored; text background and backdrop color stored

Here is an example of how a program might start out:

```
10    D$ = CHR$(4)
20    PRINT D$; "BLOAD K2"
30    PRINT D$; "BLOAD AMPERSPRITE"
40    PRINT D$; "BLOAD I&I"
50    CALL 912: REM INSTALL AMPERSAND HOOK
60    &I : REM INITIALIZE BOARD AND AMPERSPRITE
70    &RX 15,1: REM WHITE TEXT AND BLACK BACKGROUND
80    &RG2: &RE0 : &RD1 : &RZ1 : &RM0 : REM CHOOSE MODE, ETC.
```

```
90   &RS6144: REM SPRITE TABLE AT $1800
100  &RA 15104: REM SPRITE ATTRIBUTE TABLE AT $3B00
110  &RP0 :REM PATTERN TABLE AT $0000
120  &RC 8192 :REM COLOR TABLE AT $2000
130  &RN 14336 :REM NAME TABLE AT $3800
140  CALL 831 :REM ERASE VDP RAM BEFORE LOADING
150  REM NOW ACTUALLY LOAD ALL TABLES FROM APPLE MEMORY TO VDP
     RAM
160  REM START THE ACTION (SPRITE MOVES, ETC.)
```

&I zeros all VDP table base addresses stored in Apple RAM, displays VDP graphics only, and initializes the SuperSprite board.

Remember that CALL 912 must happen before the "&I" Ampersprite command.

The "&RX 15,1" command makes all text white (15), and all text background black (1). The value specified for the text background defines the backdrop as well.

Follow along on your command card as we go through these register commands.

Note: Do not use &RX, &RG, &RE, &RD, &RZ or &RM again in your program (without restarting with &I) if you have altered locations $DB-$DF, $D4, $D5, $EB, $EC, $19, $ED, $EE, or $EF.

Do not use &C or &N or &P again in your program (without restarting with &I) if you have altered locations $DB, $DE, or $DF.

Do not use any sprite attribute (&A) command if you have altered locations $DD or $DC.

The "&RG1" command simply chooses Graphics I mode and "&RG2" chooses Graphics II mode.

The "&RE0" command disables external video, (a signal from another VDP). This command is a set-up command required with the SuperSprite. Note: there is no provision on SuperSprite for external video feed.

The "&RD1" command enables display.

The "&RD0" command blanks VDP display.

In the demonstration programs, &RD0 is sometimes used in the subroutines at lines 9-10 to blank the VDP video and then filter Apple video through it. Then in 22009 &RD1 lets normal VDP video to be displayed. The point is to improve the way Apple video looks on color monitors.

The "&RZ1" command specifies size 1.

The "&RZ0" command specifies size 0.

Size 0 gives only eight bytes worth of sprite data per shape, while size 1 gives 32 bytes and is therefore much more useful for most purposes.

```
Size 1 + magnification 1 gives 32 X 32 sprite
Size 1 + magnification 0 gives 16 X 16 sprite
Size 0 + magnification 1 gives 16 X 16 sprite
Size 0 + magnification 0 gives  8 X  8 sprite
```

The "&RM1" command specifies magnification 1.

The "&RM0" command specifies magnification 0.

Magnification 0 specifies one-to-one relationship between sprite data bits and screen-displayed pixels.

Magnification 1 gives a 2 X 2 pixel of screen display for each bit of sprite data.

The size and magnification factors refer to **all** sprites on the screen. Sprite size and magnification may not be done on an individual sprite basis.

The &RS, &RA, &RP, &RC and &RN commands specify where in video RAM you wish to locate the sprite generator table, sprite attribute table, pattern generator table, color table, and name table.

There is $4000 (16K) of VDP RAM in which to store all your VDP-related data (see next chapter's memory map). You must specify to the VDP where you intend to store which data.

Now let's look at permissible addresses of tables:

In Graphics II mode the pattern table must start at $0000 or $2000 (0 and 8192 decimal). The color table also must start at $0000 or $2000, whichever was not used by the pattern table.

In Graphics I mode the pattern table must start at any address evenly divisible by $800 from $0 to $3800 (14336). In Graphics I mode the color table must start at any address evenly divisible by $40 from $0 to $3FC0 (16320).

In either mode the name table must start at any divisible-by-$400 number from $0 to $3C00 in steps of $400 (1024).

The sprite table is independent of the graphics mode selected, and must start at an address evenly divisible by $800 number from 0 to $3800 (14336) in steps of $800 (2048).

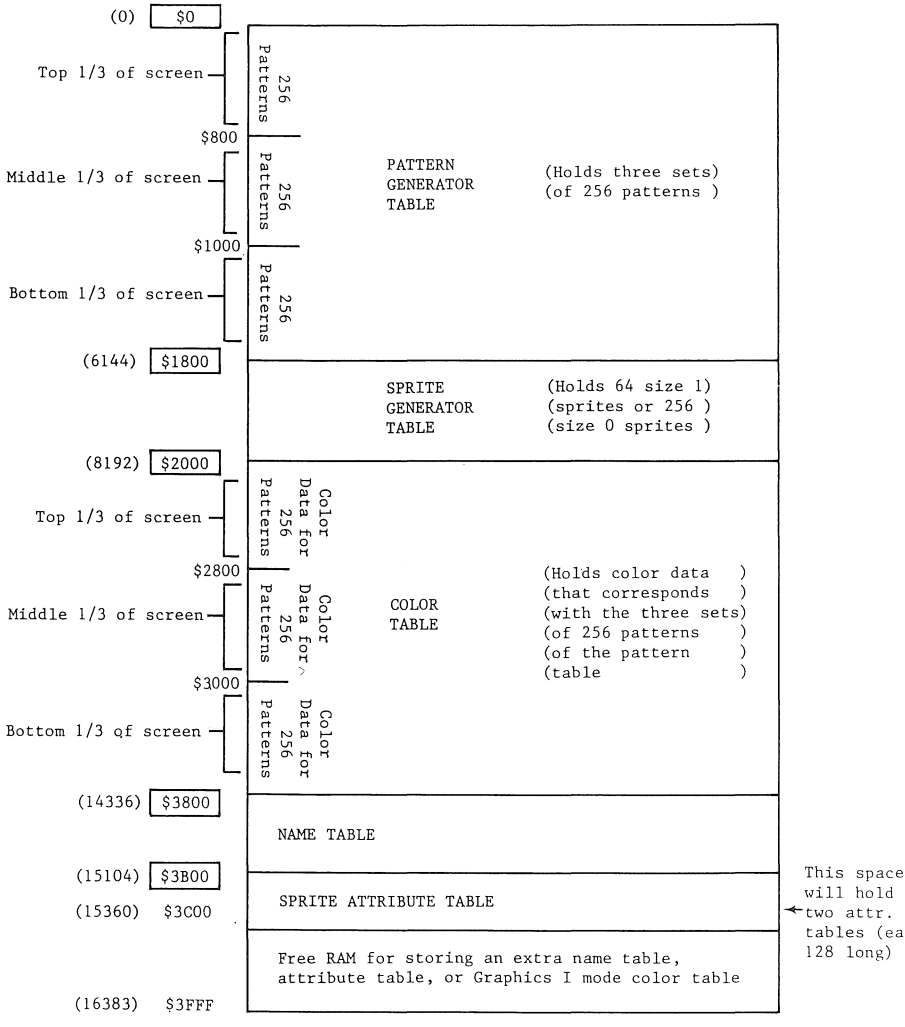
In either mode the sprite attribute table must start at any address evenly divisible by $80 from $0 to $3F80 (16256).

Overlapping is permitted but you must be careful not to overwrite areas controlling current screen displays. For example, you may have a sprite table at $1800 and a sprite attribute table at $1F80, so long as you have only 60 sprites (16 X 16) in your table rather than the maximum 64.

15. VDP RAM - a Memory Map

On the next page is the suggested allocation of VDP RAM for best overall efficiency in the widest variety of cases.

GRAPHICS II MODE VDP MEMORY MAP



(In text mode put name table at $3C00 so as not to overflow into $3B00, since text mode name tables are 960, not 768, long)

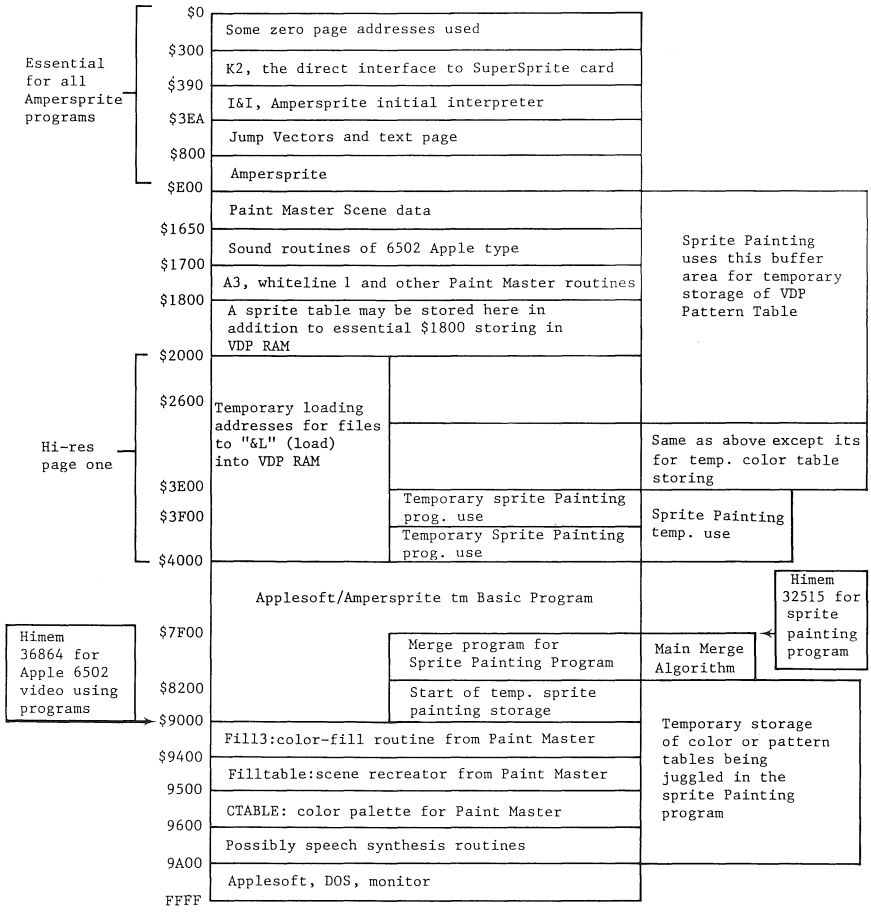
16. Ampersprite Memory Considerations

The following zero page (Apple RAM) locations are used by the Ampersprite language so they should not be used by your programs while Ampersprite is in effect:

$DB-$DF, $D4, $D5, $EB, $EC, $19, $ED, $EE, $EF, $8, $9, $7, $1E, $1F, $F9-$FF

Other addresses to avoid are $300-$3EA and $800-$DFF.

On the next page is a map of Apple memory usage for an average Ampersprite program:


Essential for all Ampersprite programs
$0
Some zero page addresses used
$300
K2, the direct interface to SuperSprite card
$390
I&I, Ampersprite initial interpreter
$3EA
Jump Vectors and text page
$800
Ampersprite
$E00
Paint Master Scene data
$1650
Sound routines of 6502 Apple type
$1700
A3, whiteline 1 and other Paint Master routines
$1800
A sprite table may be stored here in addition to essential $1800 storing in VDP RAM
Sprite Painting uses this buffer area for temporary storage of VDP Pattern Table
$2000
Hi-res page one
$2600
Temporary loading addresses for files to "&L" (load) into VDP RAM
Same as above except its for temp. color table storing
$3E00
Temporary sprite Painting prog. use
$3F00
Temporary Sprite Painting prog. use
Sprite Painting temp. use
$4000
Applesoft/Ampersprite tm Basic Program
Himem 32515 for sprite painting program
Himem 36864 for Apple 6502 video using programs
$7F00
Merge program for Sprite Painting Program
Main Merge Algorithm
$8200
Start of temp. sprite painting storage
$9000
Temporary storage of color or pattern tables being juggled in the sprite Painting program
Fill3:color-fill routine from Paint Master
$9400
Filltable:scene recreator from Paint Master
9500
CTABLE: color palette for Paint Master
9600
Possibly speech synthesis routines
9A00
Applesoft, DOS, monitor
FFFF

When using Applesoft, your Basic program starting address will default to $800 (2048). However, when using Ampersprite you need to start your Basic program at either $4000 (if you will be using either no Apple hi-res page or only page 1 ($2000-$3FFF)), or $6000 (if you will be using hi-res page 2 ($4000-$5FFF)).

To start Basic programs at $4000, you must write a short program to alter the pointers to the beginning of the program. This program looks as follows:

```
10    REM *** LOCATE PROGRAM AT $4000 ***
20    POKE 103,1: POKE 104,64: FOR I = 16384 TO 16386: POKE I,0:
      NEXT I
30    PRINT CHR$(4); "RUN YOURPROGRAM"
```

or

```
10    REM *** LOCATE PROGRAM AT $6000 ***
20    POKE 103,1: POKE 104,96: FOR I = 24576 TO 24578: POKE I,0:
      NEXT I
30    PRINT CHR$(4); "RUN YOURPROGRAM"
```

These pokes must be done prior to running the program that uses Ampersprite, or they will not work.

17. Initializing the SuperSprite

The following sequence of commands activates and initializes the SuperSprite:

1. BLOAD in AMPERSPRITE, I & I, K2, and any sprite, pattern, color, name, or attribute table you will be needing. The three titles above are essential Ampersprite files.

2. CALL 912 -- this will cause ampersand hooks to be put in place, which means that the Apple will now jump to Ampersprite routines (machine language) when it sees the "&" sign.

3. Use the "&I" command, to initialize the SuperSprite board and zero table address storage locations, and then display VDP graphics only.

4. Use the "&RX #,#" command to establish background color for text and screen backdrop color and text color.

5. Use the "&RG1" or "&RG2" or "&RT" or "&RU" command to establish Graphics I, Graphics II, text or multicolor mode.

6. Use the "&RE0" command and the "&RD1" command. Using "&RE1" or "&RD0" are not for normal use and go beyond the scope of StarSprite I. (An exception is occasional &RD0 use in the example subroutines at lines 9-10 and &RD1 in the subroutine at line 22009).

7. Pick a sprite size with "&RZ1" or "&RZ0". The former command (size 1) is the one normally used; the latter command makes sprites that are too small for most applications.

8. Pick a sprite magnification with "&RM0" or "&RM1". the former command (magnification 0) gives better resolution but smaller size (16 X 16 at size 1) while the latter command (magnification 1) gives lower resolution but larger size (32 X 32 at size 1).

9. Tell the VDP where your data tables are. Here are some normally used addresses for Graphics II mode:

```
10   &RS 6144: REM PUT SPRITE TABLE AT $1800
20   &RA 15104: REM PUT SPRITE ATTRIBUTE TABLE AT $3B00
30   &RP 0 : REM PUT PATTERN TABLE AT $0000
40   &RC 8192: REM PUT COLOR TABLE AT $2000
50   &RN 14336: REM PUT NAME TABLE AT $3800
```

10. CALL 831 which will clear your card's memory to all zeroes. This prevents leftovers or random data.

11. Use the "&L" command to load all VDP data tables (such as sprite attribute table) from Apple memory, to VDP RAM memory, where they can be used for VDP graphics effects.

12. Sometimes it is desirable to load the color table from a special CALL rather than predetermined data.

In some cases it is desirable to have the same color data in all 6144 bytes of the color table. Rather than saving a table full of $F1 (decimal 241), we will use the following line from Basic:

```
10   POKE 7,241: POKE 8,0: POKE 9,0: CALL 852
```

You may now proceed with your program. The following commands are the most frequently used:

```
&AA sprite plane,Y-coord,X-coord,sprite number,color
                (update all sprite attributes)

&AX sprite plane,X-coord
                (change horizontal location of sprite)

&AY sprite plane,Y-coord
                (change vertical location of sprite)

&AN sprite plane,sprite number
                (change priority of sprite)

&AC sprite plane,sprite color
                (change color of sprite)

&C offset,color byte
                (place data in color table)

&N offset,name table byte
                (place data in name table)

&P offset,pattern byte
                (place data in pattern table)

&L Apple High,Apple Low,Counter,VRAM High,VRAM Lo
                (upload data from Apple RAM to VRAM)

&RM1            (set magnification 1)

&RM0            (set magnification 0)
```

When switching video screen switches to select/deselect Apple or VDP graphics, try the following subroutines contained in the sample programs:

```
GOSUB 9 for Apple video only
GOSUB 22009 for VDP video only
GOSUB 23009 for mixed video
```

GOSUB 9 works because of POKE 255,1: POKE 254,128: CALL 768: GOSUB 23009.

GOSUB 22009 works because of POKE 49395,0: POKE 49398,0 and POKE 255,1: POKE 254,195 (or whatever goes in register 1): CALL 768;

GOSUB 23009 works because of POKE 49396,0: POKE 49398,0.

Use POKE 49396,0: POKE 49397,0 for Apple video only without VDP blanking filter, which improves the display.

POKE's to 49395 through 49399 affect soft switches that control the VDP, video mixing, and related functions.

The above GOSUBs can be found in many of the programs in the StarSprite system.

Use POKE 49399,0 to Reset the VDP and center Apple video.

18. Using Ampersprite for Graphics

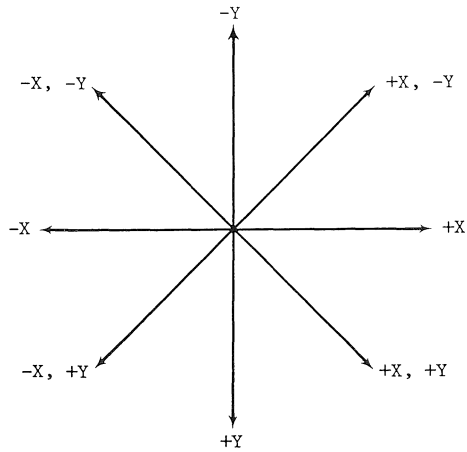
18.1. Animation

If you have not run the Ampersprite graphics tutorial on side A of the main program disk, please do so now. Study the program, then read any parts of the manual you may have skipped until now.

Complex animation using Ampersprite is explained thoroughly in the StarSprite II system. Simple animation will be explained here.

To animate is to move. This means either changing coordinates of existing sprites or changing sprite numbers of existing sprites, or both.

If you change X coordinates sprites move sideways. If you change Y coordinates sprites move up or down. If you change both X and Y coordinates sprites move diagonally. Increasing X moves a sprite rightwards; decreasing it moves a sprite leftwards. Increasing Y moves a sprite downwards; decreasing Y moves a sprite upwards.



A sprite's coordinates define its upper left pixel, regardless of the sprite's color, magnification, or size.

In Ampersprite you change an X coordinate by:

```
& AX 3,49
```

In the above example, the sprite in video plane 3 had its X coordinate changed from whatever it was to 49.

The syntax of the & AX command is:

& AX video plane number, X coordinate

Changing the Y coordinate of a sprite is accomplished similarly. The syntax of the & AY command is:

& AY video plane #, Y coordinate

Note: the "A" in all the above means sprite attribute table change, and the "&" means Ampersprite command.

Changing an X coordinate from 10 to 210 suddenly is not good animation. You need to slide from one location to another to make the animation sequence realistic. This is often done with a FOR-NEXT loop (see your Applesoft manual). For example:

```
10   FOR E = 10 TO 210: & AX 3,E : NEXT
```

The above Basic line will move your sprite quickly and smoothly from left to right. If this movement is too fast, use a delay loop:

```
10   FOR E = 10 TO 210: &AX 3,E: FOR W = 1 TO 50: NEXT : NEXT
```

The same animation principles apply to Y (vertical) movement. If you want to test your hand at using Ampersprite without having to write the preliminary ("get-ready") lines, simply RUN AMPERSPRITE BASIC (the Ampersprite tutorial program on graphics) and press CTRL-Reset.

GOSUB 22009 in immediate mode (no line number), then type in immediate mode FOR-NEXT commands, Ampersprite commands, and delay loops.

Note: Make sure you are in a place where a shape is moving when you hit CTRL-Reset. Expect moving shapes to be using a low-numbered video plane, such as #0.

You may test out Y coordinate animation if you like, using what you have learned in the X coordinate animation section above.

Let's examine two ways of getting diagonal animation:

```
10   FOR E = 0 TO 159: & AX 0,E : & AY0,E: NEXT
```

OR

```
10   FOR E = 0 TO 159: & AA0, E, E, 8, 6: NEXT
```

In the first example we use E as X coordinate and E as Y coordinate (and we operate on video plane #0, the highest priority plane your VDP has). In the second example we do the same thing but use a different command. In this command the two "A"s stand for **all attributes** are specified. The syntax for the & AA command is:

& AA video plane #, Y coord., X coord., sprite #, color

Notice that I specified 8 for the sprite number and 6 (red) for the color (the last two parameters in the list). Normally, you would not change the color or the sprite number in an animation sequence of one sprite.

Now let's examine sequence animation. Suppose you have a sequence of eight sprites. Each sprite (0-7) is an edited version of the previous one -- legs change their positions only a grid-square or two at a time. The last sprite has cycled around until it is nearly like sprite number 0. To make the figure walk in place (no horizontal or vertical movement) we might use the following:

```
10   FOR E = 0 TO 7 : & AN 0, E*4: FOR W=1 TO 50: NEXT : NEXT
20   GOTO 10
```

Notice that the "W" FOR-NEXT loop is a delay loop -- you may need to increase it in order to make the movement slow enough.

The sprite number, E in the example above, is multiplied by 4 because although Ampersprite is normally dealing with size 1 (16 X 16) sprites which contain 32 bytes, sprite numbers are always based upon size 0 (8 X 8) sprites which contain eight bytes.

It is important to make a distinction between sprite numbers and sprite table numbers. Sprite table numbers are all based on size 1 (16 X 16) sprites, so the numbers 0-7 as **sprite table numbers** are correct. But the correct **sprite number** for a sprite table number is obtained by multiplying it by 4.

A more efficient way to animate a sprite sequence in place is:

```
10   FOR E = 0 TO 28 STEP 4: & AN0,E : FOR W = 1 TO 50: NEXT :
     NEXT
20   GOTO 10
```

& AN selects a new sprite for the table and places it on the specified video plane.

In order to make the animated sprite sequence move horizontally, use the following code:

```
10    H = 0 : & AA 0, 0, 0, 0, 6
20    FOR E = 0 TO 28 STEP 4 : & AN 0,E : & AX 0,H
30    H = H + 1 : IF H > 210 THEN H = 0
40    FOR W = 1 TO 50: NEXT
50    NEXT
```

Does the sprite need to move further per step? Change line 30 so it adds 2 to H rather than 1.

Adjust the speed of movement by changing the 50 in line 40.

If the man is walking down hill you might want to add a line 25 that looks like this:

```
25    &AY 0,H
```

To be certain that your sprite does not disappear off the bottom of the screen (the lowest screen coordinate for Y is 191) you will need to change 210 in line 30 to 159.

Moving many shapes at once is a bit more involved. All you need to do is have a variable that represents the video plane number then use either a FOR-NEXT or an increment algorithm:

```
10    REM START SPRITES ON VIDEO PLANES 0-5 AT Y COORDINATES 0,
      32, 64, 96, 128, AND 160
20    FOR E = 0 TO 210: FOR S = 0 TO 5: &AX S,E: NEXT
30    FOR W = 1 TO 50: NEXT
40    NEXT
```

The above code changes each sprite's X coordinate before incrementing the X coordinate. In order to incorporate animation sequences you might use a program that looks like this:

```
5     H = 0
10    REM AGAIN Y COORDINATES ARE SPREAD OUT FOR EACH SPRITE
15    REM LET'S SAY SPRITE TABLE NUMBERS 0-7 COMPOSE THE FIRST
      SEQUENCE, 8-15 COMPOSE THE 2ND, 16-23 THE THIRD, 24-31 THE
      4TH, 32-39 THE 5TH, 40-47 THE 6TH
20    FOR E = 0 TO 28 STEP 4: H = H+1: FOR S=0 TO 5
30    &AXS,H
40    &ANS, E+(32*S)
50    NEXT
60    IF H> 210 THEN H=0
70    FOR W=1 TO 50: NEXT
80    NEXT
```

Line 20 establishes the sequence FOR-NEXT, makes sure the horizontal coordinate keeps increasing, and establishes the video plane number FOR-NEXT. Line 30 updates X coordinates for all sprite planes since it is inside the FOR S=0 TO 5 loop preceding it. Line 40 updates sprite sequence numbers for all planes (it is inside the FOR S=0 TO 5 loop also). The 32 is the number of sprites per sequence lines 4 (we are dealing with size 1 (16 X 16) sprites). This number (32*S) is added to sprite numbers to load the correct sprite number into the attribute table for each of the six (0-5) planes' sprite sequences.

19. Color Changes

With sprites, color changes are a snap:

```
&AC 0,6
```

The above command turns the sprite on video plane 0 red (6). With the "&AA" command color change is specified as the last parameter:

```
&AA #, #, #, #, color
```

To update an entire 6144-byte Graphics II mode color table to some specific number, simply use the following algorithm, POKEing the new color data into 7:

```
10    POKE 7, 241: POKE 8,0: POKE 9,0: CALL 852:REM THIS ROUTINE
      ASSUMES COLOR TABLE IS AT $2000 AND IS $1800 LONG
```

For a sprite to cycle through all the colors at an approximate rate of four colors per second, use a simple FOR-NEXT loop:

```
10    FOR B = 0 TO 15: &AC 0,B: FOR W=1 TO 200: NEXT : NEXT
20    GOTO 10
```

In the example above, if you do not want the sprite to be black or transparent (colors 0 and 1), have the first FOR-NEXT be from 2-15. The second FOR-NEXT is a delay loop. A delay of 200 is approximately 1/4 of a second, so you should get about four changes per second.

If you need to deal with color table bytes, one at a time:

```
&C #,#      (the first number is the offset past the color table
            starting address; the second number is a color data
            byte)
```

The first number specifies the actual color table byte number that you wish to change; the second number specifies the new data.

To update an entire 8-byte pattern's color, the following sample is recommended:

```
      10    FOR Q = 5776 TO 5783: &CQ, 241: NEXT
```

or

```
      10    FOR Q = 4096 + (8*210) TO 4096 + (8*210) + 7: & CQ,241:
            NEXT
```

The first version (above) gets the job done faster and specifies the byte numbers of the color table (5776-5783) while the second version points out the pattern position number being dealt with (210) and shows that it is the third section of the screen being dealt with (4096 means section 3, 2048 means section 2, and nothing added means section 1.)

20. Pattern Changes and Using Text

The essence of the Sprite Painting program is to change pattern tables and color tables as you paint with sprites all over the pattern plane.

Note: Sprites have no effect on pattern plane bytes, nor do backdrops or Apple video in the background.

The exact nature of the change in pattern and color tables in the Sprite Painting program in particular was explained earlier.

The data stored in both color and pattern tables are inverses of what you would expect. The CHARINV file (full of inverse characters) on Side B of the StarSprite I diskette was specifically designed with inverse characters for Sprite Painting text labeling (option 5) because of this inverse situation.

A normal pattern is a bit mapped 8X8 tile built exactly the same as an 8X8 sprite. The bit-mapping is essentially the same as that used with Apple graphics block shapes, except that the Apple only displays seven bits per byte while the VDP displays all eight bits in every byte.

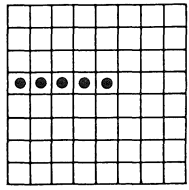
A dash is $FF or 255. A dot can be any one of $1, $2, $4, $8, $10, $20, $40, $80. See below:

Bit number	7	6	5	4	3	2	1	0	
Hex value	$80	$40	$20	$10	$8	$4	$2	$1	
Dec. value	128	64	32	16	8	4	2	1	
Sample byte	1	0	0	0	0	1	0	1	= $85 = 133

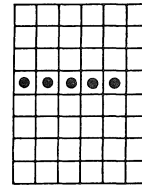
In practice a text dash would be a shape pattern that used seven zero bytes, and in the center of the pattern, one byte with all bits on except for the last three. This allows space between characters, even in the text mode, which uses only the six highest bits of every byte for display, which allows it to have a 40-characters-wide screen rather than 32 wide -- the normal 8-bit-per-byte-using graphics mode.

Now, to explore the exact nature of a dash-shaped pattern in both graphics and text mode, let's look here:

Graphics Mode Dash



Text Mode Dash



As you can see, the same pattern table could be used for both graphics mode text and text mode text. This is the situation with the CHAR file on side B of the StarSprite I disk.

One thing is essential here: the last three (lowest significant bits) pattern bits in each pattern byte must remain off (zero or blank).

In using CHAR patterns, simply subtract 32 from the correct ASCII number of a character to get its proper pattern table number. The best way to see how this can be done is to study TYPE.GR and TYPE.TX, two programs found on side B of the StarSprite I disk. These programs use the CHAR program and allow for text color and background color changes, within certain limits.

Neither of the programs use Ampersprite, to provide examples of dealing with the VDP from Basic without Ampersprite. Now let's look at changing data in a pattern table:

```
10   &P 5776,0
```

The above line changes one byte in a pattern table -- byte 5776. To change the entire pattern that begins at byte 5776:

```
10   FOR Q = 5776 TO 5783: &P Q,0: NEXT
```

OR

```
10   FOR Q = 4096 + (8*210) TO 4096 + (8*210) + 7: &P Q,0:
     NEXT
```

Both of the lines of Applesoft above function identically, however, the first one has the exact table byte precalculated while the second one calculates the proper location in the algorithm itself.

You add 4096 when the pattern will go onto the bottom section of the screen, 2048 when it will go onto the middle section, and nothing when it will go onto the top section. The 210 is the pattern position number the pattern will occupy on the bottom section of the screen. The 8 is because each pattern has eight bytes.

In dealing with pattern and color tables, one way to avoid confusion is to always use a name table that contains the data 0-255, then 0-255 again, then 0-255 again. These 768 numbers will provide a sequential series of pointers to color and pattern table entries.

You will always know that byte 4087 of color and pattern tables will be displayed lower on the screen than byte 3129 (of color and pattern tables) if you use this convention. All Sprite Painting files use this arrangement -- their name table file is called NAMEPAINT and contains the sequential name table arrangement discussed above.

To see how a program like TYPE.TX works with Ampersprite, LOAD it and DEL 60-168, add:

```
PRINT D$"BLOAD I&I": ?D$"BLOAD AMPERSPRITE": CALL 912
```

to line 5, then transfer lines 60-151 in Sprite Painting to TYPE.TX.
Transferring lines from one program to another is to either capture them (see Apple DOS manual, page 76) or LOAD SPRITE PAINTING and LIST 60-151 and then LOAD TYPE.TX and use your editor functions to type over lines 60-151, which should still be on the screen. As you do this please change the &RG2 in line 90 to &RT. The program is ready to RUN. But change it some more: change line 350 to read:

```
350  &N PSN,CH:GOTO 310 and change line 500 to read: 500 &N
     PSN+1, CH:RETURN
```

You have just improved the program's responsiveness and speed: a line of letters (using the REPEAT key) takes seven seconds without Ampersprite but using Ampersprite takes five seconds or (eight characters per second).

For even greater speed, remove some of the IF statements (311-313, 315, 318, 325, and 327).

Before going on, notice that line 500 is doing what lines 500-560 used to do -- and doing it better and easier. You can eliminate lines 505-560 once the new line 500 is in place.

Text is best handled with the "&N #,#" command.

21. Magnification and Size

Size 1 displays 32 bytes of consecutive sprite table data (for 16 X 16 sprites) but size 0 displays only eight bytes.

A sprite that was constructed in the size (16 X 16) will only be partially displayed if size 0 is specified.

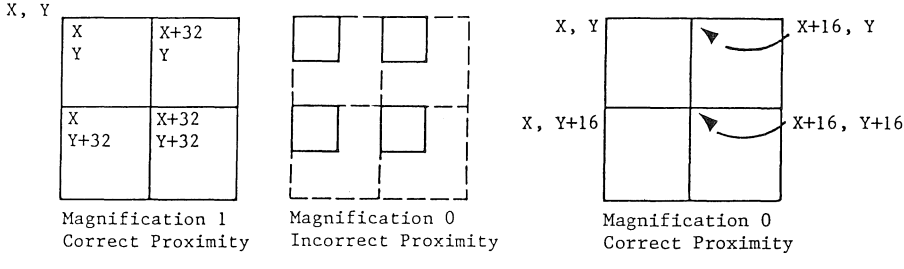
Size 1 is more useful for most purposes except when tiny sprites are needed (8 X 8). When constructing tiny sprites, remember that each of the four 8 X 8 quadrants of your 16 X 16 sprite grid is a **separate** sprite. You will be creating four at once. If you ever use size 1 with them you will see four at once. Here are the Ampersprite commands:

```
&RZ0 enables size 0
&RZ1 enables size 1
&RM0 enables magnification 0
&RM1 enables magnification 1
```

One thing to bear in mind when using these commands is that you must not alter certain zero page locations in programs using "&RZ" and "&RM" commands. Here are those addresses: $DB-$DF, $D4-$D5, $EB-$EF, $19, $7-$9, $1E-$1F, and $F9-$FF. Also avoid $300-$3EA and $800-$DFF.

Size and magnification commands affect all sprites on the screen. To have a mixture of sprite sizes, plan to draw your sprites in different sizes at the start. Or use multiple-sprite shapes.

One factor to consider with multiple sprite shapes is that a magnification change requires X and/or Y coordinate adjustments. Here is a 4-sprite (16 X 16) shape to consider:

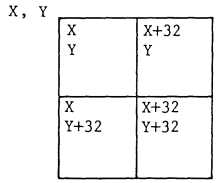


Magnification 1
Correct Proximity

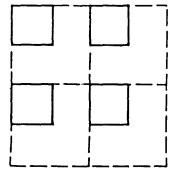
Magnification 0
Incorrect Proximity

Magnification 0
Correct Proximity

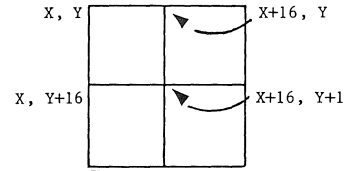
Magnification makes a sprite's area increase/decrease by a factor of 4, while its dimensions change by a factor of 2. Here is a size and magnification table:



Magnification 1
Correct Proximity



Magnification 0
Incorrect Proximity



Magnification 0
Correct Proximity

Pixel size 1X1	Size 0	Magnification 0	8 bytes used	8X8
Pixel size 2X2	Size 0	Magnification 1	8 bytes used	16X16
Pixel size 1X1	Size 1	Magnification 0	32 bytes used	16X16
Pixel size 2X2	Size 1	Magnification 1	32 bytes used	32X32

22. Using Ampersprite for Sounds

22.1. Programmable Sound Generator Registers

The PSG (programmable sound generator) on your SuperSprite has 16 registers which can be either read from or written to.

To make sounds you write to them. (If you decide you need to read from them do the following: POKE 49375, (register number) and then PRINT PEEK (49375) to get contents of register.) Here is what each register does:

Registers 0-5 are for tone generator control. You enter program tone periods in them. Registers 0 and 1 are for channel A, 2 and 3 are for channel B, and 4 and 5 are for channel C.

Numbers up to 4095 may be written to any of these pairs of registers. In each pair, the lower numbered register is the fine tune (low byte) register and the higher one is the coarse tune (high byte) register. For example: for channel A, register 0 is fine tune, register 1 is coarse tune. The higher the tone period, the lower the tone.

Register 6 is the noise generator control. You enter noise periods into this register. Values may range from 0 to 31. Noise frequency is regulated with this register and different types of noises (guns, explosions, waves, hisses, steam engines, etc.) are the result.

Mixer control and I/O port control are the functions of register 7. The register puts up to three tones together with or without sound effects.

Each of the PSG's sound channels may have tone, noise, or both. Any one, two, or all three of the channels may be active at once. And since register 14 holds data used by the programmable sound filters on the SuperSprite, register 7 always has an extra 64 added to it to enable filter control via port A, whose data is controlled by register 14. (Register 7 is one of the two registers dealt with to use the filters, since it controls direction of the I/O ports.)

Registers 8, 9, and 10 are the amplitude (volume) controls for channels A, B, and C respectively. Volume settings may range from 0 (off) to 15 (loud). If a number greater than 15 is put into any of registers 8-10, this signals the PSG (programmable sound generator) to vary amplitude using a sound envelope, defined in higher-numbered registers.

If an envelope is specified, the number placed in registers 8, 9, or 10 (16 or greater) are flags and give no specific volume setting. See the description of the next three registers.

Note: The SuperSprite has an adjustment for sound volume at the top front of the board.

Registers 11 and 12 are set up to hold the sound envelope period. A number as high as 65535 may be put into this pair of registers.

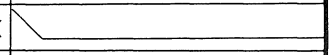
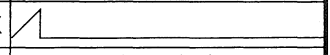
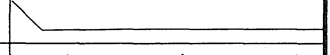
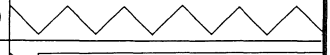
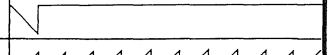
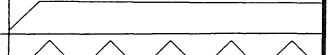
Register 11 gets the fine tuning (low byte) data and register 12 gets the coarse tuning (high byte) data.

The higher the sound envelope's period, the lower the resultant envelope frequency. An **envelope** is a pattern of sound variation using amplitude control.

Tremolo is the name of a sound envelope in which sound amplitude varies at a cyclical rate.

Register 13 is the envelope shape/cycle controller. With this register, whose values may vary from 0-15, we get 10 different shape/cycle configurations. (Values 0-3 are identical in function, and so are values 4-7, so that is why 16 different values give only 10 combinations.)

On the next page are the envelope generation output (amplitude) graphs for all viable values:

Register 13 BITS				ENVELOPE GENERATOR OUTPUT
B3	B2	B1	B0	
CONTINUE	ATTACK	ALTERNATE	HOLD	
0	0	X	X	
0	1	X	X	
1	0	0	0	
1	0	0	1	
1	0	1	0	
1	0	1	1	
1	1	0	0	
1	1	0	1	
1	1	1	0	
1	1	1	1	

Register 14 is the filter control register. This register gets values for the programmable filters and register 7 decides which direction these I/O ports are going (input or output). (Register 7, if you are using Ampersprite, is always on "enable filter port" setting.

Putting 3, 7, 11, or 15 in register 14 specifies an unfiltered sound, and so does any number less than 64 being put into register 7.

Register 15 is unused.

The Ampersprite sound commands allow easy programming of sound effects from Applesoft, with no POKEs necessary.

22.2. Sound Filters

In the Ampersprite language the various sound commands automatically enable register 14. Technically, this means that whatever value gets put into the sound mixer (register 7) gets 64 added to it. Values of 64 to 127 are to be used in register 7 -- it is inadvisable to use 0-63 or 128-255.

Values under 64 or from 128-191 disable sound filters

There is no reason to use values over 127 or under 64 in register 7, the mixer. If you desire to disable filtering, the best way is to put a 3, 7, 11, or 15 in register 14 and leave register 7 alone. This leaves filters enabled, and all you need to do to filter the sound is to replace the 3, 7, 11, or 15 with any other value between 0-15. This way, you never need to touch the mixer in register 7 when you deal with sound filtering.

Filtering sounds is extremely easy. Simply use a "fill GI chip register with specified contents" command: &TR, &BR, or &SR.

And make sure you give this command immediately **after** a tone or noise command (&TN, &TE, &SN, &SE, &BN, or &BE).

The reason is simple: all six of the above commands zero all GI chip registers 0-15 before inserting values related to their specific command. This means that register 14 contains a zero.

This is a "filter on" condition. To turn off filters with a 3, 7, 11, or 15 or filter sound with some value other than 0, you will need to put 1, 2, 4, 5, 6, 8, 9, 10, 12, 13, or 14 in register 13, the filter control register.

Here is how you talk to this register, bearing in mind that "&SR", "&BR", and "&TR" are equivalent and interchangeable:

```
& TR 14,# :REM # is 0-15 (see table below)
```

For example: & TR 14, 5

The example above emphasizes low tones. See the Installation and Technical Reference manual for a more complete description of the functions of the various filters.

The range of frequencies with the PSG is from 15.609 Hz to 63.92 KHz. The A above middle C is 440 Hz. The tone period to put into an Ampersprite command to get 440 Hz is 145. To work out other period values for other notes' frequencies, use:

period value = 63,920.4 ÷ desired frequency

You must round all decimals to the nearest integer value.

If you just want convenient values to use as tone periods, use:

NOTE	OCTAVE	IDEAL FREQUENCY	TONE PERIOD VALUE
C	1	32.703	1955
C#	1	34.648	1845
D	1	36.708	1742
D#	1	38.891	1644
E	1	41.203	1552
F	1	43.654	1465
F#	1	46.249	1382
G	1	48.999	1305
G#	1	51.913	1232
A	1	55.000	1163
A#	1	58.270	1097
B	1	61.735	1036
C	2	65.406	978
C#	2	69.296	923
D	2	73.416	871
D#	2	77.782	822
E	2	82.406	776
F	2	87.308	732
F#	2	92.498	691
G	2	97.998	652
G#	2	103.826	616
A	2	110.000	581
A#	2	116.540	549
B	2	123.470	518
C	3	130.812	489
C#	3	138.592	461
D	3	146.832	435
D#	3	155.564	411
E	3	164.812	388
F	3	174.616	366
F#	3	184.996	346
G	3	195.996	326
G#	3	207.652	308
A	3	220.000	291
A#	3	233.080	274
B	3	246.940	259
C	4	261.624	244
C#	4	277.184	231
D	4	293.664	218
D#	4	311.128	206
E	4	329.624	194
F	4	349.232	183
F#	4	369.992	173
G	4	391.992	163
G#	4	415.304	154
A	4	440.000	145
A#	4	466.160	137
B	4	493.880	129

NOTE	OCTAVE	IDEAL FREQUENCY	TONE PERIOD VALUE
C	5	523.248	122
C#	5	554.368	115
D	5	587.328	109
D#	5	622.256	103
E	5	659.248	97
F	5	698.464	92
F#	5	739.984	86
G	5	783.984	82
G#	5	830.608	77
A	5	880.000	73
A#	5	932.320	69
B	5	987.760	65
C	6	1046.496	61
C#	6	1108.736	58
D	6	1174.656	54
D#	6	1244.512	51
E	6	1318.496	48
F	6	1396.928	46
F#	6	1479.968	43
G	6	1567.968	41
G#	6	1661.216	38
A	6	1760.000	36
A#	6	1864.640	34
B	6	1975.520	32
C	7	2092.992	31
C#	7	2217.472	29
D	7	2349.312	27
D#	7	2489.024	26
E	7	2636.992	24
F	7	2793.856	23
F#	7	2959.936	22
G	7	3135.936	20
G#	7	3322.432	19
A	7	3520.000	18
A#	7	3729.280	17
B	7	3951.040	16
C	8	4185.984	15
C#	8	4434.944	14
D	8	4698.624	14
D#	8	4978.048	13
E	8	5273.984	12
F	8	5587.712	11
F#	8	5919.872	11
G	8	6271.872	10
G#	8	6644.864	10
A	8	7040.000	9
A#	8	7458.560	9
B	8	7902.080	8

The discussion of frequencies and period values above will help you understand what to expect from the filters.

A high pass filter with a 400 Hz cutoff allows only frequencies above 400 Hz to pass through at full volume -- volume is reduced on all the frequencies up to 400 Hz. So if the two A's around middle C are 440 Hz and 220 Hz, then by hitting a couple of keys on a piano, you will be able to see what frequencies we are talking about. The second A above middle C is 880 Hz, the next is 1760, the next is 3520, so this will let you see how 400, 800, 1600, and 3200 Hz cutoffs will effect PSG tones.

"&TR 14,1" gives the lowest sounding harmonies and "&TR 14,12" gives the highest, for comparable pitches. (The high pass filter allows frequencies through that are **higher** than the given cutoff frequency, the low pass filter allows frequencies through that are lower than the given cutoff frequency, and the band pass filter allows frequencies through that are close to and centered upon the given center frequency.

Filters give your PSG the capability of **voices.** They add a whole world of dynamics to the already-versatile GI sound chip. In combination with tremolo and vibrato effects, well-chosen filtering can give you sounds that simulate musical instruments.

For learning purposes, it is best to start with one note at a time, as double and triple harmony are more difficult for the GI chip to handle, more difficult to use filters with, and more difficult to create effective vibrato and tremolo with.

23. Tones

In the Sound Effects Tutorial Using Ampersprite there are examples of the use of the Ampersprite language to create tones.

A good example is the European Police Car, lines 59-80. Lines 65 and 75 are simple delay loops. Line 78 stops the siren action if the tones have sounded over five times. The GOSUB 3 goes to a CALL 3578 which zeroes all registers and therefore turns off all sound. The GOSUB 59999 merely prints "European Police Car" on the screen. Line 79 cycles back to line 60 to keep the siren going. The other two lines are:

```
60    &TN 254,0,0,15,0,0
70    &TN 342,0,0,15,0,0
```

As you can see by looking at your command card, the "T" means tone, the "&" means an Ampersprite command, and the N means no sound envelope will be used.

The first three numbers specify the tone periods for channels A, B, and C, and the last three numbers specify the amplitude values for channels A-C.

In the above example, only channel A is being used. The higher the period number the lower the tone frequency, since the tone period is the inverse of the frequency.

Tones or sounds last forever if you do not turn them off, which is why using the subroutine at line 3 to zero all registers is necessary.

As soon as the subroutine at line 3 is executed, the tone is turned off so it is important to observe that the delay loops are not the delay between tones -- they are the delay **during** tones. The PSG takes virtually no processor time, so to make the tone be longer than a thousandth of a second, you must let the tone be sustained over a period of time.

How will you know what tone period numbers to put in Ampersprite sound commands? Try consulting lines 1330-1341 in the GI (General Instruments PSG) program. Notice that all G's are half what higher G's are and twice what lower G's are. The same is true for the other 11 notes.

In the program, AB means A flat -- and BB means B flat and EB means E flat.

24. Multiple Tones

In the Sound Effects Tutorial Using Ampersprite, there is an example of multiple tones in an Ampersprite program.

Both the Ocean Liner from line 999-1000 and the Train Or Factory's Noon Whistle from line 1299-1300 use two tones and one sound effect each.

Notice that line 1300 is a bit different. After getting all prepared for two tones by use of the "&BN" command, both the mixer register (reg. 7) and the channel A tone period register (reg. 0) are changed. A value of 48 in the mixer register (register 7) specifies tones on all three channels and sound on channel A only.

The 252 in register 0 specifies channel A's new tone period.

The difference is that we have gone from noise on channel A and tones on Channels B and C to noise on Channel A and tones on all three channels (trains usually have three tones or more).

The "&BN" or "&BE" commands normally have the mixer in register 7 (ignoring filter settings) set to 49 so channel A tone is disabled. "&BR" or "&SR" commands would have worked just as well in line 1300, since they are equivalent. (The 49 setting, and all other mixer settings, will have 64 added to them to enable filtering potential).

Line 1000, shows the normal way the "&BN" command.

For three tones, two tones, or one tone, use "&TN" or "&TE". For any combination of sound and tone, use "&BN" or "&BE".

24.1. Sound Effects

A good sound effect is the gunshot found in line 199-200 of the (GI) Sound Effects Tutorial Using Ampersprite. The "&SE" command is used which means sound effects will use a sound **envelope**. "S" stands for sound effect, "&" stands for Ampersprite command, and "E" stands for envelope.

The syntax of the & SE command is as follows:

& SE Noise Period, A vol., B vol, C vol., Envelope Period, Envelope Shape

The first number is the noise period, the next three numbers are amplitude values for each channel; for each channel using the sound envelope the amplitude value should exceed 15. Some values over 15 and others less than or equal to 15 would give mixed envelope use/disuse. All three amplitude values are 16 on line 200, just as the previous value, 9, represents the noise period value. The last two values describe the envelope's shape/cycle. The envelope period of 2250 precedes the shape/cycle value of 0. This last value specifies an envelope that starts loud but trails off fast. The period value is usually a matter of experimentation.

The Explosion sound effect in lines 299-300 is the same as the Gunshot sound except for a different sound period and a longer envelope period.

Notice how the Bombdrop With Explosion sound in lines 499-510 creates the bombdrop first in lines 500-510 and then jumps to line 300 for the explosion. The bombdrop is a simple tone-lowering routine that changes the channel A tone period from 0 to 255. The tone is turned off with the "&TR 8,0" command so the explosion will not have a tone in it. "&TR", "&BR" and "&SR" commands can be used interchangeable -- they function identically.

A more complex sound effect is the Wolf Whistle in lines 699-770 where a man's whistle needs a whooshing air sound to give it realism, accomplished with the one in the noise period register.

25. Sound Effects and Tones Together

In line 999-1000 is an Ocean Liner sound that gives both low ocean liner tones and steam sound simultaneously, to add realism to the sound effect.

The "&BN" or "&BE" commands are needed to get sound and noise simultaneously. The "&" stands for Ampersprite command, the B stands for both sound effects and noise, and the N stands for no sound envelope. (Sound envelopes with tones will give tremolo effects, as we will see in a minute.)

The tone periods 511 and 712 were chosen for channels B and C, while the sound period 1 was chosen for channel A. Amplitudes of 11 for sound and 15 for both tones were chosen also. The delay loop of 1800 loops tells how long the sound will be allowed to continue (over two seconds). An optional way of getting the correct length of time on a sound is to have over two seconds worth of other types of programming occur before using CALL 3578 to shut off the noise and sound.

The following command provides an example of tremolo:

```
1000 &TE 64, 64, 64, 12, 16, 8, 300, 14: FOR QW = 1 TO 1800:
     NEXT: GOSUB 3: GOTO 1
```

Specify the period value 64 in the first three value positions in the above "&TE" command. That is all that is required.

Tremolo wavers volume. **Vibrato** wavers pitch. Observe:

```
700  &TN 0, 164, 0, 0, 15, 0
710  FOR A = 162 TO 166: &TR2,A: FOR B = 1 TO 10: NEXT: NEXT: FOR
     A = 165 TO 163 STEP-1: &TR2,A: FOR B=1 TO 10: NEXT: NEXT
720  GOTO 710
```

In the above vibrato example, register two is pitch control for channel B.

Vibrato is caused by varying this pitch (around 164) from 162-166. Try "&TN 137, 164, 0, 15, 15, 0" next and you will have harmony with only one note doing vibrato. Can you figure out how to vibrato them both, and then add a third part to the harmony, making a chord, and vibrato that too? You have all the information you will need. Try it!

25.1. Using Ampersprite for Sound and Graphics

Boot the StarSprite I demonstration disk, and press CTRL-RESET as soon as the sprite character creature starts pushing the star across the screen.

Type 540 END <Return> to change the program from exit to end.

Type RUN <Return>.

Note: If you do not follow the above instructions, the program will automatically exit before you get a chance to list and study it.

The following is a summary of how the sound and graphics were used together in the program you now have in memory:

Line	Function
(0-2)	Load files
(3)	Initialize ampersand hooks
(30-140)	Initialize VDP and give values and modes.
(150)	Erase screen
(152)	Put 0 (transparent) throughout color table for background and 15 (white) for text. ($F0 is the hex for 240; $F is 15, and $0 is 0.)
(155-165)	Load sprite & sprite attribute tables.
(167)	Use 208 as "quit processing any sprites after this" (from this video plane number on) signal to turn off all sprites for now. Then (GOSUB 22009) display VDP graphics only. Then (the POKE's) change the color table to white text and black background.
(169)	Put name table into VDP memory.
(210)	Load and display Apple graphics scene behind VDP display.
(391)	Load some new attributes into Apple memory.
(400)	Load this new data into VDP at $3B00
(410)	Get note data (which gives the makings of a C chord, an F chord, and a C 7th chord (the TB% really means B flat.)
(415)	Give character something to walk on.
(420)	Animate the three sprites (32 X 32, size 1, mag. 1) of the character and the one-sprite star.
(422)	When the character reaches X=108 quit and stop all sound (CALL 3578) and go on to next part of the program.
(423)	Make sprite sequence continue to go through its sprite numbers (N). Initialize CH at minus one so that line 424 works right after CH gets above 7. (CH stands for which chord number. There are three chords but in a special 7-chord sequence.)
(424)	Chord sequence number chooses which chord.
(425-427)	Play chord
(435)	Shhhhh! Quiet!
(438)	Turn off sprites -- erase walking

The DANCE file contains even more elaborate sound and graphics demonstrations.

26. Error Handling

If you hear your Apple speaker "beeping", and you are not in a routine where such a signal is used as a confirmation (for example, Sprite Painting, or Maze Drawing), your Apple most likely just processed an error. If this happens, these are some of the probable causes and cures:

1. You tried to load a file not on the disk in your drive. You either **spelled it wrong** or the **wrong disk is in the drive.**

   CURE: Run the program again, and spell file names correctly this time, and be sure to switch to a data disk when prompted to do so (sometimes this CAN be the program disk).

   Remember that you can catalog your disk after pressing CTRL-RESET almost anytime. Also, remember that when pattern tables are saved or loaded, .P is appended to their names, and color tables get .C appended to their names, so you need not type that part.

2. Your disk is full (it holds 496 sectors of user-addressable data). CURE: Keep better track of data disk fill up so that when your disk seems pretty full you can INITIALIZE a new one (see DOS manual) and use it as a new data disk. Sector lengths are given before file names when you do a CATALOG.

3. I/O error. Drive door was open, disk media was unformatted, or damaged.

   CURE: Make certain drive door is closed.

   If media is unformatted, use the DOS INIT command to format it (see the DOS 3.3 manual for more information on formatting media).

   If media is damaged switch to a new diskette.

4. CTRL C was hit. In many Basic utilities in the StarSprite system this is a legitimate way to quit what you are doing and return to the main program menu. Data will occasionally be lost in memory, but disk data will be unaffected. Even though usually legitimate, the error beeps will be heard.

5. You ran a program from immediate mode and pointers were set wrong.

   CURE: Reboot and run the program from the disk menus.

6. Write protected. Do this to your BACK-UP COPIES only, not the disks you use -- especially data disks.

7. Bad INPUT response. CURE: If your choices are 1-5 or ESC, hitting N (or any other letters) is incorrect and inadvisable and you will have to choose again.

27. Interrupts and the Status Register

The StarSprite I demonstration disk contains three files not used by the self-running demo. They are:

AMPERSPRITE.INT	(a second version of Ampersprite)
I&I.INT	(the first letter interpreter)
STATUS REGISTER USE	(a BASIC program you may run)

Here is how to run the interrupt demonstration:

- Boot the StarSprite I demonstration disk.
- When the disk drive stops spinning, press CTRL-RESET to break out of the program.
- Type: **RUN STATUS REGISTER USE**

The entire status register demonstration is self-explanatory. The implications of this are not so easily understood, and bear more discussion.

The purpose of the Status Register Use program is to allow you to learn how to use the features of the AMPERSPRITE.INT language.

AMPERSPRITE.INT, differs from Ampersprite in one major way: it allows you to access the status register of the VDP. The implications of this are twofold:

1. You may detect sprite collisions (the coincidence flag in the VDP status register).
2. You may detect any "fifth sprite problem" than may occur in your program.

The fifth sprite problem, as it is sometimes called, is essentially a limitation of the VDP. No more than 4 sprites may be displayed on a given horizontal scan line.

If your program moves more than the allowable four sprites into a given horizontal scan line, the portion of the lowest priority sprite that occupies that scan line will cease to be displayed.

This problem does <u>not</u> affect the four sprites of higher priority.

Note: This limitation is horizontal only, not vertical.

By highest priority, we mean lowest numbered video plane. Each sprite or portion of a multicolored sprite has to have its own video plane. There are 32 sprite planes (0 - 31). The highest priority plane is 0; the lowest is 31. Sprite planes 0 through 3 are never in danger of being obscured by the fifth sprite problem, but the rest are. You must bear this in mind when programming.

The advantage of being able to detect a fifth sprite problem is that you can correct the direction of motion of at least one sprite to clear up the problem well before it is visually detectable.

Although 32 sprites may be on the screen at once, the four-sprite horizontal limit must be carefully monitored. The best programming practice is to move only four objects as sprites, and manipulate all the rest of your objects as pattern by changing the name table.

Note: the pattern plane never interferes with the sprite planes, but there is no hardware way to detect collisions between areas or objects on the pattern plane and sprites.

The way you check for a fifth sprite problem is to PEEK location 233. If the value is greater than 0, a fifth sprite problem is occurring, otherwise everything is fine.

If a fifth sprite problem is occurring, location 234 contains the location of the offending sprite.

Using the information in these two locations, you will be able to effectively avoid the fifth sprite problem.

For examples of the use of these locations, see lines 293 through 295 of STATUS REGISTER USE. Run this program until you are comfortable that you know what it does.

Now change line 279 so that other higher number sprite planes are involved. For example, &AA4, 99, 96, 128, 13 could be changed to &AA31, 99, 96, 128, 13 -- now 31 will be the "offending" sprite. StarSprite III goes into the use of the status register from machine language, and StarSprite II allows you to use it from BASIC.

For collision checking, you simply PEEK location 232. If a collision occurs the contents will be non-zero, and if no collision occurs a zero. For example:

```
10 C = PEEK (232) : IF C <> 0 THEN some sort of collision routine
```

To find out which sprites collided, check backwards through the last sequence of programmed sprite movements until you find the culprit.

Refer to line 193 in Status Register Use. For a clearer idea of where to use AMPERSPRITE.INT and collision detection, think of the "shoot 'em up" games you have played. For every event of a bullet or a laser hitting an alien or asteroid, a collision must be detected for the game to function properly.

Why the suffix INT? This version of Ampersprite uses the interrupt facility of the VDP to time the status register checking. These interrupts are generated when the VDP is finished "painting" a video frame. This is no coincidence as the only safe time to read the status register is at the end of a frame.

So the suffix INT is short for interrupt-driven. AMPERSPRITE.INT handles the turning on (enabling) and off (masking) of interrupts to enable you to perform these checks without having to worry about the various machine language protocols to observe.

To enable VDP interrupts:

CALL 3826

To enable 6502 interrupts:

&D (for "do interrupts")

To disable VDP interrupts:

CALL 3838

To disable 6502 interrupts:

&Q ("quit doing interrupts")

It is essential that you enable both VDP and 6502 interrupts when you want to perform status register checking, and that you disable both before leaving your program. If you do not issue the

CALL 3838 : &Q

sequence, your Apple may do unpredictable things (at unpredictable times), and your disk I/O is in severe danger!

Note: The files I&I.INT and AMPERSPRITE.INT replace the files K2, I&I, and AMPERSPRITE, so for interrupt-driven processing, you need not load K2.

It is essential that you issue a CALL 3826 before the normal call 912 to make this version of Ampersprite work.

Again, and this cannot be stressed enough, you must shut off interrupts before exiting AMPERSPRITE.INT by issuing a:

CALL 3838 : &Q

You will probably not need to use &D or &Q between your program initialization and end because all the necessary instructions for status register checking are built into AMPERSPRITE.INT.

Referring to the Status Register Use program, notice that CALL 3826 was issued not only before the CALL 912 in line 3 (an initialization step), but also in line 145 just after the initialization commands and before the CALL 831 that clears video RAM.

If no calls to 3826 are issued, or if the &Q command is issued, AMPERSPRITE.INT functions just like Ampersprite (normal version). This precludes the use of the status register for collision detection and fifth sprite checking.

To use this version of Ampersprite correctly, you must define, or mark the last sprite in your attribute table. If you have N sprites displayed on the screen, you must set the Y attribute of the N+1th sprite to 208. This is done by issuing the command:

&AYN+1,208

If you fail to make this definition, collisions and fifth sprite problems will be detected for sprites that are either off the screen, transparent, or both -- in any event not the sprites you are tracking.

If you look at line 180, you will see that sprite planes 2 through 31 are shut off. This was done by issuing the command:

&AY2,208

Specifying a Y coordinate is a VDP signal to turn off all higher numbered sprite plane processing, including the plane specified. If you are using 7 planes (0-6), make sure to use the &AY7,208 command before your main program begins.

Progamming Hint: To speed up program execution in collision handling, rather than a series of IF statements, try dividing the X and Y coordinates by a constant (say, 10), and using ON X GOTO or ON Y GOTO.

StarSprite II will contain a sample game that is totally dependent upon status register checking.

28. Utilities for Table File Generation

28.1. Methods of Designing Sprites

After booting side B of the StarSprite I disk, you will see six options on the main menu. Option 3 is Making/Editing Files Used By Games.

Type 3 and read the Sprite Making/Editing & Scene Examination menu.

Option 1 is the file named Sprite Maker (the menu name is Shape-To-Sprite-Convertor), and can be used to transform parts of the following Apple graphics into sprites:

- block-shapes
- hplot shapes
- vector shapes
- binary pictures

It also allows creation of new sprites.

If you are going to be creating new sprites (or by editing pre-existing sprites), use either option (2) Single Sprite Editor or option (3) Multiple Sprite Editor.

The Single Sprite Editor is for 16 X 16 sprites used in one-sprite shapes, (size 0 sprites -- which are 8 X 8 -- may be created with this utility as well).

The Multiple Sprite Editor can be used for either one-sprite shapes or multiple-sprite shapes, whether single color or multi-color.

One last word before moving on to an examination of Sprite Maker. There is one more way to create a sprite that has not been discussed: using VDP pictures as the source.

To turn portions of a picture into a sprite, you must choose the Paint With Sprite option from the StarSprite disk's main menu (side B), then either load in a picture or begin painting one.

Once you are satisfied with the painting, and wish to create a sprite from a portion of it, press Q to quit and use the option to create a sprite from part of your "painting".

28.1.1. Sprite Making

Once you have entered a sprite making program (as directed above), place a data disk in your disk drive and type 2 for option (2) Draw New Sprite Starting Fresh.

Note: A data disk is an initialized, DOS 3.3 floppy disk.

You may press ESC at any time for a summary of the commands available.

A grid will be drawn on the screen. That is your 16 X 16 sprite-creation grid.

Press ESC now to see the commands.

I moves the cursor up

J moves the cursor left

K moves the cursor right

M moves the cursor down

These are Apple's normal editing keys -- to be used, in editing, following the hitting of the ESC key.

P means plot (fill in a square in the grid)

E means erase (open up a previously filled in square)

ESC takes you to the menu you should now be reading

Q allows you to quit.

Use the Quit option to:

- start your shape over
- start your sprite table over
- draw another sprite
- save a sprite table to disk
- save the current sprite in memory
- load in a shape or scene
- quit computing altogether

Press any key to return from the commands menu.

Now press K several times in a row, then M several times in a row. Observe the movement of the flashing cursor on the grid.

Next, try the I, J, K, and M keys while holding down the REPT key.

Try pressing P occasionally and notice that the square on which the cursor rests is filled in. Return to some of the white squares and use E to erase them.

That is about all you need to know about creating sprites from scratch.

Once your sprite satisfies you, press Q to quit, save current sprite in memory, and use option 2 to Draw New Sprite Starting Fresh.

Note: If you do not save the current sprite in memory, that portion of your work will be **lost forever.**

When plotting sprites, a tiny square to the right of your grid is displayed. This is the actual-size display of what you are creating. Use it to figure out if the figure you are working on is going to make an effective sprite shape or not.

28.1.2. Block, Hplot, and Vector Shapes

If you have no block (bit-mapped), hplot, or vector shapes (see your Applesoft Manual, pages 91-100) you may skip this section.

A **block shape** is a block of bytes, in table form, that, loaded properly onto the Apple hi-res screen, create a specific shape.

An **hplot shape** is a table containing X and Y coordinate data of a group of points on the hi-res screen which will be connected up with lines, as well as a number representing how many coordinate sets the shape contains.

A **vector shape** is a vector-plotted Apple shape as described in the Applesoft Manual on pages 91-100.

Ignore the parts about SHLOAD and instead remember to save tables as BINARY FILES on disk. To get the address and length of a binary file, after loading it, type

PRINT PEEK (43634) + PEEK (43635) *256, PEEK (43616) + PEEK (43617) *256

and hit RETURN. A binary file is merely a copy of memory, saved as binary codes but displayed, in machine language, as hex numbers.

Three files have been included on side B of the StarSprite I disk to provide a sample of each of the three types of shapes mentioned above: T2 (HPLOT), MANC (BLOCK), and CHAR (VECTOR).

For a demonstration of saving shapes from one of the shapes mentioned above, choose option (5) LOAD BLOCK SHAPE OR HPLOT SHAPE OR PICTURE TO CONVERT PART OF TO SPRITE from the program menu.

At the "LOAD IN" menu, choose option 1 for block shape.

Choose normal (rather than inverse) saving when prompted.

You will not need to switch to a data disk for this example.

Specify MANC (BLOCK) as your shape table name.

Choose shape number 1, a VT of 20, a VB of 41, an HR of 6 and an HL of 2. You will see a little man on the screen.

Next you will see instructions about using your game paddles to position the four blinking dots to encircle the intended sprite shape.

Warning: Since colors are effected in different ways by the VDP video and the Apple video, only shapes, and not colors, will be reflected in the created sprite. However you are free to choose amongst 16 colors when using that sprite.

The man is taller than the dot enclosure, so you will be saving only 16 dots worth of the little man, (vertically).

Press the Space Bar once positioning of the enclosure is done.

You will see your sprite turn inverse as it is saved to memory.

You will save a short-legged man.

Specify 0 for sprite number (a sprite table's numbers can range from 0 to 63). When asked if you want another one, choose N for No.

At the program menu, select option 5 again to load in an hplot shape.

In the "LOAD IN" menu choose 2 for hplot shape.

This time choose I for inverse saving when prompted. Again do not switch to data disk.

Specify T2 (HPLOT) as the shape table name.

Choose shape number 1 when prompted. You will see a wing plane from above. Next you will see the dot enclosure instructions.

Move the enclosure to the front tip of the plane and "bite off" as much as you can.

Press the Space Bar to convert.

Specify 1 as the sprite number (you have filled up number 0 already.

Press N -- you do not want another one.

From the main menu press 5 to make a sprite from an Apple hi-res vector shape.

Choose 3 for vector shape and N for normal saving.

Do not switch disks.

Choose CHAR (VECTOR) as your shape table name.

Choose 0 for ROTATION, 1 for SCALE, 3 for HCOLOR, 1 for shape number, 99 for X coordinate, and 99 for Y coordinate.

The A in the center of the screen is shape 1. Shape 2 is a B, 3 is a C, 26 is Z, and various other shape numbers produce numbers and symbols.

Next you will see dot enclosure instructions. If you are precise you should be able to fully encircle the A. Specify 2 as the sprite number.

Press N -- you do not want another one.

28.1.3. Pictures (Binary)

Before saving the sprite table (with 3 sprites) to disk, add two more sprites: one from a binary picture and one you scratch-build.

Choose option 5 again -- this time to load a picture from disk.

Choose option 4 in the "LOAD IN" menu to load in a 33-34 SECTOR BINARY PICTURE.

Choose normal saving.

Specify PICTURE (BINARY PIC.) as your picture name. Do not switch disks either before or after the picture is loaded.

Once you see the picture and enclosure instructions move the dots until they enclose the brave knight's-sword-holding hand and press the Space Bar.

Specify 3 as the sprite number.

Choose not to have another one.

28.1.4. Plot A Sprite

In the program menu choose option 2 to Draw A New Sprite Starting Fresh. Use the I, J, K, M, P, E, ESC, Q keys (as documented above) to construct a sprite of your own.

Press Q to quit and choose to save current sprite in memory.

Save your sprite as sprite 4.

Use option 4 to save the table to disk. When asked if 5 is the correct number of sprites in the table press Y for Yes.

Specify sprite table name of your choice (for more information on file names, consult the DOS 3.3 manual).

Switch to an initialized data disk when prompted and switch back to program disk when your disk drive shuts off.

In the program menu press <ESC> to quit, which will bring you back to the main disk menu.

Choose option (3) Sprite Making/Editing and Scene Examination.

In the Sprite Making menu choose option 2 for Single Sprite Editor.

In the program menu choose option 5 to Load In Sprite Generator Table.

Specify the same name as you used above to save the table as table name. Switch to your data disk when asked to.

When asked if you want to see all the sprites at once, choose Yes by pressing Y.

Specify 5 as the number of sprites in the table.
The table of sprites you created above should be displayed on the screen.

Notice that the second sprite (1) is black on white (inverse) while the others are white on black. That is because we specifically asked for inverse on that sprite.

Also notice that there are 64 squares on the screen grid. The upper left corner is sprite 0 and the lower right corner is sprite 63 (a total of 64 possible sprites).

Sprite tables may be up to $800 in length. $800 is 2048 in decimal. 2048 is 32 times 64. This means each of the 64 sprites that will fit into a sprite table contain 32 bytes.

In the program menu, choose option 6 to Place A Sprite From Table On Grid.

For this example, place sprite number 0 on the grid (notice how sprites are called "16 X 16").

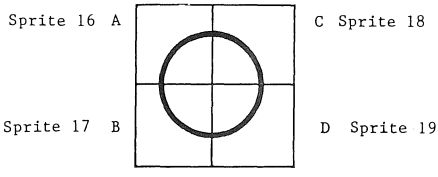
The short-legged man should appear on the grid. Notice that the 16 X 16 grid is divided into four 8 X 8 grids.

All 16 X 16 sprites are made up of four 8 X 8 sprites (although they are regarded by the VDP as a single entity).

Size 1 sprites are 16 X 16. In Ampersprite, you must specify this by issuing the command:

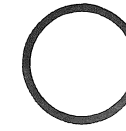
&RZ1

The following diagram shows how 16 X 16 sprites are organized:



(A 16X16 sprite of a circle)

Size 0 (8X8)
(Quarter circle)



Size 1 (16X16)
(Full circle)

The term "size" is actually a misnomer. The size parameter does not specify how large the sprite is, but rather how many pixels it occupies in magnification 0 (either 8 or 16).

In size 0, only one 8 X 8 pixel sprite is displayed, but in size 1, four 8 X 8 sprites are displayed as one (in the organization depicted above).

Ampersprite has been written to make most efficient use of 16 X 16 sprites, as these have proven the most desirable in actual use. Note that 8 X 8 sprites are **not** precluded, but they must be specified with the &RZ0 command.

When a 16 X 16 sprite is referred to in a sprite table, its relative sprite number is multiplied by 4 (the number of 8 X 8 sprites it takes to make one 16 X 16). For example, to tell Ampersprite that we want to place the 16 X 16 sprite number 5 on sprite plane 0, we must use the & AN command as follows:

&AN 0,20

Size 1 (16 X 16) sprites are numbered as follows:

0,4,8,12,16,20...

When using 8 X 8 sprites, the multiplication by 4 rule described

above does not apply, so 8 X 8 (size 0) sprites are chosen from the table in the following order:

1,2,3,4...

Editing

Return to the program menu and choose option 7 -- Edit Sprite On Grid.

Move the cursor down as described earlier, and plot feet on the man using the P command.

This will change 8 X 8 sprites 1 and 3, the bottom sprites.

Observe the changes on the tiny "actual-size" man at the right. Is the sprite better now? If so, choose to save current sprite in memory and specify a sprite number of 0.

Now use options 8, 9 and 10 to rotate, inverse, and mirror the grid sprite. Any time you like what you see you may save the new sprite just as you did above.

Note: You may save a sprite as:

1. The sprite number you are currently editing

2. A new sprite number not yet used

In doing animation all you need to do to your sprites is to save a set of several slightly changed versions of the sprite.

Legs in several different positions can depict walking.

Waving, blinking, talking, and jumping are other animation exercises you may want to try.

If a sprite table has been constructed that changes the positions of a man's legs three different ways and the sprite numbers for these leg positions are 5, 6, 7, and 8, we may program a sequence, using Basic's FOR-NEXT command, to move the man's X coordinate one pixel (dot) or 2 at a time, and cycle his sprite table number from 5-8, then restart the cycle:

X coord.:	20	22	24	26	28	30	32	34	36	38	40	42	44
Sprite table #:	5	6	7	8	5	6	7	8	5	6	7	8	5
Sprite #:	20	24	28	32	20	24	28	32	20	24	28	32	20

Editing, allows saving a sequence of progressively changed sprite shapes without having to "draw" any of them from scratch, except for the first one.

The rotations, inverses, and mirror images are useful for making animation sequences that are the exact reverse of a previous one.

For example, if you have created a sequence of a man walking from right to left, and you realize you also need him to walk from left to right, you may simply mirror the old sequence.

Load shape 5 by using option 6 to Place A Sprite From Table On Grid

Use option (A) Mirror Image of Sprite On Grid.

Use 3 to Save Current Sprite In Memory and specify 9 as the sprite number.

Now mirror 6 and save as 10, mirror 7 and save as 11, mirror 8 and save as 12.

The sequence from 9-12 is the reverse of the sequence from 5-8.

Suppose you have need a tank that shoots in eight directions: north, east, south, west, northeast, northwest, southeast, and southwest.

You can start with one sprite, a north facing tank. The north-facing characteristic is appropriate only for north-facing shooting, so the entire tank must rotate 360 degrees in steps of 45 degrees.

The first thing to do is to save the north-facing tank sprite as, let's say, sprite 0.

Next select option 8 to Rotate Sprite On Grid and save a 90 degree rotation as sprite 2, a 180 degree rotation as sprite 4, and a 270 degree rotation as sprite 6.

Next, draw a northeast-facing version of the tank (with a northeast-facing turret) and save it as sprite 1.

Rotate this sprite 90 degrees to obtain a southeast-facing tank and save as sprite 3, rotate 180 degrees for #5, and 270 degrees #7. (you will only be using the 90 degree rotation option since a 90 degree rotation of a sprite that has already been rotated 90 degrees is a 180 degree rotation).

The techniques of mirror imaging and rotation can save a great deal of time in the creation of animation sequences.

The inverse option can be used for special effects, such as the ZAP sequence in the StarSprite Maze game.

29. Multiple Sprite Shapes

It is possible to get 32 X 32 sprites with the simple Single Sprite Editor program discussed above, but for larger shapes or multiple colors in sprite shapes, you will need the Multiple Sprite Editor program.

To run the Multiple Sprite Editor, select option 3 Sprite Making/Editing & Scene Examination from the StarSprite disk menu (side B), then choose option 3, Multiple Sprite Editor in the Sprite Making/Editing menu.

You will be able to create 64 X 64 sprites in this program (composite sprite clusters composed of four 32 X 32 sprites).

Note: There is really only one type of sprite -- 8 X 8 pixels.

If you use size 0, this 8 X 8 pixel matrix is all that is displayed.

If you use size 1, a 16 X 16 sprite will be displayed (which represents four 8 X 8 sprite numbers in a row to make up its separate parts).

You will **not** get a larger version of your sprite by going from size 0 (&RZ0) to size 1 (&RZ1). Size 1 means only to use four sprites in a row for your screen display, while size 0 means use only 1.

Magnification, on the other hand, will give you a larger sprite.

An 8 X 8 sprite will turn into a 16 X 16 blocky-looking sprite (still using only eight bytes of data), when magnification 1 is specified (&RM1). Similarly, a 16 X 16 sprite will be displayed in 32 X 32 pixels when magnification 1 is selected.

Magnification 1 takes each pixel on the screen and display it as a square of pixels 2 pixels wide and 2 pixels high.

Magnification 0 gives a 1-to-1 correspondence between data and display, at the pixel level.

A **pixel** is the smallest unit of sprite-building material available. The 2 X 2 pixels of magnification 1 are slightly blocky but still effective for certain displays.

To reiterate, if you use size 1 rather than size 0, four (rather than one) sprites (8 X 8) will be displayed at once, giving a 16 X 16 shape.

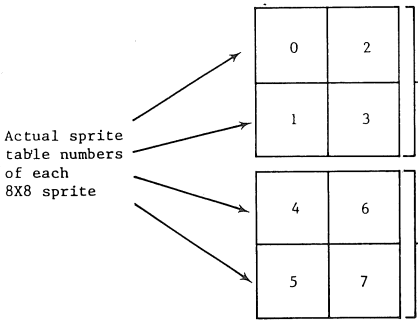
If you switch from magnification 0 to magnification 1 each of these four 8 X 8 sprites will use 2 X 2 pixels to represent each pixel of sprite table data so your 16 X 16 (four 8 X 8 sprites) sprites will be displayed as 32 X 32 sprite shapes.

If you decide to have two sprites, one on top of the other, while using size 1 and magnification 1, you will have a 32 X 64 "sprite". Or use four sprites in a cluster and end up with 64 X 64 "sprites".

When using multiple sprites together to produce a large, complex shape, the burden is upon you to keep all these sprites at the correct locations to produce a single shape.

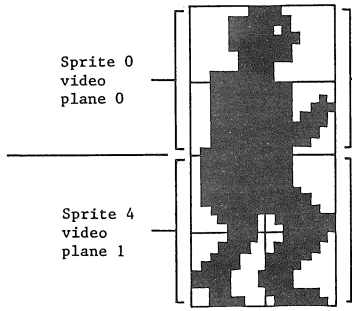
This means that a "32 X 32 sprite" (four sprites displayed as one) is as large as you will ever display with only one command.

Let's pretend we have a 32 (wide) X 64 (tall) man shape now, requiring two commands to deal with his entire shape.



In size 1, all four of these 8X8 sprites can be addressed (dealt with) with only one command; let's call this video plane #0 and sprite #0.

This will be video plane #1 and sprite #4. Notice that each sprite (16X16, size 1, mag. 1 or 2) requires one video plane number for itself.



To move man horizontally both upper and lower sprite must be changed.
&AX 0,40 moves this part of the man.

&AX 1, 40 moves this part of the man. Remember that Y coordinates must have previously been set to the top one being 32 (mag. 1) or 16 (mag. 0) less than the bottom.

29.0.1. Multiple Sprite Inspection

Enter the Multiple Sprite Inspecting program selecting option 3 from the StarSprite I disk menu: Sprite Making/ Editing & Scene Examination)

Next select option 4 (Multiple Sprite Inspector).

Once in the program specify a sprite table name of SPRITE and give 64 as the number of sprites in the table. Note: SPRITE is on the system disk, so you do not need to switch disks.

When asked if you would like to load a sprite attribute table press Y for Yes.

Use the name SPRITE.ATR. Again, there is no reason to switch disks.

Specify 8 as the number of sprites in the sprite attribute table.

If all went well, you should be looking at two versions of a dwarf.

Eight video planes will be displayed on the screen, each of which contains one 32 X 32 sprite (magnification 1, size 1). Both top halves and both bottom halves contain two sprites.

Notice that hats and faces are different colors. So are legs and boots.

Sprite 0 is a face that faces right. Sprite 1 is an arm that faces right. Sprite 2 is the shirt and shoes (facing right). Sprite 3 is the hat and collar (facing right).

Sprites 4 through 7 are the mirror images of 0 through 3 (so the man can face left as well as right).

Note: The numbers 0 through 7 discussed above are the sprite table numbers. To use these sprites in Ampersprite commands, you must multiply the table number by 4 (these are 16 X 16 sprites).

Each sprite can be given its own color and position on the screen. These are called attributes.

Location of a sprite is specified in terms of the location of the upper left corner of the sprite.

In the example on your screen, the face and hat sprites are given the same X and Y coordinates (they go in the same place).

The leg and shoe sprites are 32 vertical units lower than the face and hat coordinates, so they have the same X coordinate, but

the Y coordinate is 32 greater.

To examine why the face and hat sprites are placed at the same coordinates, get back to the main disk menu and select option 3 (Sprite Making/Editing & Scene Examination).

From the Sprite Making menu, select option 3 (Multiple Sprite Editor).

Select option 5 (Load in Sprite Generator Table).

When asked if you would like to see all the sprites in the table, type Y for Yes. Say there are 64 sprites in the table.

There are fewer than 64 sprites in the table, so the empty locations in the sprite display are blank.

The reason the dwarf sprites seem not to be consecutive in memory is that face and hat sprites need not change during animation, but leg and shoe sprites do change.

Follow downwards from the second, third, sixth and seventh sprites in the top row. Each vertical column of sprites is an animation sequence of seven shapes.

Notice that underneath the hat and face sprites there are only blank or unrelated sprites.

As the dwarf walks, only leg and shoe sprites change shape. This is accomplished by specifying the same sprite number for the hat and face for the entire sequence while cycling through the different leg and shoe sprites. Each step in the sequence changes the X (horizontal) coordinate, as well as some of the sprite numbers.

There is actually no such thing as a multicolor sprite; the sprites that appear in more than one color are combinations of several sprites, closely overlaid.

Choose option 6 (Place A Sprite From Table On Grid) and specify table 0. You will need to erase old sprites.

Now use option C to Choose Current Sprite Color and choose color 1 (green), then use option 6 again.

Choose sprite 3 (the relative number in the table). Do not erase old sprites.

The small square at the right shows you magnification 0 representation of the sprite on which you are currently working. The grid provides you with an expanded version of the sprite to facilitate editing.

Notice that the sprites in the same grid quadrant can use different parts of the quadrant, and different colors, thereby allow-

ing multiple colors in what looks like one sprite (actually several sprites overlaid).

Choose option B (Choose Grid # 1-4 For Sprite). Choose grid number 2, and select option 6 to display a sprite.

When prompted for the sprite number to display, select sprite number 2.

Do not erase old sprites. Now choose option C (Color Change), and specify color 3.

Select option 6 to Put Sprite on Grid. Again, do not erase old sprites.

As you may have noticed, there are several sprites overlaid, one on top of another, on this grid. You can get a good picture of how a multicolor set of sprites will look using this technique.

30. Apple Graphics Versus VDP Graphics

There are several major differences between conventional Apple graphics and the graphics generated by the VDP (TMS9918A). One of the first differences, and one of the most important concept involved in the generation of sprites and patterns is that of mapping the eight bits in a data byte to the screen.

When one of the utility programs displays the message **CALCULATING**, what is actually happening is that the program is encoding these data bytes, or decoding pre-existing data bytes for convenient display.

These utility programs function independently of the SuperSprite because, although they design sprites for display on the SuperSprite, they use conventional Apple graphics for the displays.

31. Sprite Animation and Path Testing

31.1. Sprite Animation Testing

From the StarSprite I disk menu, choose option 3 (Sprite Making/Editing & Scene Examination).

In the submenu choose option 6 (Sprite Animate, Sideways).

Choose option 2 (Specify Magnification Factor).

Press 1 for to get magnification 1.

Press 1 for Machine Language Animation; rightwards.

In response to the "default sprites" question, type N for No -- you do not want default sprites.

Specify ET for the sprite generator table name and 3 for the number of sprites in the table.

Switching disks is unnecessary.

Give the sequence 0, then 1, then 2, and then press <Return>.

Now give a step value of 1 for horizontal moves (X coordinate incrementing) and a delay loop high byte of 111.

Read the instructions about CTRL Reset and RUN.

The figure's feet move back and forth too much with these values.

Try a step value of 2.

The action is getting jerky.

Try Step 1 with magnification 0 (using menu option 2). A little better but still not too good.

Here are some of the facts we might learn about sprite sequences from these experiments:

1. Use many (8-32) shapes with small hardly-noticeable changes rather than three shapes with large differences between them.

2. Try to get between 12 and 24 frames per second in speed. (Experiment with delay loop values.)

3. Increment with a step value of one if possible or two if necessary, but no more.

4. Use appropriate orientation. The sprite's walk would probably be more effective if viewed head-on.

5. Use appropriately shaped sprites. ET's feet do not wiggle back and forth like seal flippers -- he sort of rocks back and forth from side to side, with small forward leg movements. He also does not use his hand as depicted.

6. Don't be afraid to have shape's details animated as a sequence progresses. Ears or hands or clothing might swing or flutter, an eye might blink, etc.

You can see now what this Sprite Animation program is used for: Simple testing of rightward animation sequences you have created in sprite tables with the sprite creation utilities in the StarSprite system. Quit with option 3 or else use RUN MENU in immediate mode.

31.2. Sprite Path Testing

Select option 3 from the StarSprite menu (Sprite Making/Editing & Scene Examination) and then choose (7) Sprite Animate, On Paths, from the submenu.

In the program, select option 9 to RUN Animator, and press <Return> for default values for all of the questions asked.

Adjust paddle knob 0 until there are red shapes on a black background (this paddle tests background colors).

Press each paddle button.

One paddle button chooses magnification 0, the other, magnification 1.

Adjust paddle 1 so that the sprites are moving slow, fast, or inbetween. A little less than fully counterclockwise gives the fastest speed. Clockwise gives the slowest speed (100% counterclockwise works also).

causes action to pause, S is the sound toggle (turns the sound on and off), and pressing the keys 0 through 9 turn the sprite on the video plane corresponding to the number typed different colors. You can also test collision checking by changing the collision parameter in the questions you answer. Length and volume of the collision sound are also changeable.

The sprite pauses when done going in a circle. Pauses are put into paths by answering yes to the Do You Want A Pause? question you encountered in the Path Creation program.

Perhaps you noticed in the program menu that there are many file loading and file creating utilities. You could load in PATH3 (from the A side of the disk) by choosing (3) BLOAD Sprite Path Table and using the name PATH3, and then use option 9 to Run Animator.

There is even a subprogram to create path tables the hard way -- inputing numbers from the keyboard. The Alien Path Creator Using Paddles utility is a much easier way to go. If you prefer not to use paddles, you would use the keyboard input utility.

Once you have made sprite sequences for animation, you should either test them in the StarSprite Arcade game (using the option to load in user-created game files before playing the game) or with the Sprite Animate, On Paths utility described above.

32. Making/Editing Files to Be Used By Games

32.1. Maze Creation

From the main system menu, select option 4 (Making/Editing Files Used by Games).

In the submenu, choose 1 (Maze Creation Using Game Paddles).

Note: If you have never played the StarSprite Maze game, do so before reading on.

In the program, select option 2 (Load and See Maze).

Specify NAMEM as the maze file's name (no need for any disk switching).

Like sprites, patterns are based upon 8-bit-per-byte hi-res graphics display, so VDP pattern table pictures do not work on Apple hi-res screens.

The various pattern characters that make up the maze picture are all hplotted one at a time in this maze viewing/creating utility, causing the maze display to be somewhat slow.

The NAMEM file is used in the StarSprite Maze game.

Now choose option 1 (Create Maze) and choose not to have SCORE:00000 put into your maze.

Choose to have maze frame characters POKEed in (into the pattern table).

After reading the instructions, use the paddles and keyboard to create a maze.

Save (S) the maze file on a different disk (switch to a data disk while the file saves).

Use option 2 to Load And See Maze. Use ESC to exit program.

32.2. Maze Direction Table Creator

From StarSprite system menu, select option 4 to get to Making/Editing Files Used By Games.

Next, choose option 3 to get to Maze Direction Table Creator.

In the program press <Return> to continue and give the name of your newly-created maze table. You will need to switch to your data disk while the file loads.

The maze will be displayed on the screen, and you will see "WORKING!" flash on your screen.

When the load is complete, you will see a squadron of dots hopping around on your maze.

What is happening here is that the Apple collision counter is being used to figure out which directions are possible to move in.

Every 32 X 32 section of the maze is tested.

The data representing whether one can move north, east, west, or south from each section is saved in a direction table.

A 0 means the path is not available. A 1 or 2 or 3 or 4 means you can move east, south, west, or north respectively.

You will need to have this table so that your program can know what directions your superbug can travel in the StarSprite Maze game, and you will also need it for the creation of a path table for your maze, which is coming next.

When you save your direction table, give it a name that relates to your maze's name, and switch to the same data disk your maze is on.

32.3. Random Maze Path Creator

From the StarSprite system menu choose option 4 to get to (Making/Editing Files Used By Games).

In the submenu choose option 2 Random Maze Path Creator.

Press the <Return> key to continue and specify the name of your maze's direction table. You will need to switch to your data disk.

When the program asks you to switch back to your program disk may leave the data disk in the drive **only if you go on to finish and save this path table.**

Notice that there will be 32 different paths created, each with 40 different segments. A "segment" means that an alien has moved 32 units in a given direction and is now in a different maze segment than the one it just left.

The direction randomly chosen in any segment must be a possible direction, so before it is recorded it is checked against the direction table's data.

Specify a Maze Path table name that relates to the direction table and maze table. Switch to your data disk if it is not currently in your disk drive.

You now have the three files necessary to run your own mazes in the StarSprite Maze game.

Let's test out your maze in the game now.

Note: Incidentally, make sure your data disk has room for a 34-sector file before beginning or you will get an error and the program will start over.

From the StarSprite system disk choose option 2 (StarSprite Maze).

Once in the game choose option 3 (Alter Game Characteristics By Loading In Game Tables You've Made).

In the submenu, use option 6 to Load Maze Name Table.

Select option 7 (Load Direction Table), then option 3 (Bload Sprite Path Table).

You will need to switch to your data disk but you switching back to the program disk in between file loading is not necessary.

Use ESC to Return To Game Menu.

Select option 1 of the main game menu (Play Game As It Is).

Choose a speed, read the instructions, and play.

32.4. Path Creator (For StarSprite Arcade Game)

From the StarSprite system menu, choose option 4 (Making/Editing Files Used By Games).

In the submenu choose option 4 (Alien Path Creator Using Paddles).

Choose option 2 (Load & See Old Sprite Path Table).

Specify PATH3 for your path table name. No disk switching is necessary.

Give 6 as the number of sprite paths.

The first path will be drawn on the screen, and you will hear a beep.

Press any key and the second path will be drawn. Again, the Apple will beep to signify that it has completed the path.

Once the sixth path has drawn, press the space bar twice to return to the menu and choose (1) Create Sprite Path Table.

Enter 6 for number of sprite paths to make.

You will be creating paths 0-5, starting with 0.

Read the instructions.

Use game button 0 to mark the first path's starting place. Give a step size of 1, 2, or 3 for now.

Move the paddles and press button 0 for each segment's endpoint.

Hit Q when the path is done. Use the Space Bar to erase.

Once paths 0-5 are done you will be prompted for a sprite path table name. You will have to switch to a data disk to save your paths.

If you happen to hit button 0 (to signify segment endpoint) and then hit it again, without first moving the cursor at least five dots (in any direction) from where it just was, you will be asked "Do You Want A Pause Here?".

If you decide to have the animation stop and for the sprite to pause for a bit here, answer Y for Yes, otherwise answer N for No and your last move will be disregarded.

Keep your segments at least five dots long.

Notice that the segments of your paths are allowed to be oriented in eight directions only: north, west, east, south, northwest, northeast, southwest, and southeast.

So when you create 30 degree, 47 degree, or any other angle where segments intersect, it will be corrected to 45 degree, 90 degree, 135 degree, 180 degree, 225 degree, 270 degree, 315 degree or 360 degree. If such a correction would run your segment off the screen, you will be asked to erase it. Erasing is done with the Space Bar.

When you have completed a path table, which must have at least six paths in it, it is time to test it in the StarSprite Arcade Game.

Note: The most effective paths go from one side of the screen to the other.

From the StarSprite menu select option 1 (Play A Game), and in the game submenu choose option 1 (StarSprite Arcade).

In the arcade game, choose option 3 (Alter Game Characteristics By Loading In Game Tables You've Made).

In the User-Made File Loading menu choose option 3 (Bload Sprite Path Table).

Switch disks to your data disk. Type in your file name, then press ESC to Return To Game Menu.

Now, in the game menu, select option 1 (Play Game as it Is).

If you used step values of 5 through 7, your aliens will be traveling incredibly fast and be hard to hit. Step values over 7 make your aliens (at speed 1-5) a blur of color. Try values of 1 through 3 for a reasonably paced game. Use 1 to 10 for game speed.

32.5. Sequence, Limit and Step Table Creation

From the StarSprite menu choose option 4 (Making/Editing Files Used By Games).

In the submenu choose option 5 (Sequence Table, Sequence Limit Table And/Or Step Table Creation).

In this menu choose option 2 (Create Sprite Sequences Table & Also Sprite Sequence Limit Table).

A sequence table specifies the exact sequence of sprites to be used in animation. One sequence is done on one video plane, using from one to eight sprite table numbers.

You will have to give the number of sprites in the sprite table and then the number of sprites (or video planes used) in the sprite attribute table.

The following display will appear:

FOR EACH SPRITE PLANE USED TYPE THE SEQUENCE ORDER IN WHICH THE SPRITES WILL APPEAR:

You will be told to type the sprite table numbers of the sprites in the sequence, followed by <Return>.

When you are finished entering your sequence, press <Return> alone to signal the end.

If the concept of a sprite sequence is not clear, play StarSprite Arcade and use the slowest possible delay loop so that the speed is very slow.

Notice how each shape changes in about eight different ways and then repeats this cycle over and over again.

Suppose you created an alien and called it shape (table) number 0, modified this shape with the sprite editor then resaved it as sprite 1.

This would be the first two sprites in an animation sequence.

If you keep modifying and saving until sprite 7 has been "drawn" and saved, you will have a complex sprite sequence.

Note: Try to remember that in sprite sequence TABLE creation you give sprite TABLE numbers, while in Ampersprite or machine language you give 16 X 16 sprite numbers. For the sequence just created above that would mean that the sprite TABLE numbers are 0, 1, 2, 3, 4, 5, 6, and 7 while the sprite (16 X 16) numbers to use in Ampersprite for these same sprites would be 0, 4, 8, 12, 16, 20, 24 and 28.

Once you save your sprite sequence table you will be asked for your sprite sequence limit table name.

Once you input the name, this will cause both the sequence table and the sequence limit table to be saved on whatever disk is in the drive. (The sequence **limit** table has been created automatically during sequence table creation.)

A sequence limit table tells the arcade game program how many shapes there are in the sprite sequence currently being used for animation. The above sequence has eight sprites in its sequence so the sequence limit number on this sequence would be eight.

32.6. Step Tables

Steps per move means units on the X and/or Y axis.

In the StarSprite Aliens and Asteroids game each of the aliens and asteroids that move on the screen has its own video plane and set of sprite attributes (X, Y, color, sprite number). Each time an alien or asteroid sprite moves, it must move between 1 and 9 pixels per step. A table is kept in memory, and consulted, to determine the step values for each video plane's sprite.

This utility creates a table of step values that will be applied to the sprite moves (separate step values for each sprite video plane's sprite) in the StarSprite Aliens & Asteroids game. Step tables are unnecessary for the other games, because they are already woven into the data of regular StarSprite Arcade path tables, and StarSprite Maze game steps remain constant, even if the speed can have wide variance.

32.7. Sprite Table Creation

This subject is covered in greater detail under Sprite Making/Editing & Scene Examination (above).

In the Sprite Making/Editing & Scene Examination submenu choose option 1 to make sprites from block shapes, hplot shapes, pictures, etc.

Choose option 2 to create sprites from scratch or by editing simple one-color 16 X 16 sprites.

Choose option 3 for making (from scratch or by editing) multiple-sprite shapes and/or multicolored sprites (each color requires one separate sprite).

The way one creates sprites by editing in the sprite editing programs is to:

1. Use option 5 to Load In Sprite Generator Table.

2. If you do not know the number of sprites in the sprite table, input 64.

3. Use option 6 to Place A Sprite From Table On Grid. When you are asked about erasing sprites on the screen, hit Y for Yes the first time at least.

4. Press any key to continue and choose (7) Edit Sprite On Grid. Hit ESC once into that subprogram to see the command options.

5. Press Q to quit and use option 3 to Save Current Sprite In Memory. Give a different sprite number for each sprite you create. Sprite table numbers must be 0-63, (the sprite numbers to use with Ampersprite or machine language must be the sprite table number multiplied by 4. Possible sprite numbers are therefore 0, 4, 8, -- up to 252 (which would correspond to a sprite table number of 63 since 252/4 = 63.

6. Once your sprite table is complete, use option 4 to Save Sprite Table To Disk. When asked what the correct number of sprites in the table is, simply remember the highest sprite number you have given when saving sprites in memory, and add one. (If your highest sprite number was 31, you would input 32, since sprite numbers 0 through 31 are 32 separate numbers.

The way one creates sprites from scratch is to use option 2 to Draw New Sprite Starting Fresh and then use steps 5 and 6 above.

33. Sprite Attribute Table Creation

Examples of sprite attribute tables on side B of StarSprite I are SPRITE.ATR, PAINT.ATR and ET.ATR. The suffix ".ATR" is not required, but is the convention adopted in the StarSprite system.

A sprite attribute table is a table from 4-128 bytes long that gives the characteristics of the sprite on each specific video plane. See the following table:

Attributes of the sprite on sprite video plane 0				Attributes of the sprite on sprite video plane 1				Attributes of the sprite on sprite video plane 2			
Y coord.	X coord.	Sprite number	Color	Y coord.	X coord.	Sprite number	Color	Y coord.	X coord.	Sprite number	Color
43	229	44	6	98	101	0	15	7	199	240	4

Suppose the above 12 numbers were POKEd into memory and (&L) loaded into VDP RAM. You would have a sprite attribute table giving data on the three highest priority video planes, 0, 1, and 2. There can only be one sprite per plane, but all 32 (0-31) planes may be used. (Use &RA (address) to tell the VDP where you are putting your sprite attribute table.)

It is a good idea to add a thirteenth piece of data to that table, in case any stray numbers happen to already be in VDP RAM when you load in the table.

The thirteenth (one more than four times the number of planes used) number should be 208.

This number is a signal not to put any more of the video planes' sprites on the screen. It must be used as a Y coordinate, so in the above example it would follow the 4 and be sprite plane three's Y coordinate. VDP video resolution is only 256 (X) by 192 (Y), so you can see why 208 can work as a signal to the VDP rather than being really seen as a coordinate.

The one time such signals are inappropriate is when a full 128-byte (32 plane) table is loaded in. No more sprites are possible beyond plane 31 (the 32nd plane because of plane 0) so a 208 signal would overflow the table.

Often it is unnecessary to create an attribute table since you merely use Ampersprite commands to give sprite attribute information to the VDP chip. Here is an example of loading the above table in memory, then loading it into the VDP RAM, and finally changing all the attributes of a video plane's sprite at once:

```
10   DATA 43, 229, 44, 6, 98, 101, 0, 15, 7, 199, 240, 4, 208
20   FOR A = 0 TO 12: READ B: POKE 14336+A,B: NEXT: REM 14336 IS
     $3800 -- YOUR ATTRIBUTE TABLE'S ADDRESS
30   &L 56, 0, 13, 56, 0: REM $38 IS 56 IN DECIMAL; CONSULT YOUR
     COMMAND CARD FOR AMPERSPRITE TM INFORMATION
```

```
40   &AA 2, 9, 99, 48, 8: REM SEE COMMAND CARD (AA STANDS FOR
     ATTRIBUTES AND ALL)
```

Another way to load and use an attribute table is simply to give Ampersprite &AA commands to begin with:

```
10   &AA 0, 43, 229, 44, 6: & AA1, 98, 101, 0, 15: &AA2, 7, 199,
     240, 4: &AY 3, 208
```

If you want to create a sprite attribute table by inputing data, choose option 3 (Sprite Making/Editing & Scene Examination) from the StarSprite menu.

In the submenu choose option 7 (Sprite Animation, On Paths).

In the animation program choose option 6 (Create Sprite Attribute Table). You will be specifying Y coordinate, X coordinate, sprite number, and color attributes for each sprite plane's sprite.

Note: The sprite number is derived by multiplying the sprite table number by 4.

One use for a sprite attribute table is with the multiple sprite inspector utility.

The StarSprite I games allow you to load sprite attribute tables (for the more advanced, who can figure out what to so with the files once they are loaded).

The sprite path animation program is an important place not only to input but also to use sprite attribute tables for testing and experiments of all kinds.

34. Files on the Included Disks

34.1. StarSprite I - Side A

1. HELLO - boot program

2. K2 - interface between SuperSprite board and everything else (reads/writes VRAM/registers)

3. CTABLE - table containing 32 eight-byte colors for use with Scene Recreate or Apple graphics scenes in games or educational demo

4. A3 - line drawing routine for scene (Apple graphics) creation

5. AVANTGARDE - compressed logo scene to BRUN in LOGO

6. PDL MAZE CREATE - maze creator for StarSprite Maze game

7. AMPERSPRITE BASIC - tutorial on using Ampersprite

8. EDUCATIONAL DEMO - sample educational application program using Ampersprite

9. EDUCATR - sprite attribute table for use in above Educational Demo

10. CHAR - general pattern table for upper and lower case letters, numbers, symbols

11. STARSPRITE - compressed logo scene to BRUN in LOGO

12. LOGO - introductory program showing Apple graphics and VDP sprite graphics together

13. SQUARETB - direction table for STARSPRITE MAZE game, telling which ways can/cannot be travelled

14. MENU - system menu

15. ARCADEINPUT - StarSprite Arcade game (Basic driver)

16. A&AINPUT - StarSprite Aliens & Asteroids game (Basic driver)

17. FILLTABLE - algorithm for color-filling Apple graphics line-drawing scenes

18. FILL3 - color-fill algorithm for color-filling Apple graphics scenes (fill on white scenes with black lines)

19. STEPA - step table for StarSprite Aliens & Asteroids game

20. MAZE1INPUT - StarSprite Maze game (Basic driver)

21. PDL PATH CREATE - path creator, for aliens, to use in Star-Sprite Arcade game

22. ALIENS - sprite table for StarSprite Arcade game

23. SCENE6502 - color-filled Apple graphics scene for use in educational demo, scene recreate, or games

24. PATH3 - path table for StarSprite Arcade game

25. SCORE - algorithm for scorekeeping in games

26. ANIM5 - machine language portion of StarSprite Arcade game

27. SEQ3 - sequence table for StarSprite Arcade game

28. SEQ3L - sequence limit table for StarSprite Arcade game

29. CHAROBJ - sprite table with all 26 letters and all numbers, and objects (A is for Apple) to go with each letter (16 X 16)

30. ABC - fastload routine

31. ATR - the Ampersprite Basic program's sprite attribute table

32. ANIM6 - machine language portion of StarSprite Maze game

33. CHARMAZE - general pattern table with maze line patterns appended to it so both lines and score can be created in StarSprite Maze game

34. NAMEM - name table for StarSprite Maze game

35. SEQM - sequence table for StarSprite Maze game

36. RND MAZE PATH CREATOR - path creator for StarSprite Maze game (uses direction tables as data)

37. PATHM - path table for StarSprite Maze game

38. SEQML - sequence limit table for StarSprite Maze game

39. ALIENSM - sprite table for StarSprite Maze game

40. DIR TABLE CREATE - maze direction table creator for Star-Sprite Maze game uses maze name table as data)

41. ANIM7 - machine language portion of StarSprite Aliens & Asteroids game

42. A&A - sprite table for StarSprite Aliens & Asteroids game

43. SEQA - sequence table for StarSprite Aliens & Asteroids game

44. SEQAL - sequence limit table for StarSprite Aliens & Asteroids game

45. NAMEA - name table for StarSprite Aliens & Asteroids game

46. NAMEO - name table for general use -- it is blank

47. SEQUENCE & STEP TABLE CREATE - creates either type of table for games or other animation uses

48. I&I - Ampersprite's first letter interpreter algorithm

49. AMPERSPRITE - StarSprite system's core routines

50. SPRITE - sprite table to be used for general purposes or as sprites for LOGO program

34.2. StarSprite I - Side B

1. HELLO - boot program

2. ABC - fastload routine

3. SQUARE - vector shape for sprite editors

4. MENU - system menu

5. TYPE.TX - VDP text mode character typing program

6. TESTTB - block shape scanning/drawing routine

7. TYPE.GR - VDP graphics mode character typing program

8. ANIM3 - algorithm for sprite path animator

9. SPRITE PATH ANIMATE - sprite animation/path tester

10. K2 - interface between the SuperSprite and all other StarSprite routines (reads/writes VDP VRAM/registers)

11. ALIENS - sample sprite table for use in sprite utilities or as StarSprite Arcade game's sprite table

12. SPRITE MAKER - creates sprites from Apple shapes, scenes, etc.

13. TEST 0 (CALL2048) - hplot shape recreation algorithm

14. SPRITE EDITOR - single sprite editor

15. CHARINV - text pattern table, inversed, for sprite painting program

16. SPRITE - sprite table to be used for general purposes

17. CHAR (VECTOR) - vector shape table with letters, numbers, and some symbols, 15 X 15 at scale 1

18. MANC (BLOCK) - block shape table with seven block shapes, 21 X 4

19. ALIENSM - sprite table sample, also used in Maze game

20. T2 (HPLOT) - hplot shape table with one hplot shape

21. A&A - sprite table sample, also used in Aliens & Asteroids game

22. ET - sample sprite table for use in Sprite Animate program

23. ET.ATR - sample sprite attribute table for use with above

24. PATH1 - sample path table for use with Sprite Path Animate program

25. CHAR - general text pattern table for use in typing programs

26. SEQ1 - test sequence table for use with Sprite Path Animate

27. SEQ1L - test sequence limit table for use with Sprite Path Animate

28. AMPERSPRITE - StarSprite system's core routines

29. I&I - Ampersprite's first letter interpreter algorithm

30. PICTURE (BINARY PIC.) - sample black & white 33 - sector 6502 Apple graphics binary scene picture

31. SPRITE.ATR - sprite attribute table for use with SPRITE

32. MULTISPRITE INSPECTOR - inspects multiple-sprite shapes

33. CHAROBJ - sprite table with all 26 letters and all number, and objects (A is for Apple) to go with each letter (16 X 16)

34. LIBRARY - algorithm to display an entire sprite table at once on a 64-sprite grid (uses Apple white graphics only)

35. MULTI.C.SPRITE - sprite editor for multi-color/sprite shapes

36. SQUARE.TEENY - vector shape square for above editor

37. NAME0 - blank name table for general or typing program use

38. ANIM - algorithm for Sprite Animate program

39. SPRITE ANIMATE - program to test moving a sprite sideways

40. TABLE - table containing 32 8-byte colors for use with Scene Recreate or Apple graphics scenes in games or educational demo

41. A3 - line drawing routine for scene (Apple graphics) creation

42. WHITELINE1 - algorithm that doubles all white lines (horizontally) of a binary picture, BRUN it to use it

43. FILLTABLE - algorithm for color-filling Apple graphics graphics line-drawing scenes

44. SCENE6502 - color-filled Apple graphics scene for use in educational demo, scene recreate, or games

45. FILL3 - color-fill algorithm for color-filling Apple graphics graphics scenes (fill on white scenes with black lines)

46. SCENE RECREATE - Apple graphics color-filled scene inspector -- NOT for compressed or normal binary pictures (33 sectors) (use with Paint Master Scene Utility type data files)

47. MERGE - VDP algorithm merging sprites into pattern plane-- used in the Sprite Painting program; 10 sprites per second

48. GI - Ampersprite sound creation tutorial, uses GI sound chip

49. SPRITE PAINTING - program to allow creation of VDP graphics masterpieces, use sprites as brushes and pattern plane as canvas

50. PAINT.ATR - Sprite Painting's sprite attribute table

51. NAMEPAINT - Sprite Painting's name table (sequential)

52. BRUSHES - Sprite Painting's sprite table

53. ZEROTABLES - Sprite Painting's algorithm for clearing out (zeroing) temporary color and pattern tables in Apple memory

54. PATH2 - early test path to use in Sprite Path Animate

34.3. Demonstration Disk (StarSprite I Side)

1. HELLO - booting program

2. DANCER - sprite table for DANCE

3. K2 - interface between SuperSprite board and everything else (reads/writes VRAM/registers)

4. DANCE - sprite table DANCER and FLYING are featured in this sprite animation demo using Ampersprite

5. CHAR - Arcade game demo uses this text pattern table

6. ALIENS - sprite table for Arcade game demo

7. PATH3 - path table for Arcade game demo

8. SCORE - scorekeeping algorithm for Arcade game demo

9. ANIM5 - mach. lang. aspect of Arcade game demo

10. SEQ3 - sequence table for Arcade game demo

11. SEQ3L - sequence limit table for Arcade game demo

12. ABC - fastload routine

13. NAME0 - blank name table for animation or game demos

14. I&I - first letter interpreter for Ampersprite

15. ADJUST - demo disk's SuperSprite boards adjust/align program

16. DANCER.ATR - sprite attribute table for DANCE program

17. FLYING - sprite table for use in DANCE

18. ARCADE - demo of StarSprite Arcade game's animation

19. NAME0PAL - name table for Adjust program's VDP color palette

20. BUTTERFLY - Paint program's sprite table

21. NAMEPAINT - name table for PAINT.P and PAINT.C painting

22. PAINT - butterfly animation on top of a neat sprite painting

23. LOGO - introductory program showing a sprite pushing a star and showing logos

24. AVANTGARDE - compressed logo picture to BRUN in LOGO

25. STARSPRITE - compressed logo picture to BRUN in LOGO

26. PAINT.P - pattern table for painting in PAINT program

27. PAINT.C - color table for painting in PAINT program

28. AMPERSPRITE.INT - Ampersprite language with collision checking routine

29. STATUS REGISTER USE - A program demonstrating Amper-
    sprite.INT

30. I&I.INT - First character interpreter for the Amper-
    sprite.INT

A

AMPERSAND HOOKS, 47

Ampersand hooks are locations that are specify where ampersand locations are located in Apple RAM.

ANIMATION, 4

The process of making an image appear to move.

APPEND, 75

To add to the end of something; for example, a file or a sprite table.

ASCII, 32

American Standard Code for Information Interchange. This is a standard method of storing character data in a computer.

ATTRIBUTE, 20

Sprites have four attributes: Y, or vertical location on the screen; X, or horizontal location on the screen; sprite label; color.

AMPERSPRITE, 15

A language written by Don Fudge which interfaces Applesoft BASIC with the SuperSprite.

B

BACKDROP, 20

The VDP plane behind the pattern plane. The backdrop plane may be transparent, or any or 15 other colors. It is higher priority than the Apple video, but lower priority than any other VDP images.

BINARY, 81

Binary notation is the base two notation for numbers. Computers "think" in binary, and in some instances, it is useful to represent numbers in this notation.

BIT MAPPED, 83

Bit mapping is a method of computer graphics generation. In bit mapped graphics, each bit in memory represents a pixel on the display.

C

CASE

Case refers to whether a letter is capitalized.

LOWER, 9

Not capitalized.

UPPER, 9

Capitalized.

CHANNEL, 62

Channel refers to a discretely controllable, independent signal generator. On the SuperSprite, the PSG has three channels.

COLOR ATTRIBUTE, 5

The color attribute defines the color of a sprite. This is the fourth attribute you must specify for a sprite.

COLOR FILL, 11

The process of filling a high resolution graphics area with a solid color.

COLOR

The VDP has 16 standard (hardware) colors, including black and transparent.

TABLE, 23

A color table is a table that determines the colors of individual pixels on the background (or pattern) plane of the VDP. In Graphics I mode, this table is 2048 bytes in length, and in Graphics II mode, it is 6144 bytes in length.

COMPOSITE SPRITE CLUSTERS, 90

Composite Sprite Clusters are groups of more than one sprite placed on the screen such that they form a larger, or more complex object. One use for composite sprite clusters is to produce the effect of multicolored sprites.

COORDINATE, 19

A coordinate is a location, either vertical or horizontal, of an object. By convention, the X coordinate of an object defines its horizontal position, and the Y coordinate, its vertical position. Unlike many coordinate systems, coordinate 0,0 defines the upper left corner of the screen (0,0 is often a center or bottom left point).

D

DATA DISK, 84

A data disk is a disk that may or may not contain programs, but it must have room for data storage. Data disks are normally working disks, and the contents of such disks should be backed up regularly.

DEFAULT VALUE, 6

A default value is supplied for certain parameters. These values are user-definable, but need not be set each time a program is run since they will take on the default unless otherwise specified.

DELAY LOOPS, 4

Delay loops are "do-nothing" program sequences which are used to slow the speed of a given program. These are particularly useful in sprite animation because sprites can move more quickly than you might desire. A good example of a delay is a BASIC FOR-NEXT loop with no statements in between.

E

EARLY CLOCK, 20

The early clock bit is the most significant bit of the color attribute byte in a sprite attribute table. The early clock bit is ignored if 0, but if it is set on (add 128 to the color value), it shifts the sprite left by 32 pixels. This allows the sprite to bleed in from the left edge of the backdrop.

ENVELOPE, 63, 71

The General Instrument PSG has the facility of varying the amplitude of a tone or noise over time. The shape of this amplitude variance is called the envelope.

F

FILTER, 65, 66

There are 16 programmable filters on the SuperSprite to allow you to achieve special effects with sound generation. These filters block out certain portions of the full tone spectrum producing brighter highs, or "boomier" lows.

G

GRID, 10

A matrix of lines defining a given area. Grids are used to portray the array of pixels that define a sprite.

H

HEXADECIMAL, 25, 29

Base 16 numerical representation. Hexadecimal representation of numbers is extremely convenient shorthand for computers.

HZ, 68

Abbreviation for Hertz, which is the conventional notation for cycles per second.

I

I/O DEVICE, 5

Any device that performs input or output, for example a joystick, a disk drive, or a printer.

INITIALIZE, 47

The process of setting up or perparing. For example disks must be initialized, and the VDP must be initialized.

INTERFACING, 32

The process of making diverse elements interact with one another. For example, the SuperSprite must be interfaced with Applesoft in order to access it from BASIC.

INTERRUPT, 38, 79

An interrupt is a signal generated by an external device that requests that current processing be stopped until its request has been serviced. The SuperSprite can be made to generate interrupts.

INVERSE, 32

Inverse, when used in conjunction with video display, means a black display on a color background, rather than the customary color display on a black background.

L

LO-RES, 22

Short for low resolution graphics mode. This mode of graphics is economical on memory, but is, as the name implies, very low resolution.

M

MAZE, 6

The StarSprite Maze game is one of the demonstration games supplied with the StarSprite I system. This game demonstrates some basic sprite movement techniques, such as sprite paths.

MEDIA

Media is normally used to refer to disks.

UNFORMATTED, 75

As was mentioned under "initialize", disks must be formatted, or initialized, before use.

MERGE, 36

The process of transferring a sprite from its sprite plane into the pattern plane.

N

NOISE, 70

The General Instrument PSG has the capability of generating white noise, which may be varied to give the effect of gun shots, explosions, or combined with tones to produce more exotic effects.

P

PROCESSOR, 69

A processor is an electronic component capable of executing certain instructions. The SuperSprite contains three processors: the video display processor; the programmable sound generator; and the speech synthesis processor.

Q

QUADRANT, 60

A quadrant is normally one fourth of a Cartesian coordinate system. In StarSprite, a quadrant is taken to mean a section of a grid.

R

RAM, 38

Random Access Memory

VIDEO, 25

Random Access Memory dedicated to, and only accessible through the video display processor.

REGISTER

A storage area in a processor. The three processors on the SuperSprite use registers primarily for receiving instructions from the Apple.

MIXER, 70

Register 7 on the PSG--controls which channels are enabled for tone, which for noise, and whether or not filtering is to take place.

STATUS, 37

The VDP register that contains the coincidence flag, and the fifth sprite flag.

VDP, 19

The Texas Instruments TMS9918A Video Display Processor registers.

WRITE-ONLY, 37

Registers dedicated to receipt of data--you may not read the data from these registers.

S

T

A table that defines the path to be taken by a sprite or a series of sprites in an animation sequence.

V

Abbreviation for video display processor.

A processor dedicated to controlling video images--for example the TMS9918A.

The plane on which a graphical image is displayed

A video signal not generated by the TMS9918A VDP such as Apple video.

Z

Locations $00 through $FF in Apple RAM. These locations are special to the 6502 processor, and should only be used if you know you are not conflicting with another running program.

THIS PAGE LEFT INTENTIONALLY BLANK

AMPERSPRITE™ VDP Command Card

(Video Display Processor Commands)

&A SPRITE Attribute commands:

&AA sprite plane number, Y coord., X coord., sprite number, sprite color
(the first A means attributes (sprite); the second A means *all* 4 attributes will be given)

&AA #(<32), #(<256),#(<256),#(<256),#(<16)

&AC sprite plane number, *sprite color* . . . . . **&AC #(<32),#(<16)**

&AN sprite plane number, *sprite number* . . . . . **&AN #(<32),#(<256)**

&AX sprite plane number, *X coord.* (example: &AX 3,25) . . . . . **&AX#(<32),#(<256)**

&AY sprite plane number, *Y coord.* . . . . . **&AY#(<32),#(<256)**

&C number of units of offset past color table address, color data (write to *color table)* . . . . . **&C #**(16 bit),**#**(8 bit)

&I *initialize* StarSprite card (**CALL 912** should happen just before **&I** command)

&L Apple high byte, Apple low byte, counter (<256), VRAM high byte, VRAM low byte (*load* VRAM from Apple memory, use counter=Ø for 256 byte transfer (maximum)) . . . . . **&L#**(<256),**#**(<256),**#**(<64),**#**(<256)

&N position, pattern number (write to *name table)* . . . . . **&N#**(<768),**#**(<256)

&P number of units of offset past pattern table address, pattern data (write to *pattern table)* . . . . . **&P#**(16 bit),**#**(8 bit)

&R VREGISTER write-to commands:

&RA sprite *attribute* table address (16 bit) . . . . . **&RA#**(<=$3F8Ø)

&RC *color* table address (16 bit) . . . . . **&RC#**(<=$2ØØØ or <=$3FCØ)

&RD 1 or Ø (to enable or disable active *display*) . . . . . **&RD1** (normal)

&RE 1 or Ø (to enable or disable *external video)* . . . . . **&REØ** (normal)

&RG 1 or 2 (to enable *Graphics* I or II mode) . . . . . **&RG2** (normal)

&RM 1 or Ø (to enable sprite *magnification* 1 or Ø) . . . . . **&RM1** (normal)

&RN *name* table address (16 bit) . . . . . **&RN#**(<$3CØØ)

&RP *pattern* generator table address (16 bit) . . . . . **&RP#**(<=$2ØØØ or =$38ØØ)

&RR *VREGISTER* number, data (direct register write; not to be used during start-up) . . . . . **&RR#**(<8),**#**(<256)

&RS *sprite* generator table address (16 bit) . . . . . **&RS#**(<=$38ØØ)

&RT (enable *text* mode) . . . . . **&RT**

&RU (enable *multicolor* mode) . . . . . **&RU**

&RX *text* color, backdrop color and text background . . . . . **&RX#**(<16),**#**(<16)

&RZ 1 or Ø (to enable *size* 1 or Ø) . . . . . **&RZ1** (normal)

CALL 804 read VRAM into Apple memory (Applelo=$1E;Applehi=$1F; VRAMlo=$FC,VRAMhi=$FB;counter=$F9...use Ø for 256 bytes)

CALL 831 erase StarSprite card

CALL 768 write to VREGISTER (register #=$FF;data=$FE)

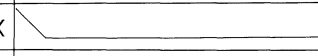
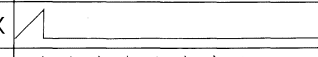
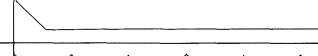
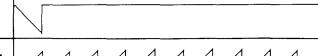
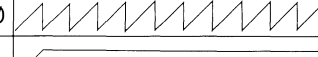
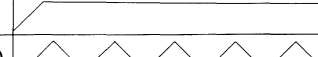
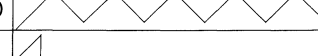
CALL 780 write to VRAM (Applelo=$1E;Applehi=$1F;VRAMlo=$FC; VRAMhi=$FB;counter=$F9...use Ø for 256 bytes)

CALL 852 write to color table (color data=$7;positionlo=$8;positionhi=$9;updates from position on in color table)

For AMPERSPRITE.INT only: &D allows 65Ø2 interrupts, &Q disables 65Ø2 interrupts, CALL3826 for VDP interrupts, CALL3838 disables VDP interrupts.

COLOR VALUES

Ø = transparent
1 = black
2 = medium green
3 = light green
4 = dark blue
5 = light blue
6 = dark red
7 = cyan
8 = medium red
9 = light red
A or 10 = dark yellow
B or 11 = light yellow
C or 12 = dark green
D or 13 = magenta
E or 14 = grey
F or 15 = white

Register 13 BITS				ENVELOPE GENERATOR OUTPUT
B3	B2	B1	B0	
CONTINUE	ATTACK	ALTERNATE	HOLD	
Ø	Ø	X	X	
Ø	1	X	X	
1	Ø	Ø	Ø	
1	Ø	Ø	1	
1	Ø	1	Ø	
1	Ø	1	1	
1	1	Ø	Ø	
1	1	Ø	1	
1	1	1	Ø	
1	1	1	1	

STANDARD BASIC VIDEO SWITCH LINES

GOSUB 9
(standard for *Apple 65Ø2 video only)*

GOSUB 22ØØ9
(standard for *VDP video only)*

GOSUB 23ØØ9
(standard for *mixed Apple 65Ø2 and VDP video)*

AMPERSPRITE™

GI PSG (programmable sound generator) COMMANDS:

CALL 3578 (to cease sound generation)
(T = tone S = sound effects B = both tones and sound effects) (N = NO ENVELOPE) (E = ENVELOPE IS UTILIZED)

&TR or **&SR** or **&BR** (all mean fill "registers" with "contents" and are all interchangeable)
&TR#(<16) (register),#(<256)(contents)
(all commands except **&TR** or **&SR** or **&BR** zero all GI sound registers before performing)

SOUND

&SN noise period &SN register 6, &SN #(<32), &SN #,	amplitude ch. A register 8, #(<16), #,	amplitude ch. B, register 9, #(<16), #,	amplitude ch. C register 10 #(<16) #			
&SE noise period &SE register 6, &SE #(<32), &SE #,	amplitude ch. A, register 8, #(<32)(>15), #,	amplitude ch. B, register 9, #(<32)(>15), #,	amplitude ch. C, register 10 #(<32)(>15), #,	envelope period, register 11, 12, #(<65536), #,	envelope pattern register 13 #(<16) #	

TONE

&TN tone per. ch. A &TN register 0, 1 &TN #(<4096) &TN #,	tone per. ch. B, registers 2, 3, #(<4096), #,	tone per. ch. C, registers 4, 5, #(<4096), #,	amplitude ch. A register 8, #(<16), #,	amplitude ch. B, register 9, #(<16), #,	amplitude ch. C register 10 #(<16) #	
&TE tone per. ch. A &TE register 0, 1 &TE #(<4096), &TE #,	tone per. ch. B, registers 2, 3, #(<4096), #,	tone per. ch. C, registers 4,5, #(<4096), #,	amplitude ch. A, register 8, #(<32)(>15), #,	amplitude ch. B, register 9, #(<32)(>15), #,	amplitude ch. C, register 10, #(<32)(>15), #,	envelope period register 11, 12, #(<65536), #,	envelope pattern register 13 #(<16) #

SOUND & TONE

&BN tone per. ch. B &BN register 2,3 &BN #(<4096), &BN #,	tone per. ch. C registers 4,5, #(<4096), #,	noise period, register 6, #(<32), #,	noise ampl. ch. A, register 8, #(<16), #,	ampl. tone ch. B, register 9, #(<16), #,	ampl. tone ch. C register 10 #(<16) #		
&BE tone per ch. B, &BE register 2,3, &BE #(<4096), &BE #,	tone per. ch. C, registers 4,5, #(<4096), #,	noise period, register 6, #(<32), #,	noise ampl. ch. A, register 8, #(<32), #,	ampl tone ch. B, register 9, #(<32), #,	ampl. tone ch. C, register 10, #(<32), #,	envelope period register 11,12, #(<65536), #,	envelope pattern register 13 #(<16) #

FILTER USAGE: &TR 14,# (filter #) (use right after above commands)

NOTE	OCTAVE	IDEAL FREQUENCY	TONE PERIOD VALUE
C	1	32.703	1955
C#	1	34.648	1845
D	1	36.708	1742
D#	1	38.891	1644
E	1	41.203	1552
F	1	43.654	1465
F#	1	46.249	1382
G	1	48.999	1305
G#	1	51.913	1232
A	1	55.000	1163
A#	1	58.270	1097
B	1	61.735	1036
C	2	65.406	978
C#	2	69.296	923
D	2	73.416	871
D#	2	77.782	822
E	2	82.406	776
F	2	87.308	732
F#	2	92.498	691
G	2	97.998	652
G#	2	103.826	616
A	2	110.000	581
A#	2	116.540	549
B	2	123.470	518
C	3	130.812	489
C#	3	138.592	461
D	3	146.832	435
D#	3	155.564	411
E	3	164.812	388
F	3	174.616	366
F#	3	184.996	346
G	3	195.996	326
G#	3	207.652	308
A	3	220.000	291
A#	3	233.080	274
B	3	246.940	259
C	4	261.624	244
C#	4	277.184	231
D	4	293.664	218
D#	4	311.128	206
E	4	329.624	194
F	4	349.232	183
F#	4	369.992	173
G	4	391.992	163
G#	4	415.304	154
A	4	440.000	145
A#	4	466.160	137
B	4	493.880	129

NOTE	OCTAVE	IDEAL FREQUENCY	TONE PERIOD VALUE
C	5	523.248	122
C#	5	554.368	115
D	5	587.328	109
D#	5	622.256	103
E	5	659.248	97
F	5	698.464	92
F#	5	739.984	86
G	5	783.984	82
G#	5	830.608	77
A	5	880.000	73
A#	5	932.320	69
B	5	987.760	65
C	6	1046.496	61
C#	6	1108.736	58
D	6	1174.656	54
D#	6	1244.512	51
E	6	1318.496	48
F	6	1396.928	46
F#	6	1479.968	43
G	6	1567.968	41
G#	6	1661.216	38
A	6	1760.000	36
A#	6	1864.640	34
B	6	1975.520	32
C	7	2092.992	31
C#	7	2217.472	29
D	7	2349.312	27
D#	7	2489.024	26
E	7	2636.992	24
F	7	2793.856	23
F#	7	2959.936	22
G	7	3135.936	20
G#	7	3322.432	19
A	7	3520.000	18
A#	7	3729.280	17
B	7	3951.040	16
C	8	4185.984	15
C#	8	4434.944	14
D	8	4698.624	14
D#	8	4978.048	13
E	8	5273.984	12
F	8	5587.712	11
F#	8	5919.872	11
G	8	6271.872	10
G#	8	6644.864	10
A	8	7040.000	9
A#	8	7458.560	9
B	8	7902.080	8

Fig. 23 EQUAL TEMPERED CHROMATIC SCALE (f_{CLOCK} = 1.023 MHz)

TABLE OF CHARACTER PATTERN NUMBERS

COLUMN 1				COLUMN 2				COLUMN 3				COLUMN 4			
ASCII		CHAR pattern #	CHAR-AC-TER	ASCII		CHAR pattern #	CHAR-AC-TER	ASCII		CHAR pattern #	CHAR-AC-TER	ASCII		CHAR pattern #	CHAR-AC-TER
dec	hex			dec	hex			dec	hex			dec	hex		
32	20	0	(space)	64	40	32	@	96	60	64	\	128	80	96	┌
33	21	1	!	65	41	33	A	97	61	65	a	129	81	97	┐
34	22	2	"	66	42	34	B	98	62	66	b	130	82	98	┘
35	23	3	#	67	43	35	C	99	63	67	c	131	83	99	└
36	24	4	$	68	44	36	D	100	64	68	d	132	84	100	|
37	25	5	%	69	45	37	E	101	65	69	e	133	85	101	─
38	26	6	&	70	46	38	F	102	66	70	f	134	86	102	|
39	27	7	'	71	47	39	G	103	67	71	g	135	87	103	─
40	28	8	(	72	48	40	H	104	68	72	h				
41	29	9	)	73	49	41	I	105	69	73	i	(the above 8			
42	2A	10	*	74	4A	42	J	106	6A	74	j	characters are found			
43	2B	11	+	75	4B	43	K	107	6B	75	k	in CHARMAZE only)			
44	2C	12	,	76	4C	44	L	108	6C	76	l				
45	2D	13	-	77	4D	45	M	109	6D	77	m				
46	2E	14	.	78	4E	46	N	110	6E	78	n				
47	2F	15	/	79	4F	47	O	111	6F	79	o				
48	30	16	0	80	50	48	P	112	70	80	p				
49	31	17	1	81	51	49	Q	113	71	81	q				
50	32	18	2	82	52	50	R	114	72	82	r				
51	33	19	3	83	53	51	S	115	73	83	s				
52	34	20	4	84	54	52	T	116	74	84	t				
53	35	21	5	85	55	53	U	117	75	85	u				
54	36	22	6	86	56	54	V	118	76	86	v				
55	37	23	7	87	57	55	W	119	77	87	w				
56	38	24	8	88	58	56	X	120	78	88	x				
57	39	25	9	89	59	57	Y	121	79	89	y				
58	3A	26	:	90	5A	58	Z	122	7A	90	z				
59	3B	27	;	91	5B	59	[	123	7B	91	{				
60	3C	28	<	92	5C	60	\	124	7C	92	>				
61	3D	29	=	93	5D	61	]	125	7D	93	}				
62	3E	30	>	94	5E	62	^	126	7E	94	~				
63	3F	31	?	95	5F	63	─	127	7F	95	■				

THIS PAGE LEFT INTENTIONALLY BLANK

ECHO][SPEECH SYNTHESIS

ON THE SUPERSPRITE

Written by Mike and Fern Kory

NOTICE

Street Electronics Corporation reserves the right to make improvements in the product described in this manual at any time and without notice.

DISCLAIMER OF ALL WARRANTIES AND LIABILITY

Street Electronics Corporation makes no warranites,either express or implied except as explicitly set forth in the Limited Warranty, with respect to this manual or with respect to the software described in this manual, its quality, performance, merchantability, or fitness for any particular purpose. Street Electronics Corporation software is sold or licensed "as is." The entire risk as to its quality and performance is with the buyer. Should the programs prove defective following their purchase, the buyer (and not Street Electronics Corporation, its distributor, or its retailer) assumes the entire cost of all necessary servicing, repair, or correction and any incidental or consequential damages. In no event will Street Electronics Corporation be liable for direct, indirect, incidental, or consequential damages resulting from any defect in the software, even if Street Electronics Corporation has been advised of the possibility of such damages. Some states do not allow the exclusion or limitation of implied warranties or liability for incidental or consequential damages, so the above limitation or exclusion may not apply to you.

1140 Mark Ave.
Carpenteria, California 93103
(805) 684-4593

Echo][- Table of Contents

THIS PAGE LEFT INTENTIONALLY BLANK

1. Introduction to the Manual

We have attempted to make this manual both informative and easy to use. It was designed to accomodate the not-so-experienced user as well as the more experienced. Each section is meant to be read from beginning to end and to be used in conjunction with the SuperSprite. By the end of each section you should have not only an understanding of the material covered, but also experience with it.

The 3 main sections of the manual concern

1. Introduction to the ECHO][software and this manual, also some very basic help for the less experienced Apple user

2. The TEXTALKER (TM) software tutorial

3. The SPEAKEASY(TM) software tutorial.

There is also a section on the ECHO software's exclusive features to aid blind users, a technical or advanced section and various appendices. These sections are all outlined in more detail in the table of contents. Turn the page for installation instructions and you are on your way.

2. Installation Instructions

Complete instructions for installation of the SuperSprite board are contained in the Installation and Technical Reference Manual. Please read at least the installation portion of that manual before reading on in this section.

2.1. Testing

A test of the SuperSprite was provided on the Checkout Disk. Should you wish to test the speech portion of the SuperSprite in greater depth, here are some procedures to enable you to do so.

If you have some experience with the Apple, you may test the ECHO portion of the SuperSprite by booting the diskette marked Echo][Software, and running the program called SEC DEMO. This program shows some of the features of the ECHO hardware. To end the program type a Control-C. You should now also copy your diskette. If you do not understand what was just said, don't worry. You will be able to test ECHO speech after reading the following two sections on "KEYBOARD NOTATION" and "BOOTING A DISKETTE."

2.2. Keyboard Notation

In this section you will be introduced to a simple notation system which will show you exactly what to type on your Apple keyboard. For instance, when we want you to type the word "HELLO" you will see:

HELLO

Each bold letter represents a key on your Apple's keyboard. For some characters you are required to hold down one key while pressing another.

For example, to type an exclamation point (!), you must hold dowr the SHIFT key while pressing the 1 key (just as you would on an ordinary typewriter). However, we show only the exclamation point.

!

When it is necessary for you to type a "control character" we will print this:

<CTRL-E>

The key marked CTRL or CONTROL should be held down while you press the other key. So, typing a Control-E is similar to typing a Shift-E except that you press the CTRL key instead of SHIFT. Note that when you type a control character, in most circumstances nothing will appear on the screen.

Special keys such as the keys marked RETURN and ESC will be enclosed in brackets as follows: <RETURN> <ESC>.

2.3. Error Messages

If you accidently misspell something then, most likely, the Apple will reject it by printing "SYNTAX ERROR" on your screen and emitting a "beep" from its speaker. This is not a problem, but you will need to type the line over again. REMEMBER: Don't ignore syntax errors--you must retype the line!

2.4. Booting a Diskette

Several programs are included for use with your ECHO speech and they can be found on the diskette labeled ECHO][SOFTWARE. To discover exactly what programs are on the diskette, you must "boot" and "CATALOG" this diskette. The process of adding DOS (Disk Operating System) commands to your Apple is called "booting" the diskette and is accomplished by following the simple steps outlined below.

1. Insert the diskette into the disk drive with the label facing upwards. The edge of the diskette with the label should enter the drive last. If you have more than one drive, place the diskette in Drive #1.

2. Turn on the power to your Apple. If it is already on, then turn it off and then on again. This clears the memory and, on most Apples, automatically boots the diskette. If the disk drive is now making a whirring noise and the red "in use" light on the drive is lit--Congratulations! The deed is done. The diskette is booted and you should skip to step 4. If, on the the other hand, your disk drive is just sitting there not making a sound, and there is an asterisk (*) in the lower lefthand corner of the screen, then proceed to step 3. Now if your disk drive is not whirring or showing any other signs of activity, and your screen does not show an asterisk (you did remember to turn on your t.v. or monitor, right?), then please check your DOS manual to make sure that your disk drive is installed properly.

3. Type:

   6 CTRL-P <RETURN>

Now it should boot. Congratulations. (If it still does not work, then check your DOS manual.)

4. If you have not yet tested ECHO speech, type:

RUN SEC DEMO <RETURN>

To end the program type:

END <RETURN>

Also, if you have not already copied the diskette do so now by placing your DOS 3.3 Master diskette in drive 1 and typing:

RUN COPYA <RETURN>

Follow the instructions with this program and when you are finished put the original diskette in a safe place and boot the copy. You should always use a copy.

2.5. The Catalog

If you have not already done so, please boot the ECHO][Software diskette. The Apple will now respond to commands concerning the diskette. Type:

CATALOG <RETURN>

A "listing" or "catalog" of program names and other information will appear. We will now show you how to interpret and use this information.

At the top of the catalog the words "DISK VOLUME 254" appear. Every diskette is assigned a volume number when it is set up to store programs. The volume does not concern us here, so we will not discuss it any further. (If you are really curious you can check your DOS manual). The next line of the catalog looks like this:

```
*A 002 HELLO
```

The word "HELLO" is the name of the first program on the diskette. Whenever this diskette is booted, HELLO is automatically run. This program simply prints a short message and a copyright notice. The number to the left of the program name (002) is a measure of the space on the diskette that is used to store this program. Disks are divided into "sectors," and each sector can hold 256 characters. The HELLO program uses 2 sectors of storage. The letter "A" to the left of the "002" indicates that the program is written in APPLESOFT(TM). APPLESOFT is a version of the BASIC programming language. We will be writing a short and simple APPLESOFT program later on in this manual, and all of our examples will assume that you are using APPLESOFT. You can tell if your Apple is ready to understand APPLESOFT by checking the "prompt" on your screen. If this character, which appears before the flashing square (the "cursor"), is a "]" then your Apple is ready for commands in APPLESOFT. If the prompt is just a flashing square that means that there is more to the catalog. Press any key you like and the catalog will continue to list, one full screen at a time, until the entire catalog has been listed. At this point, the normal prompt will appear (usually a "]" followed by a flashing square).

Instead of the letter "A" you may find any of the following letters:

I : This means that the program is written in "INTEGER BASIC" which is another of the programming languages available for the Apple.

B : This means that the information stored here is stored in "Binary " form. It may be a program; it may be data; it could even be a high resolution picture. There is no way to tell from looking at the catalog listing. You just have to trust the literature that comes with the diskette (in this case, you have got to trust us!)

T : The "T" indicates that this is a text file. A text file is characters: letters, words, etc.-- that is, "text," stored as a group (file) on the diskette.

The asterisk (*) which precedes the catalog entry indicates that this entry is "locked." "Locking" a program is a way of protecting the program from being accidentally deleted or erased from the diskette. If you did want to erase the program, you would first need to "unlock" it. This procedure is explained in more detail in your DOS manual. Now that you have some familiarity with the catalog, we will now offer a brief explanation of what each of the programs listed on your diskette can do.

2.6. Software

HELLO is automatically run when you boot this disk. It runs the HELLO.LOGO program and then pauses for about 10 seconds before exiting. You may press any key at any time during the HELLO program and it will exit immediately.

HELLO.LOGO is the program which displays the TEXTALKER LOGO.

ECHO COM is a communications aid for non-oral individuals. This program allows you to create messages to be spoken by selecting words and sentences from user created menus.

LIST BUILDER allows you to create word lists for use in the SPELLING TEST program.

LISTO is a sample word list which can be used with the SPELLING TEST program.

PHONEME TRANSLATOR translates words into their equivalent SPEAKEASY phoneme codes. (More on this in the SPEAKEASY tutorial.)

ROBOT.DEMO loads into memory and displays ROBOT.PICT. It then allows you to type in phrases which will then be spoken in a robotic voice.

ROBOT.PICT is the picture used in ROBOT.DEMO.

SEC DEMO displays SEC DEMO.PICT and gives a short demonstration of the ECHO][section of the SuperSprite.

SEC DEMO.PICT is the picture used by SEC DEMO.

SPEAKEASY is a program which allows you to use speech in your own programs by using special phoneme codes. The advantages of this program over TEXTALKER for certain applications is covered in the SPEAKEASY tutorial.

SPELLING TEST tests spelling ability by speaking words and then checking the typed response for spelling accuracy.

TALKING TYPEWRITER names each character on the Apple keyboard as the key is pressed.

The TEXTALKER programs allow you to have the Apple say a word when you type in the word. These programs utilize nearly 400 rules of pronunciation to correctly convert your typed text to speech.

Use TEXTALKER if you do not own a RAM card.

TEXTALKER.RAM is used instead of TEXTALKER if you own a RAM card because it loads the TEXTALKER program into your RAM card, thus giving you more available memory for your programs. This version of TEXTALKER also has special features for users who are blind.

TEXTALKER.BLIND contains all the same features as TEXTALKER.RAM, but does not require a RAM card. Use this program if you need the features for the blind, and you do not own a RAM card.

TEXTALKER.RAM.OBJ is the program that is loaded into the RAM card when you run TEXTALKER.RAM. This program can not be run or loaded separately of TEXTALKER.RAM.

3. Textalker Tutorial

In order for you to get the SuperSprite to speak, you must use one of the following programs: TEXTALKER(TM), TEXTALKER.RAM(TM), TEXTALKER.BLIND(TM) or SPEAKEASY(TM). Once you have run one of these programs you will not need to run it again unless you reboot a diskette or turn the computer off. Each of these programs, when run, will slightly modify your DOS (Disk Operating System). Because of this it is a good idea for you to initialize any new diskettes with an unmodified DOS, like that on your system master diskette. In the following sections we will assume that you are using the TEXTALKER program. All of the commands used in TEXTALKER are also used in TEXTALKER.RAM and TEXTALKER.BLIND. TEXTALKER.RAM and TEXTALKER.BLIND have a few additional commands designed for users who are blind. These additional commands are covered in the section titled "Features to Aid Blind Users." To run TEXTALKER type:

BRUN TEXTALKER <RETURN>

Note that you should type "BRUN" and not "RUN." If you typed "RUN" your Apple responded with a"FILE TYPE MISMATCH ERROR" because this is a binary file and not an APPLESOFT file. If you made this error you should simply retype the line. After the disk drive stops whirring you will see the following message on your screen:

```
TEXTALKER (TM)
ECHO ][ SPEECH GENERATOR
VERSION 1.3
COPYRIGHT 1981 VISEK,MAGGS,STREET & KORY
]
```

Do not worry if the version number is different. Running TEXTALKER.RAM or TEXTALKER.BLIND is accomplished in the same way (except that you would type a different program name of course). We will cover SPEAKEASY in a separate section.

You are probably ready to hear the SuperSprite speak. Type:

PRINT "HELLO" <RETURN>

You probably noticed that the SuperSprite said each letter as you typed it and, when you pressed RETURN, that it said the word "HELLO." If this was not the case, then make sure that you typed exactly what is shown above, especially the quotes. (Notice that the SuperSprite did not say anything when you typed the quotation marks. Later we will show you how to have punctuation spoken.) If you want to hear it again, then just type it again. What did we just do anyway? Well, the word "PRINT" is an APPLESOFT BASIC command. Anything in quotation marks that follows this command will be printed on the screen. The TEXTALKER program is also at work. It looks at everything printed to the screen and pronounces it to the best of its ability. Pressing the RETURN key signals the Apple that this is the end of the

command, and that it should now carry out that command--in this case it should print and speak the word "HELLO." Using this information you can now make the SuperSprite say anything that you want. For example, you could type:

PRINT "I AM ECHO" <RETURN>

Go ahead and play around a little, it will help you to get a feel for the ECHO speech on the SuperSprite and also allow your ear to become better accustomed to the SuperSprite's speech. Try CATALOGing the diskette again and listen to the SuperSprite read it to you.

3.1. Mispronunciation

Welcome back. While typing in phrases for the SuperSprite to speak, you may have come across a word or two which the SuperSprite has mispronounced. Although the TEXTALKER program uses many rules (almost 400 in fact) to guide it in correctly pronouncing your text, English is so full of exceptions to these rules, such as compound words and foreign words, that it is simply not feasible to make a perfect speech synthesizer for a microcomputer. With a few simple guidelines, you can help to bridge the gap between the SuperSprite and perfection. Our first example involves a compound word, but the solution to the problem applies to more than just this type of exceptional word. Type:

PRINT "TYPEWRITER" <RETURN>

The SuperSprite said what sounded like "Tipwriter." Now type it again and leave a space between "type" and "writer."

PRINT "TYPE WRITER" <RETURN>

Often breaking up the word is all that is needed to correct the pronunciation of an exceptional word. For more examples of this, type: "equals" (E QUALS), "create" (CRE ATE), and "program" (PRO GRAM).

Other words may need to be misspelled for them to be pronounced correctly. For instance, type:

PRINT "ROBOT" <RETURN>

Notice that the first "O" is mispronounced. We need to change the vowel sound from a short sound to a long sound. (Remember "sounding out" words phonetically when you learned to read? If you don't remember--the "short o" sound is the "o" in "rock" and the "long" sound is found in "row." The long sound of each vowel is its name.) Try typing "ROWBOT" instead of "ROBOT." By following the "o" with a "w" we have made the pronunciation of the "o" as a long "o" more consistent with the patterns of English speech. (The word "robot" is foreign.) Along the same lines we can change the short "a" sound to a long one by spelling it "ay," and the short "e" will be pronounced "long" if it is doubled

("ee"). To make a vowel go from long to short, try doubling the following consonant. Using these hints, your own common-sense knowledge of English, and experimentation (which is really the key), correct pronunciation will be made simple. Now, what if you want the ECHO to pronounce one of these exceptional words correctly and also print it on the screen spelled correctly? This is not difficult and we will show you how soon.

3.2. PRINT or ?

This is a good time to look at an easier way to type PRINT. APPLESOFT BASIC allows an abbreviation for the word "PRINT" and that is the question mark (?). Try it.

```
? "HELLO" <RETURN>
```

We will continue to use PRINT in our examples, but feel free to type the question mark instead. Also, notice that you have been hitting the RETURN key at the end of each line. We are going to stop telling you do this each time. You must take care to remember to hit RETURN after each command, however, so that the Apple knows that you are done with that line.

Another point of general interest concerns the RESET key. When you press RESET, or CTRL and RESET on newer Apples, the TEXTALKER program no longer works. The program is still there, but it can not react to what is being printed to the screen, so it just sits there. To revive TEXTALKER (or TEXTALKER.RAM or TEXTALKER.BLIND) type:

```
PR#0
```

Note that the last character is a zero, not the letter "O". Give it a try; press RESET and then type PR#0 as shown above.

3.3. Textalker Commands

The TEXTALKER program has several different modes in which it can operate. The rest of this tutorial will explain these to you. Type:

```
PRINT  "<CTRL-E>0"

        or...

?  "<CTRL-E>0"
```

Are you still remembering to press RETURN? If you are carefully following instructions, you did not type a space after control-E. You should never type a space after control-E. Now type:

```
PRINT  "HELLO"
```

The SuperSprite did not make a sound! Control-E (common notation for the result of pressing the control key and another letter at the same time) precedes all SuperSprite speech commands so, whenever you print a Control-E, the TEXTALKER program interprets the following characters as a command. In this case the character following was an "O." This command puts the SuperSprite speech into "Output only" mode (get it? "O" for "Output"). In this mode the computer will print to the screen but the SuperSprite will remain silent. Now type:

```
PRINT  "<CTRL-E>T"
```

This command puts the SuperSprite into a "Talk only" mode. In this mode the SuperSprite will speak, but nothing will be shown on the screen. Type:

```
PRINT  "HELLO"
```

To return to the first mode, in which the SuperSprite does Both printing and talking, type:

```
PRINT  "<CTRL-E>B"
```

Test this mode by printing a message on the screen.

Notice that all of the commands have been preceded by a Control-E. Now suppose that for some very practical reason (or even a totally impractical reason) you do not wish to use Control-E to indicate a TEXTALKER command. You have to ability to change this if you so desire. To change the Control-E to Control-Q type:

```
PRINT  "<CTRL-E><CTRL-Q>"
```

Now Control-Q is the new command character. To double-check the change, type the following commands:

```
PRINT  "<CTRL-E>T"

PRINT  "HELLO"
```

Notice that "HELLO" was printed on the screen, which it should not have been if we had activated the "Talk only" mode in this usual way. Now type:

```
PRINT  "<CTRL-Q>T"

PRINT  "HELLO"
```

Now the "Talk only" mode is in effect. Return to the "Both" mode by typing:

```
PRINT  "<CTRL-Q>B"
```

And let us also change the command character back to a Control-E. Type:

```
PRINT  "<CTRL-Q><CTRL-E>"
```

You may change the command character to any control character. But you should not make it Control-M (which will act just like the RETURN key), Control-U (the ->), Control-H (the <-) or Control-J (a linefeed).

3.4. Using Echo Speech Within a Program

To demonstrate how to use ECHO speech from within your own program, we will now write a short APPLESOFT BASIC program. To speed things up, first issue the "Output only" command (CONTROL-E O) and then type:

```
NEW
```

to let your Apple know you are about to type a "new" program. Now type the following lines exactly as shown (including the line numbers). End each line by pressing the RETURN key, of course.

```
10 INPUT A$
20 PRINT A$
30 GOTO 10
```

Now type:

```
LIST
```

and double check your listed program with the program in the manual. If there is a mistake you just have to retype the line containing the mistake (including the line number).

3.5. A BASIC Primer

Those of you who know BASIC should skip this section. If you do not know much BASIC--stick around. The three line program that you just typed in is a BASIC program. In BASIC, each command or instruction must be preceded by a number, which is called a line number. The Apple will perform these commands in numerical order. The numbers do not need to be in increments of one however. (For example, in your program the computer will do line 10 and then line 20--it will not sit there waiting for line 11.) Our program will therefore perform as follows... The first line (line 10) uses the BASIC command "INPUT." When the Apple encounters this command it will print a question mark and wait for the user (that's you) to type a response as an input. The program will store what you have typed in its memory and label it "A$." We will assume that you type in the word "HELLO" as an input. Line 20 contains the "PRINT" command. We have used the PRINT command before, but only as followed by a phrase within quotation marks. Since A$ is not in quotes, the computer will not print the characters "A" and "$," but will instead print the

"value" of A$, which is the input/response to line 10--the word "HELLO." Therefore, when the Apple encounters line 20, it will print "HELLO." Line 30 simply tells the computer to "GO TO" line 10 and start all over again by first asking for an input and then printing the input and then starting all over again... Let's try running the program. Type:

RUN

Enter any word you like when you see the question mark, and press RETURN. The Apple will obediently print the word that you entered and then ask you for another word. To end this limited game, type:

<CTRL-C><RETURN>

in response to the input request (the question mark). The program will stop and your Apple will tell you that you stopped it at line 10 of your program, or, in its words, "BREAK IN LINE 10." Now you can turn the speech function back on and have both that and the screen output by issuing the now familiar speech command:

PRINT "<CTRL-E>B"

Now RUN the program again. Note that you do not have to put quotation marks around your response to the input request to hear it spoken unless the phrase contains a comma. Also, keep your response down to less than six lines (your Apple manuals can explain the reason for this if you are curious). Type in a few of your favorite phrases, and also try typing the commands that you have learned. If you do type a phrase which contains a comma, the SuperSprite will pause at the comma. Also note that if you type:

<CTRL-E>THELLO <RETURN>

the TEXTALKER program will stop printing to the screen as soon as you enter the T command--even before you press RETURN.

To make our program even easier to use, we will modify it slightly. First, exit the program by typing:

<CTRL-C><RETURN>

To add the following lines to the program, simply type them in:

```
5 PRINT CHR$(5)"O"
15 PRINT CHR$(5)"B"
```

These lines will automatically assume their place in proper numerical order within your program. Now change line 30 by typing:

```
30 GOTO 5
```

The Apple will automatically replace the old line 30 with the new one. Now type:

LIST

Compare your program to the following listing. If there are any mistakes, just retype the line containing the mistake.

```
5 PRINT CHR$(5)"O"
10 INPUT A$
15 PRINT CHR$(5)"B"
20 PRINT A$
30 GOTO 5
```

Again, if you know BASIC, skip this section. You see that the new line 5 uses the now familiar PRINT command, but that following it we have "CHR$(5)." This is a BASIC command equivalent to typing Control-E. (E is the fifth letter of the alphabet, hence the "5" in parenthesis.) After this comes the letter "O" in quotes. These two parts of line 5 combine to be the equivalent of typing "Control-E O." The reason we use the CHR$ command is that when you type Control-E, nothing shows on the screen, but while using the new command we can now easily check to see if we remembered to enter Control-E in the program. Line 15 prints out the SuperSprite's "Both" command and line 30 now jumps to line 5, the new beginning of the program.

Our new program works as follows: Line 5 puts the ECHO software in "Output only" mode. This way, when you type your response to line 10, the SuperSprite will not speak, which allows you to type faster. Line 15 puts the ECHO software in "Both" mode, so that line 20 both prints and speaks your input. Line 30 starts the program over again. You may stop the program as you did before by typing a Control-C instead of an input.

You can probably now see how we would have a word spelled correctly on the screen, and yet spelled incorrectly for the SuperSprite to pronounce. First we would enter "Output only" mode and print the correctly spelled word (e.g. ROBOT). Then we would change to "Talk only" mode and print the word spelled as we would spell it for pronunciation (i.e. ROWBOT).

3.6. Rate Mode

We are now ready to cover some additional modes. First, if it is not already running, RUN the program that you typed in. Now enter a sentence at least a few words long (e.g. "RUBBER BABY BUGGY BUMPERS"), then type:

<CTRL-E>C

Now type the same sentence again. You are now in "Compressed" mode in which the speech is compressed--that is, faster. Some people find the fast mode difficult to understand at first, but only until they have become accustomed to it. Once you have more experience with ECHO speech, you may find that you are always using the compressed mode. To return to the slower mode type:

<CTRL-E>E

You are now in "Expanded" mode in which the sounds are expanded and, therefore, slower. Type in another phrase to make sure you entered the "Expanded" mode command correctly.

PRONUNCIATION MODES

Currently, you are in "Word" mode, in which the SuperSprite pronounces whatever you type as a word. You may also have it spell out everything letter by letter by typing:

<CTRL-E>L

Type a few phrases in "Letter" mode and then return to "Word" mode by typing:

<CTRL-E>W

3.7. Punctuation Modes

You may remember that, earlier in this tutorial, we promised to show you how to get the SuperSprite to say the quotation marks as you typed them in; we will do that now. The punctuation mode that you are in now pronounces only unusual punctuation (e.g. "#", "$", "%", "&", "=", "@", "+", "<", ">" and "/") and also a "." if followed immediately by another character. (For an example of this last point have the SuperSprite say "2.9%.") This mode is known as the "Some punctuation" mode. To get the SuperSprite to pronounce Most of the punctuation characters type:

<CTRL-E>M

The SuperSprite will now pronounce all punctuation characters except spaces, line feeds (Control-J) and carriage returns (Control-M or RETURN). The SuperSprite will say even these if you type:

<CTRL-E>A

This command puts you in "All punctuation" mode. To return to the standard "Some punctuation" mode, enter:

<CTRL-E>S

3.8. Pitch Modes

It is also possible to adjust the "Pitch" of the SuperSprite speech. Type:

<CTRL-E>40P

Type in a short phrase. Notice that the voice of the SuperSprite has a higher pitch now. You may vary the pitch by varying the number in the command from 1 to 63. Try some different pitches now. When you are done, return the pitch level to a middle value, or back to the original pitch level of 22. Now type "ARE YOU HAPPY?". Listen carefully as the SuperSprite's pitch changes at the end of the sentence. Since the sentence ends in a question mark, the pitch rises, just as it does in normal human speech. Compare this with "ARE YOU HAPPY" (no punctuation) and "ARE YOU HAPPY." (with a period). When the sentence ends in a period, exclamation point or a colon, the SuperSprite's pitch will drop. If the sentence ends with a question mark or semicolon the pitch will rise. Notice too that some variations in pitch exist within each word.

If you would like the SuperSprite to speak in a robotic tone, you can put it in "Flat pitch" mode by typing:

<CTRL-E>22F

You may vary the pitch by changing the number here from 1 through 63 also. Type in a few phrases and notice that when the SuperSprite speaks now, its pitch does not vary within the sentence.

3.9. Volume Modes

If you are working or playing with the SuperSprite late at night, you, or perhaps a loved one, may appreciate the SuperSprite's ability to speak at different volume levels. To set the volume, use the ECHO speech command:

<CTRL-E>5V

The number may vary from 0 through 15. You may also use this mode to create special effects such as an "echo". Try setting the volume at different levels. Notice that at the higher levels, the volume of fricatives (F,H,S,SH,TH) increases without a corresponding increase in the the volume of some of the other sounds. Also, the lower levels sound like a whisper. To adjust the volume without these effects, you may turn the volume control on the SuperSprite itself. The potentiometer is the small box near the top front (keyboard end) of the board. Turning the knob counterclockwise lowers the volume and turning it clockwise increases the volume. If you have trouble locating the volume control, please refer to the section in the Installation and Technical Reference Manual that describes the location of controls on the SuperSprite.

3.10. List Control Commands

The last two commands can best be explained without running the program that you have typed in. Remember that to stop the program you type:

<CTRL-C> <RETURN>

Now set the SuperSprite speech into the "Both" mode and type:

CATALOG

To halt the SuperSprite in mid sentence type the following without pressing <RETURN>:

<CTRL-S>

To continue the catalog press any key. Now type:

CATALOG

again and, while it is listing, type the following, again without pressing RETURN:

<CTRL-X>

Note that the catalog continues to list, but that the SuperSprite remains silent. Control-X puts the SuperSprite in "Output only" mode until the Apple requests an input. After the catalog stops listing, the Apple waits for a command (input) which puts the SuperSprite back into "Both" mode.

This about concludes our discussion of the TEXTALKER commands and operating modes. You should know that you may combine the modes in many different combinations. You can have, for example, the "Flat pitch" mode and "Letter" mode and the "All punctuation" mode all active at the same time. If you need to review these commands; see the reference card for a complete listing, along with brief explanations of the commands.

4. Speakeasy Tutorial

The SPEAKEASY program generates speech very differently from the TEXTALKER programs. In the TEXTALKER programs, you printed a word and the SuperSprite spoke the word. In SPEAKEASY you will be printing phonemes to generate speech. Phonemes are the smallest distinguishable sound units of a language. The word "speech," for example, is composed of six letters, but only four phonemes: they represent the "s" sound, the "p" sound, the "long e" sound and the "ch" sound. Since SPEAKEASY works at the phoneme level, you will be able to have words pronounced exactly as you wish. SPEAKEASY was developed to be used within your programs, to give them new animation and uses. Also, if you do not have a RAM card or can not spare the 8K of memory that the TEXTALKER programs require, SPEAKEASY will give you speech capability while using only 3K of memory.

There are a couple of ways to run SPEAKEASY. The TEXTALKER programs contain SPEAKEASY, so if you have already put TEXTALKER into memory, then SPEAKEASY is also already in memory. If you want to just use SPEAKEASY, and we will assume in our examples that this is the case, then just "boot" the diskette (this procedure is explained in detail in the section called "Booting a Diskette") and type:

BRUN SPEAKEASY

Using the SPEAKEASY (TM) program rather than a TEXTALKER program insures that SPEAKEASY will work even if you press RESET or issue a "PR#" command. Your monitor or television screen should look like this:

```
SPEAKEASY
ECHO II PHONEME GENERATOR
VERSION 1.1

COPYRIGHT 1981 STREET ELECTRONICS CORP.
]
```

Don't worry if your version number is different.

4.1. Encoding Phonemes

To use SPEAKEASY you will need to have the reference card handy. There you will find a table which shows all of the phonemes and associated SPEAKEASY codes that you will be using to create speech. The first column contains an example of a common word containing a clear instance of a sound. The second column shows the common dictionary symbol for the sound, and the third shows the symbol that you will actually be typing into the computer to represent that sound. Using this table we can see that the word "speech" (to continue with our previous example) would be written in SPEAKEASY code as SP&C. The word "echo" would be EKO.

To actually hear the SuperSprite speak these words, type in the following short program:

```
20 V$=CHR$(22)
30 PRINT V$"EKO"
40 PRINT V$"SP&C"
50 PRINT "DONE"
```

Now type:

```
LIST
```

Double check your listing of the program you just typed in against the listing in this manual. As always, if there are any mistakes retype the line and list again. The way this program will work is that line 20 will make "V$" equal to a Control-V ("V" being the 22nd letter of the alphabet). This is done because every word that you want to have spoken by SPEAKEASY will have to be preceded by a Control-V. Line 30 will print the "value" of V$ (Control-V), thus alerting the SPEAKEASY program that it must speak the phoneme codes which follow it (E,K, and O). Line 40 will similarly say "S," "P," "&" and "C." Line 50 will print "DONE" without speaking because there is no Control-V preceding it. Try the program now by typing:

```
RUN
```

It is useful to note that SPEAKEASY will pronounce everything following a Control-V until it encounters a "carriage return." In BASIC a carriage return is automatically generated at the end of each PRINT command unless it is followed by a semi-colon (;). To better understand and use this feature, retype lines 30 and 40 like this:

```
30 PRINT V$"EKO";
40 PRINT "SP&C"
```

Type "LIST" to review the program and double check your changes and then "RUN" the program again. This time the SuperSprite said both "EKO" and "SP&C" even though, on line 40, "SP&C" was not preceded by a Control-V (V$). This is because line 30 ended with a semi-colon which cancelled the carriage return which would have told SPEAKEASY to stop speaking.

4.2. The Phoneme Editor

SPEAKEASY also includes a phoneme editor which makes experimentation with putting words into their phoneme codes easier. To enter the editor simply type:

```
&  RETURN
```

You will now see a quotation mark for a prompt. Now type:

```
EKO
```

The SuperSprite will say "EKO." Note that a Control-V was not necessary. You can continue to type in words and hear them pronounced in this manner, making changes using the Apple's normal editing commands. You may also type in more than one encoded word at a time. Try typing "I am ECHO" which encodes as:

!AMEKO

When you are finished experimenting, you can quit the editor by pressing RETURN instead of entering any phoneme codes. Use this editor to type in the words for the rest of the examples presented.

4.3. Inflection and Stress

Type:

EKO

Listen carefully. Now type:

E3KO

Note that this time the "E" is more stressed. The number 0,1,2 or 3 following a vowel, diphthong or "r-colored vowel" (you will see these categories on your reference card) determines the stress of that phoneme. If no stress number is given then a value of 2 is used. Using a 3 gives greater stress by making the phoneme higher in pitch, longer in duration, or both. Try retyping "E3KO" and substituting 2,1 and 0 for the 3. The 1 reduces the stress placed on the phoneme by making it lower in pitch and shorter. A stress of 0 reduces the sound to what is termed a "schwa." The schwa also has its own code (') which can be used.

4.4. Pitch and Rate

The base pitch of the SuperSprite is set using the same commands that we used in TEXTALKER (i.e. Control-E40P, Control-E40F...). These are outlined in more detail in the TEXTALKER tutorial. You may also set the rate of speech (speed) using TEXTALKER commands (i.e. Control-E C and Control-E E).These commands can not be issued from within the phoneme editor, however. (If you have already tried it, press Control-RESET.) To issue these commands, exit the editor by pressing RETURN instead of entering any phonemes. Now enter the command as you would in TEXTALKER, for example:

PRINT <CTRL-E>40P"

Now re-enter the editor by typing:

& <RETURN>

SPEAKEASY allows you to adjust the pitch within a word too. Simply insert a number from 1 through 9 anywhere within a word except following a vowel or a diphthong (where it would be considered a stress number). The base pitch is 5 and 9 is the highest pitch. The pitch will remain at the new level until it is reset, or until the first "stop consonant" or "fricative" is encountered. Try typing:

9!3 7AM 5AN 3E3K 103

The underlined numbers set the pitch level, while the numbers following the vowels set the stress level as explained earlier. This may seem a little confusing at first, so try experimenting with different combinations and familiarize yourself with the system.

Another way to modify the pitch is by using the symbols ">," "<" and "=." The ">" symbol will set up a pattern of rising pitch, the "<" symbol sets up a pattern of falling pitch, and flat pitch is set up using the "=" symbol. All sounds will continue to vary in the pitch pattern that you have set until either a fricative, stop consonant, or a pitch level change is encountered. Try typing the word "help" using these symbols. You might type:

H>EEEELLLLP

or...

H<EEEEEEEEEEEE>LLLLLLLLLLLLP <RETURN>

4.5. Volume Control

SPEAKEASY allows you to adjust the volume of speech within your programs by using "+" and "-." Typing a "+" increases the volume and the "-" decreases it. You may also type a series of "+"s or "-"s, just as you did with the pitch symbols. For example type:

EKO--EKO--EKO--EKO++++++EKO <RETURN>

The volume will remain modified until it is explicitly changed again using the "+" or "-" commands. When TEXTALKER or SPEAKEASY is first RUN, the volume is already set at about the maximum level. Increasing the volume one level above this will increase the volume of fricatives without a corresponding increase in the voiced sounds.

4.6. Pauses

The SPEAKEASY program ignores spaces between words although we recommend that you type them for better legibility. To make SPEAKEASY pause, use a comma. The comma may be followed by a number to vary the length of the pause. A 1 is a short pause while a 9 gives a long pause. Type in this demonstration:

EKOEKO,1EKO,3EKO,5EKO,7EKO,9EKOEKO <RETURN>

If no number is used, a value of 2 is assumed. Try:

EKO,EKO,2EKO <RETURN>

The pauses between the words are identical.

SPEAKEASY is really not difficult to use, though, due to its high degree of flexibility, you probably feel that there is an awful lot to remember. Practice by using the sample vocabulary in Appendix A and also a dictionary, which will help you encode words when used in conjunction with your reference card. There is also a program on the diskette called "PHONEME TRANSLATOR". This program will show the phonemes that TEXTALKER generates when it translates text to speech.

5. Features to Aid Blind Users

Extra commands designed to help the blind user are contained in both TEXTALKER.RAM and TEXTALKER.BLIND. You will want to use TEXTALKER.RAM if you own a RAM card; otherwise use TEXTALKER.BLIND.
Type:

RUN TEXTALKER.RAM

or

BRUN TEXTALKER.BLIND

After the disk stops spinning, the following message will be displayed on the screen:

TEXTALKER (TM)
ECHO II SPEECH GENERATOR
VERSION 1.3

COPYRIGHT 1981 VISEK,MAGGS,STREET & KORY

Type in the word "PRINT", but do not press RETURN. Now press the left arrow key (<-) a few times. Notice that as you back up the cursor, the SuperSprite will pronounce the letters that the arrow is passing over. Now press the right arrow key (->) a few times. It also speaks. The right arrow will do this in all versions of TEXTALKER, but the left arrow speaks only in TEXTALKER.RAM and TEXTALKER.BLIND. Only the characters normally pronounced in the punctuation mode currently set will be spoken.

5.1. MODES

LINE REVIEW MODE: provides a controllable audio cursor. You may move this invisible cursor around the screen to review what is currently on the screen. To explore this, first type:

HOME

This is an APPLESOFT command which clears the screen. Now type:

CATALOG

Since TEXTALKER begins in the "Both" mode, the catalog will be both spoken and printed to the screen.

To review the catalog which is now on the screen you must first type in the Line Review command:

CTRL-L

The SuperSprite responded to this command by saying "review." It is now expecting Line Review commands. When you first enter Line Review mode you may enter any of the following commands (but do not type anything yet).

Any letter from A through X: Typing one of these letters will cause the audio cursor to be positioned at the beginning of the line which numerically corresponds to the letter selected. Pressing the letter "A" will position the cursor at the beginning of the first line. Pressing "B" will place it at the beginning of the second line, and so on to the twenty-fourth line (X).

Z: Pressing Z will place the audio cursor at the same vertical and horizontal position as the normal video cursor.

SPACE BAR: Pressing the Space Bar will cause the SuperSprite to tell you the current vertical and horizontal position of the cursor. It will then exit line review mode.

A Control Character: Typing a Control character will change the command used to enter Line Review mode. This procedure is identical to that used to change the Control-E command in TEXTALKER.

Now press the Space Bar. The SuperSprite responded by saying a letter, corresponding to the vertical position of the cursor, and then two numbers, which correspond to the horizontal position of the cursor. The leftmost position is zero. To reenter the Line Review mode type:

CTRL-L

and then...

D

The SuperSprite confirmed your choice by saying "Line D." There are now several new commands available to you. You may find it helpful to use your reference card as we explain them to you.

Press the right arrow twice:

-> ->

Please remember that commands in Line Review mode should <u>not</u> be followed by pressing <RETURN>.

The SuperSprite just said "disk volume." Now press the left arrow once:

<-

The SuperSprite said "volume." (The last word it had spoken.) Now type:

T

and

-> -> ->

The SuperSprite said the letters "v," "o" and "1." What did we just do? When you first entered Line Review mode you were also in Word mode; that is, pressing the arrow keys moved the cursor right or left by one word at a time. Pressing the letter "T" allows you to Toggle between (switch between) Word mode and Letter mode. In Letter mode the cursor moves letter by letter. Press "T" again. You are now back in Word mode. Press the arrow keys and check if you are not convinced. Now return the cursor to the beginning of the line, using the <- key, and press:

<RETURN>

The SuperSprite said "Disk Volume 254." This is because pressing the <RETURN> key causes the SuperSprite to speak the entire line, starting from its current position. When it finishes speaking the line, the cursor will be back at the leftmost position of the same line.

To move the cursor from line to line type:

;

and you will move up one line. To move down one line type:

/

When moving up and down the cursor will always move to the beginning of the new line. Try moving to line F (the fifth line) and experiment with these new commands on the first catalog entry. Now move back to line C and type a comma:

The SuperSprite will say "to" and wait for you to indicate up to what line you would like it to read. You will respond with a letter corresponding to a line number greater than (further along in the text than) the current line. For example press:

G

The SuperSprite will now read lines C through G to you.

Finally, pressing:

<ESC>

will exit Line Review Mode, leaving you exactly where you were before you entered it. Since Line Review mode exits with all APPLESOFT variables and pointers intact, you may enter Line Review mode any time a program prompts you for a answer. Simply type:

CTRL-L

then review the screen using the appropriate commands and exit Review mode before answering the question.

6. Technical or Advanced Information

The speech portion of the SuperSprite is designed around the Echo][circuitry, and utilizes the Texas Instruments TMS 5220 speech processor. This circuit is an upgraded version of the one used in the Speak & Spell (trademark of Texas Instruments) which models the human vocal tract using LPC (Linear Predictive Coding). Instead of storing the actual speech signal, only those parameters needed to describe each speech sound are stored. This accounts for the compactness of the TEXTALKER system, which takes up no more memory than a HIRES page.

TEXTALKER uses nearly 400 rules of pronunciation and many common exceptions to these rules to analyze a word. After analyzing the word, TEXTALKER generates the standard SPEAKEASY phoneme and pitch codes. TEXTALKER then sends these codes to SPEAKEASY which actually pronounces the word. TEXTALKER is ,therefore, just a simpler way for the user to access SPEAKEASY. You may see the actual codes that TEXTALKER generates by running the PHONEME TRANSLATOR program that you will find on your Echo][Software disk.

6.1. Textalker and PR#0

If you type PR#0 when TEXTALKER is installed, TEXTALKER actually does the equivalent of a PR#0 and an IN#0. Therefore, any device previously activated by a IN#n will be disconected.

7. Using Textalker and Speakeasy from Machine Language

To use TEXTALKER when programming in machine language simply do a JSR to "COUT" ($FDED) putting the character to be sent into the accumulator. For example, the following program will result in the SuperSprite saying "HI."

```
A9 C8       LDA #$C8    ;"H"
20 ED FD    JSR $FDED   ;COUT
A9 C9       LDA #$C9    ;"I"
20 ED FD    JSR $FDED   ;COUT
A9 8D       LDA #$8D    ;[RETURN]
20 ED FD    JSR $FDED   ;COUT
60          RTS         ;DONE
```

SPEAKEASY is accessed similarly except that that all phoneme strings must begin with a Control-V ($96) and end with a RETURN ($8D). For example, to say "HI" you would write the following routine:

```
A9 96       LDA #$96    ;CTRL-V
20 ED FD    JSR $FDED   ;COUT
A9 C8       LDA #$C8    ;"H"
20 ED FD    JSR $FDED
A9 A1       LDA #$A1    ;"!"
20 ED FD    JSR $FDED
A9 B3       LDA #$B3    ;"3"
20 ED FD    JSR $FDED
A9 8D       LDA #$8D    ;[RETURN]
20 ED FD    JSR $FDED
60          RTS         ;DONE
```

8. Speakeasy Phoneme Categories

The types of sounds represented by the phoneme codes are divided into six categories. A brief description of each category follows. (It is NOT necessary that you understand these definitions for you to be able to use SPEAKEASY. These brief explanations are provided for your personal enrichment.)

VOWELS: In pronouncing vowels the air flows uninterrupted though the vocal cords and lips. (The English vowels are a,e,i,o and u.)

DIPHTHONGS: These are the combination of two vowel sounds within one syllable. (Commonly there is only one vowel per syllable.) To get a feel for this phenomenon, say the word "cake" very slowly and listen to how the "long a" sound moves into a "y" sound before the final "k."

"R" COLORED VOWELS: The "r" following these vowels modifies the sound of the vowel.

VOICED CONSONANTS and UNVOICED FRICATIVES: Each of the voiced consonants is created by vibrating your vocal cords. Place your fingertips against your adam's apple and say the voiced consonant "zzzzzzzzzz." The sensation you feel with your fingertips is the vibration of the vocal cords. Contrast this with with the lack of sensation which should result when you say the unvoiced fricative "sssssssssss." The "s" sound is not produced by vibrating the vocal cords, but through friction, produced as air flows over various parts of the vocal tract.

STOP CONSONANTS: The flow of air is completely cut off before we pronounce a stop consonant. Say the word "tall." Because the "l" is not a stop consonant, we flow smoothly from the "a" sound to the "l" sound. Now say "tab." "B" is a stop consonant and you will find that you actually pause, very briefly, before you pronounce the "b."

9. Using Echo][Speech with Commercial Programs

You may use TEXTALKER to make some programs speak which were not explicitly designed to run with a speech synthesizer. You would do this by running a TEXTALKER program before running your program. However, there are limitations:

(1) You must be able to run your program without booting another diskette. The reason for this is that the TEXTALKER programs modify DOS when they are first run. If you boot another diskette you replace the modified DOS with a new DOS.

(2) For similar reasons, your program must not modify DOS when it runs.

(3) If your program resets memory pointers (i.e. HIMEM, LOMEM), or requires more than 30K of memory, you will need to use TEXTALKER.RAM.

10. Appendix A - Sample Phoneme Vocabulary

```
A - @
AND - AND
ANSWER - A3NS'R
APPLE - A3P'L

B - B&
BYTE - B!3T

C - S&
CATALOG - KA3DIL*G
CORRECT - KORE3KT

D - D&
DECIMAL - DE3SIM'L
DIVIDE - DI1V!3D

E - &
EQUALS - &3KW'LS
EXCLAMATION -
       EKSKL'M@3SHUN

F - EF
FIRST - F'RST

G - J&
GOOD - GQ3D

H - 'C
HELLO - HELO1

I - !3
INCORRECT - INKORE3KT

J - J@

K - K@
KEYBOARD - K&3BORD
L - EL

M - EM
MEMORY - MEM'R&1
MULTIPLIED - MULTIPL!1D

N - EN
NO - NO
NUMBER - NUMB'R

O - O3
OFF - *F
ON - *N
OPEN - OP'N

P - P&
PROGRAM - PRO3GRAM

Q - K%3
QUESTION - KWESC'N

R - ;R3
RETURN - R&T'R3N

S - ES
SORRY - S;R3&
SPELL - SPEL

T - T&
THAT - (AT
THE (&3
THOUSAND - )#3ZS'ND

U - %3
UNDERSTAND -
      UND'RSTA3ND
V - V&

W - DUBI%1
WHERE - W@R3
WRONG - R*/

X - EKS
Y - W!3
YES - YES

Z - SZ&
```

11. Appendix B - Telecommunications and the SuperSprite

The SuperSprite may be used in conjunction with a modem with "automatic answer" capability, to enable you to receive information over the telephone. It can even be used this way as a telephone answering machine. We are including a listing of a sample program which uses the SuperSprite and an Apple-Cat (TM of Novation Inc.) modem with the Touch Tone (TM of the Bell System) decoding option.

This program allows you to telephone your Apple computer and to check and modify an itemized inventory. By following the instructions included with the Apple-Cat modem, you could also develop the capability of storing voice messages on a tape recorder like an answering machine.

Our sample program does the following things. First, when you call your Apple, the modem will answer the phone. The SuperSprite will then ask you for a password, and then an inventory item number. You respond to these prompts by simply typing in the numbers on any Touch Tone telephone. The modem will decode the Touch Tones and make them available to the program, which will send them to the SuperSprite to be spoken.

The program has quite a few remark statements, which should help you follow and better understand its logic.

You connect the SuperSprite to the Apple-Cat as follows:

1. Purchase a cable which has a mini-jack on one end, and stripped wires on the other end. Radio Shack sells one as part number 42-2434.

2. Plug the mini-jack into the SuperSprite.

3. Connect the stripped ends of the cable to the Apple-Cat so that the wire which corresponds to the tip of the cable (positive) is connected to pin 24 of the interface pins on the Apple-Cat. The other wire (negative) must be connected to pin 25. If you are unable to purchase a connector to fit on these pins, you will have to solder the connection. Soldering may, however, void your Apple-Cat warranty--check with Novation.

```
0     REM  SUPERSPRITE AND APPLE CAT EXAMPLE
4     SL = 2: REM  SLOT NUMBER OF CAT
5     DIM C(20),N(20)
7     D$ =  CHR$ (4)
8     PW$ = "321": REM  THIS IS YOUR PASSWORD
10    REM  CAT INVENTORY
20    GOSUB 1000
30    POKE 2040 + SL,0: REM  FORCE INITIALIZATION
```

```
40    PRINT D$"PR#"SL: PRINT  CHR$ (16): PRINT D$"PR#0": REM
      ENTER PHONE MODE (16= CTRL P)
50    PRINT D$"IN#"SL: REM  WAIT FOR A PHONE CALL
60    GET A$: IF A$ =  CHR$ (13) THEN 100: REM   WE GOT A PHONE
      CALL
70    IF A$ = "Q" THEN  END : REM  IF A Q WAS TYPED AT THE
      KEYBOARD THEN QUIT
80    GOTO 60: REM  IF NOT Q THEN TRY AGAIN
100   REM  ANSWER PHONE
105   P$ = ""
115   PRINT ",,,,,,,,,,": REM   DELAY
120   PRINT "HELLO, PLEASE TYPE IN YOUR PASSWORD"
140   GET A$: IF A$ =  CHR$ (13) THEN 900: REM   TIME OUT SO HANG
      UP
150   IF A$ = "#" THEN  PRINT : GOTO 170
160   P$ = P$ + A$: GOTO 140
170   IF P$ <  > PW$ THEN 900: REM  WRONG PASSWORD SO HANG UP
180   PRINT
190   PRINT "PLEASE ENTER PART NUMBER"
195   C$ = ""
210   GET A$: IF A$ =  CHR$ (13) THEN 900: REM   TIMEOUT SO HANGUP
220   IF A$ = "#" THEN 300: REM  END OF PART NUMBER
230   IF A$ = "*" THEN 180: REM  '*' MEANS START OVER
240   C$ = C$ + A$
250   PRINT A$: GOTO 210
300   REM  SEE IF VALID PART NUMBER
305   PRINT
310   C =  VAL (C$)
320   FOR A = 1 TO X: IF C(A) = C THEN 340
325   NEXT A
330   PRINT "BAD PART NUMBER": GOTO 900: REM   IF BAD PART THEN
      HANG UP
340   PRINT "THE CURRENT INVENTORY OF PART NUMBER "C",IS
      "N(A)",ITEMS"
350   PRINT ",,,,": REM  PAUSE
360   PRINT "ENTER A ONE IF YOU WOULD LIKE TO REMOVE ITEMS FROM
      INVENTORY"
370   PRINT ",,,"
380   PRINT "ENTER A TWO TO CHECK THE INVENTORY OF ANOTHER PART"
390   PRINT ",,,"
400   PRINT "ENTER A STAR IF YOU ARE DONE"
420   GET A$: IF A$ =  CHR$ (13) THEN 500
430   IF A$ = "*" THEN 900
440   IF A$ = "2" THEN 180
450   IF A$ <  > "1" THEN 500
462   K$ = ""
465   PRINT "PLEASE ENTER AMOUNT TO REMOVE"
470   GET A$: IF A$ =  CHR$ (13) THEN 500
475   IF A$ = "#" THEN 500
480   IF A$ = "*" THEN 460
490   PRINT A$
495   K$ = K$ + A$: GOTO 470
500   REM  REMOVE ITEMS FROM INVENTORY
510   N(A) = N(A) -  VAL (K$)
530   GOTO 340
```

```
900  REM  HANG UP
905  PRINT "GOOD BYE"
910  PRINT D$"PR#"SL: PRINT  CHR$ (26): PRINT D$"PR#0"
920  GOTO 40
1000 REM  INIT ARRAY
1010 X = 1
1020 READ C(X),N(X)
1030 IF C(X) = 9999 THEN  RETURN
1040 X = X + 1: GOTO 1020
2000 DATA  1234,500
2010 DATA  321,250
2020 DATA  555,365
7060 GET A$: IF A$ =  CHR$ (13) THEN 100
9999 DATA  9999,9999
```

12. Memory Usage

	INITIAL	INSTALLED	HIMEM
TEXTALKER	$7400-$75FF	$7600-$95FF	30208
TEXTALKER.BLIND	$7100-$72FF	$7300-$95FF	29440
TEXTALKER.RAM	$9300-$950F	$D800-$FFFF	-----
SPEAKEASY	$8900-$89FF	$8900-$95FF	35328

TEXTALKER COMMANDS

OUTPUT MODES	
PRINT ONLY	CTRL-EO
TALK ONLY	CTRL-ET
*BOTH	CTRL-EB
PITCH (n=1 to 63) *22	
FLAT	CTRL-EnF
*INTONATION	CTRL-EnP
VOLUME (n=0 to 15) *12	
	CTRL-EnV
RATE	
COMPRESSED (FAST)	CTRL-EC
*EXPANDED (SLOW)	CTRL-EE
PUNCTUATION SPOKEN	
*SOME (#$%&+/<=>@)	CTRL-ES
MOST	CTRL-EM
ALL (LF,CR,SPACE)	CTRL-EA
PRONUNCIATION	
*WORDS	CTRL-EW
LETTERS	CTRL-EL
CHANGING CTRL-E	
	CTRL-E CTRL-n

where n is any letter except H,J,M, or U.

*modes set automatically when TEXTALKER is first run.

SPEAKEASY COMMANDS

All phoneme words must be preceeded by a CTRL-V.
PITCH, RATE and VOLUME commands are the same as in TEXTALKER.

SPEAKEASY PHONEME CODES

VOWELS

cat	a	A
lot	o	;
caught	o	*
let	e	E
see	e	&
hid	i	I
book	oo	Q
but	u	U
due	oo	:
about		'

DIPTHONGS

cake	a	@
tie	i	!
toe	o	O
pound	ou	#
toil	oi	?
you	u	%

"R" COLORED VOWELS

car	ar	;R
chair	ar,er	@R
her	ur,er	'R
hear	er	&R
fire	ir	!R
for	or	OR
tour	oor	QR
hour	our	#R

VOICED CONSONANTS

let	l	L
many	m	M
no	n	N
sing		/
red	r	R
this		(
very	v	V
wet	w	W
yes	y	Y
zero	z	Z
azure	zh	X

STOP CONSONANTS

bat	b	B
dog	d	D
get	g	G
kick	k	K
pet	p	P
tie	t	T
check	ch	C
job	j	J

UNVOICED FRICATIVES

fit	f	F
hat	h	H
see	s	S
she	sh	$
think	th	)

INFLECTION

STRESSED - 3 NORMAL - 2 REDUCED - 1 SCHWA - 0

PITCH

RISING - > FLAT - = FALLING - < PAUSE - ,

13. Blind User Commands

These commands are available when using TEXTALKER.BLIND and TEXTALKER.RAM.

CTRL-L enters Line Review mode. One of the following commands must immediately follow:

A thru X	: Selects line to review. A is the top line.
Z	: Selects current line and horizontal position.
<SPACE>	: Speaks the current position on screen and then exits.
CTRL-n	: n may be any letter except H,J,M or U. Changes command to enter Review mode.

Once in Review mode these commands are available:

<SPACE>	: Same as above except does not exit.
T	: Toggles between Word and Letter mode.
<-	: Left arrow moves the audio cursor to the left.
->	: Right arrow moves the audio cursor to the right.
;	: Moves audio cursor to the beginning of line above current line.
/	: Moves audio cursor to the beginning of line below current line.
,	: Echo will say "to" and wait for a letter. The Echo will then read all lines from the current line to the line corresponding to the letter entered.
<RETURN>	: Echo will read entire line starting at current position.
<ESC>	: Exits Line Review mode.

THIS PAGE LEFT INTENTIONALLY BLANK

ECHO WORDS

for the

SuperSprite

Operating Instructions

Voice of the Echo: Julie Edwards
Speech Editing: Mark Pfeffer
Programming: Mike Kory
Documentation: Mike and Fern Kory

THIS PAGE LEFT INTENTIONALLY BLANK

Table of Contents

1. Introduction

Street Electronics and Synetix are proud to present Echo Words, our unique female vocabulary diskette. We think that you will find Echo Words to be a valuable, entertaining and flexible enhancement to your SuperSprite.

In this introduction we will explain what Echo Words is, who it is intended for, and why the speech that it allows you to generate sounds more natural than the speech generated by the TEXTALKER program.

First of all, don't be frightened by the size of this manual. Echo Words is really very easy to use. The reason for the size of this manual is that we've included a lot of information for advanced (and/or curious) programmers.

Echo Words is a dictionary of 719 words and phrases in a female voice to be used with the Echo][speech synthesis portion of the SuperSprite. You actually can use more than 719 words since most of the words can have an "S" sound added to them to make them plural. Also, homonyms (words that sound the same) such as "TWO", "TO", "TOO", and "2" are only represented once in the list.

While most people can use the examples in the manual to create a sentence, Echo Words is meant to be used by people who are able to program in Applesoft BASIC. There is also enough information presented so that assembly language programmers can write their own drivers for Echo Words. (The next few paragraphs are only for the curious and do not contain information necessary for using Echo Words. Feel free to skip to the next section of the manual if you wish.)

You will better understand why the speech generated by Echo Words sounds more natural than that of TEXTALKER if we first explain how the Echo portion of the SuperSprite works.

The SuperSprite uses the Texas Instruments TMS5220 speech synthesizer chip. This chip is different from most of the other chips on the market in that it does not have a fixed set of word sounds built into it.

Chips which have a fixed set of word sounds (phonemes) are limited not only to those phonemes, but also to the male voice in which they were originally encoded. Simply put, the 5220 chip is a mathematical model of the human vocal tract which will respond with human sounding speech when sent the right variables.

The mathematical model is based on what is called Linear Predictive Coding, so we call the variables that are sent to the chip LPC data. The LPC data is created by Street Electronics using a sophisticated computer system.

First, a tape recording is made of a human speaker. Then this tape is played into the computer, which analyzes it, and produces a preliminary set of LPC data. We then hand tailor this data to produce the best sounding speech possible.

For the TEXTALKER and SPEAKEASY programs, individual word sounds (phonemes) were analyzed and the LPC data for each sound stored in these programs.

When you use TEXTALKER or SPEAKEASY, these phonemes are combined to make a word. This method gives you an unlimited vocabulary, but not the highest quality speech. This is because a fixed set of phoneme sounds are used, and these can not reflect the more complex interactions of sounds within each of the hundreds of thousands of words in the English language. Therefore, for Echo Words, we have encoded entire, individual words (not phonemes) into LPC data. This captures the fine variances of pitch and rhythm within each word and makes the speech more natural sounding.

2. Overview

This section will give you a brief overview of the steps involved in using Echo Words. After this, each step will be covered in more detail and we'll include some examples for you to type in to your computer.

The first step is to decide which words you will want to use in your Applesoft program. Appendix A lists the 719 words and phrases available.

Next, you boot the Echo Words disk and run the editor. You use the editor to create your list of words. You may then print your list to a printer and also save the list to disk for future reference.

The editor will then create a customized machine language speech program. There are three different versions of the speech program: A version which loads mostly into a RAM card (or the extra 12K bank on the Apple //e), the standard long version, and a very short version which lacks the speech commands available with the first two programs. These programs can be located anywhere in memory. A customized speech program, containing only the words you have specified, can then be saved to your own diskette.

Once you've created your customized speech program you are ready to have your Applesoft program speak. To use the speech program, "BRUN" it similar to the way you do for TEXTALKER, Then you may use "labels", the actual text of the word, to call the words to be spoken. For example, assuming we chose these words to be on our list, we could say "I AM AN APPLE" by typing:

```
& "I","AM","AN","APPLE"
```

either in a program or in "immediate" mode. There are also commands to vary the pitch, volume, and delay between words. You can even add an "S" sound to any word or phrase.

Before moving on to the next section, we highly recommend that you boot the Echo Words diskette and run the demo program. Then, initialize a blank disk (or use one that has some free space on it) and boot the Echo Words diskette again, this time running the editor. Together we will now create a sample speech program as each of the steps is explained in more detail.

3. Selecting Words from the List

There are a few things you should know about how the list of words is arranged. All the words are in alphabetical order reading from left to right and the digits 0 through 9 appear at the beginning of the list. The number which follows each word is only for those concerned about memory usage. This value is the number of bytes that this particular word requires.

You may notice that there are very few words which are plurals. This is because you can append an "S" sound to any word using the speech program itself.

Probably the most important note about the list is that homonyms (words which sound the same but have different meanings) appear only once in the list.

Usually the homonym which occurs first alphabetically is listed. For example, the number "2" is in the list and should be used for the homonyms "TO", "TOO", and "TWO". "ARE" should be used for the letter "R" ("ARE" comes first alphabetically) and "U" should be used for "YOU" ("U" is first alphabetically). So, before assuming a word is not in the list, check all its homonyms. Also, you will notice that the words "PREESENT" and "PRESENT" are in the list. "PREESENT" represents the word "present" as in "we are proud to present the SuperSprite", while "PRESENT" represents the word "present" as in "past, present, and future".

4. Using the Editor

When you enter the editor you will see all the valid editor commands listed along the top of the screen. The commands are enclosed in brackets to indicate that these are control characters (i.e. that you press the CONTROL key and the character to enter the command). The inverse bar is your cursor, and the inverse number in the upper right corner keeps track of the number of words you've entered into the list. Please note that all entries must be in upper case--if you have an Apple //e, make sure the "caps lock" key is depressed.

To get a feel for the editor, carefully type in the following words:

I

LIKE

THE

APPLE

Press RETURN after each entry. You may use the right and left arrow keys to edit the line as you do in Applesoft. These words all come from the alphabetical list of the over 700 words and phrases available to you. Most entries consist of a single word but, as in the case of "SELECT ONE OF THE FOLLOWING", and "WHAT WAS THAT", some longer phrases are available. Now type a word which is not in the Echo Words vocabulary, such as "FERN".

The editor will respond with the message "'FERN' not available". As you see, you may type words in the editor to see if they are available rather than look through the list.

Now let's try all the commands.

5. Moving the Cursor

You may move the cursor up and down by typing control-K and control-J respectively. If you have an Apple //e you may also use the up and down arrow keys. To replace a word, move the cursor onto a word and type over it.

6. [D] Delete

Control-D will remove the word beneath the cursor. Let's say you've changed your mind and don't want to use the word "LIKE". Move the cursor up, over the word "LIKE", and type control-D. The screen will scroll up and fill the space that "LIKE" had occupied in the list.

7. [I] Insert

Control-I will shift down all the words from the cursor to the end of the list, leaving you with a single blank line.

You can add a word to your list here. If you decide not to insert a word, simply hit the RETURN key.

If you are not going to use "labels" to call the words in your program, it is often useful to have the words in a selected order. This is why the insert command is available. To practice this command, suppose that you've decided you want the word "LIKE" after all, and you want it before "THE". Move the cursor over the word "THE" and type control-I. (If you have an Apple //e, you may hit the TAB key instead of control-I in <u>any</u> program.) Now type "LIKE", and press RETURN. Notice that if you make an error in typing "LIKE", you will have to type control-I again.

8. [B] Beginning and [E] End

If you are working with long lists of words, these commands can be very useful. Control-B will move the cursor to the beginning of your list. Control-E will move the cursor to the end of your list.

9. [F] Find

The FIND command will search through the words you have typed and place the cursor on top of the word you are looking for. This can be helpful if you have a long list of words. For example, try typing control-F. The editor will respond with "ENTER WORD OR PHRASE TO FIND". Type "THE". The cursor is now placed on "THE". If the editor is unable to find the word you've requested, it will tell you so, and leave the cursor position unchanged.

10. [N] New

The NEW command will erase the entire list, allowing you to start a new list. After typing control-N, the editor will respond with "CLEAR MEMORY?". You would press "Y" (yes) if you did indeed want to start a new list. Pressing any other key will leave things as they were.

11. [W] Write

Control-W allows you to print (write) your list to a printer.

After typing control-W, the first question you will be asked is "WOULD YOU LIKE THE LIST PRINTED WITH NUMBERS?". If you answer "Y", the list will be numbered, starting with the number zero. In your Applesoft program, you may use these numbers instead of using labels to access the words in your list (more about using labels later).

If you hit RETURN in response to this question, you will exit the "write" mode.

The editor will then ask which slot your printer card is in. Again, if you just press RETURN, you will exit this mode.

Next, the editor will ask you for printer initialization commands. You should enter whatever control characters or commands your printer requires. Press RETURN twice after you've entered this information.

You will then be asked if this is correct. If you made an error, type "N" (no), and you will be allowed to re-enter the printer initialization information. Note that the arrow keys will not work in response to this question. This is to allow you enter anything you need to initialize your printer. If you make an error, hit RETURN twice and start over.

Finally, you will be asked for the name you want to have printed at the top of the list. Press RETURN if you don't want a title. Now the list will be printed to your printer.

12. [S] Save and [L] Load

If you've typed a long list of words and think you may wish to change it or add to it in the future, save your list of words to a disk. Note that these commands save and load the list of words and not a speech program.

After typing [S] or [L], you will be prompted for a file name. You may, at this point, type a "/" if you want to CATALOG the last accessed drive or you may also type "/1" or "/2" to CATALOG driver 1 or 2 respectively. You may use the DOS Drive, Slot, and Volume options in your filename.

If you just press RETURN, you will exit this mode. Note that the list will be saved with the suffix ".LIST". You should not type this as part of the name, it will automatically be added to the filename when you save and load the lists.

13. [X] Display Memory Usage

Control-X will present a display explaining the memory that would be used for each of the different speech program options. For example, type control-X and you will get a display similar to the following:

LPC CODE:	471	471
LABELS:	17	
WITH NO PROGRAM:	488	471
WITH RAM PROGRAM:	1631	1614
WITH LONG PROGRAM:	1716	1699
WITH SHORT PROGRAM:		697

The first line tells how many bytes of memory will be used by the words you've typed in so far. The next line shows the memory that will be used by each of the program options, both with labels and without. (The speech program options and labels will be explained further a little later in the manual.)

14. [Q] Quit

Control-Q will exit the editor and leave you in Applesoft. After typing control-Q you will be asked if you are sure you want to quit. If you answer "Y" and you haven't saved your word list, you will be asked again if you want to quit. If you answer "Y" again, you will be in Applesoft and your word list will be erased from memory.

15. [P] Program

The PROGRAM command is used to actually create the custom speech program. If you haven't saved your word list when you enter the program mode you will be warned of this fact as the process of creating a custom program will erase your word list from memory. So, if you want to save a copy of it to disk, do it before you create the speech program.

Once you enter the program mode you will be offered a selection of different programs to create. A brief description of the programs follows. A more complete description will be given later in the manual. These are your options:

(L) LONG PROGRAM: This is the program used in the demo.

[P] Program

(R) RAM PROGRAM: This is the same as the long program except that most of the program loads into a RAM card (or the extra 12K of the Apple //e).

(S) SHORT PROGRAM: This is a very short program for those interested in saving memory. It will speak the words, but it does not have any of the speech commands as shown in the demo.

(N) NO PROGRAM: This option will save only the table of LPC data and labels without a program. This option is for advanced programmers.

(E) RETURN TO THE EDITOR: This will return you to the editor.

Type "L" to select the long program for the purposes of our practice session.

You will now be asked if you want to use labels. Answer "Y" (yes) for now. The next question will be "WOULD YOU LIKE TO SPECIFY THE STARTING OR ENDING ADDRESS?".

All the speech programs are completely relocatable in memory. Appendix B shows how Applesoft programs use the memory in the Apple. Usually you will want to locate the custom speech program just below DOS. DOS starts at address $9600 hex (or 38400 decimal). Hex is the numbering system more commonly used when specifying addresses on the Apple. We will show all addresses in hex preceded by a "$" and the decimal equivalent in parenthesis.

Since DOS starts at $9600, we want our sample speech program to end at $95FF (38399). Therefore we type "E" so we can specify the ending address we want for our sample speech program. After typing "E" or "S", you will be asked for the address. You may enter the address in decimal or in hex. (Precede a hex number with a dollar sign.) For our sample program: type 38399 (a decimal address).

You will now see a display which tells you where in memory your speech program will reside (in both decimal and hex). Also, for advanced users, you are told where the beginning of the table of labels is, and where the LPC data starts. You are also given the BASIC call address. If you do not want to use the "&" in your Applesoft program, you may instead call this address to use the speech program. This was shown briefly in the demo and will be explained further in the next section of the manual. (You should make a note of the decimal value of the lowest address the speech program will use. This is the value to use in your Applesoft "HIMEM:" statement. The "HIMEM:" statement tells Applesoft not to use any memory above the value given. You may also want to write down the BASIC call address.)

Finally, you are asked if this information is correct. If you answer "N" (no), you will be able to answer the questions again correctly. An answer of "Y" will create the program. The disk will spin for a while and you will be asked to insert a disk on which to save the program, and then to type in a name for your customized speech program. You may use all the standard DOS options in the file name (Drive, Slot, Volume). Choose the name "SAMPLE" for our sample program.

You will then be asked if you want to return to the menu. If you answer "Y", the disk will re-boot and the menu will be offered to you again. Answer "N" for the purposes of this tutorial.

16. Program Options

16.1. The Long Program and the RAM Program

Both these programs have the same speech commands and are used in the same way. The only differences are the amount of memory that they use and where they are located in memory. The Long program is 1228 bytes long and may reside anywhere from $800 (2048) to just below DOS. The RAM program uses 1143 bytes in a RAM card (or the extra 12K bank of the Apple //e) and 240 bytes which may be located from $200 (512) to just below DOS.

To use a speech program in your Applesoft program, first set HIMEM: if necessary. Unless you've chosen the RAM version, and located it at $2A2 (674), this will probably be necessary. Set HIMEM: to a value below the speech program. For example, if we continue to use our program called "SAMPLE" that we just created, the first line of your program should be:

```
10 HIMEM:36684
```

The next line should BRUN the speech program.

```
20 PRINT CHR$(4);"BRUN SAMPLE"
```

(For information on using DOS command in Applesoft programs, see your DOS manual.)

You may also run the speech program without using an Applesoft program. Just type the HIMEM: statement and then BRUN SAMPLE. Do this now.

There are two ways to call the speech programs you've created. One is with the ampersand (&), the other using the "CALL" statement. For example, using the program "SAMPLE", we could say the word "I" as follows:

```
& "I"
```

Or, if you are using another program which uses the ampersand, you can call the address that was specified as the "BASIC call address" when your custom speech program was created. In our example this was 36796, so to say "i" we could type:

```
CALL 36796,"I"
```

Notice that a comma must follow the call address. If you are using another machine language program which requires the ampersand, be sure to run the speech program first, then the other program. In the rest of our examples we will use the ampersand, although in each case "CALL 36796," could be used instead.

We can say more words from our list by simply adding them to the line, separated by commas. For example:

```
& "I","LIKE","THE","APPLE"
```

All the speech commands can be entered directly or in a program. Try typing the above line. If you type a word which is not in the speech program, you will get the error message "UNDEF'D STATEMENT". This error is the same as the Applesoft error message and may be trapped by an "ONERR" statement (see your Applesoft manual).

In all the examples above we have specified the word to be spoken by using the word itself or, in Echo Words terminology, a "label". You may remember being asked if you wanted to use labels when you created the speech program. We answered "Y" (yes), but you may decide to save memory by choosing to not use labels. Not using labels saves memory not only in the speech program, but also in your Applesoft program. In an Applesoft program, one byte is used for each label character and for each quote character surrounding the label.

Even if you have specified that you would like to use labels, you may still use numbers as an alternate way of specifying the word. The number assigned to the word depends on the order in which you've typed the words. The first word in the list is always zero. The next word is one and so on. The words in our sample speech program are therefore numbered as follows:

```
0    I

1    LIKE

2    THE

3    APPLE
```

So to say "I like the Apple", we would type:

```
& 0,1,2,3
```

If you type a number which is out of the range of possible words you will see the Applesoft error message "ILLEGAL QUANTITY". This error may also be trapped by "ONERR".

You may use variables in all of the speech commands. Therefore, we can say "I like the Apple" as follows.

```
A=0 : B$="LIKE" : C$="THE"

& A,B$,C$,3
```

or

```
FOR A = 0 TO 3 : & A : NEXT A
```

You may also use expressions:

```
A=2 : B=5

& B-B,A-1,B*A/5,B-A
```

16.2. Speech Commands

There are six commands available to you when you choose the Long program or RAM program option, which enable you to vary the way the SuperSprite says words. They are:

P : to vary the pitch

V : to vary the volume

D : to vary the delay or pause between words

C : to enter the compressed mode (speak fast)

E : to enter the expanded mode (speak at normal rate of speed)

S : to add an "S" sound to any word (to make the word plural)

All these commands should be placed prior to the words you wish to modify.

16.2.1. Pitch

The pitch command (P) allows you to make a relative adjustment to the pitch of a given word. The TI 5220 speech chip used on the SuperSprite has 63 different levels of pitch available. Using the P command you may raise (+) or lower (-) the pitch level by a value from 0 to 63. for example:

```
& P-10
```

will lower the pitch by a factor of 10 in all the words which follow. The number following P should be in the range from -63 to +63, although an error message ("ILLEGAL QUANTITY") will not be generated unless it is greater than 255 or less than -255. The pitch change will remain in effect until it is reset. P0 resets it back to normal. As an example, type in:

```
& P-10,"I","LIKE",P-63,"THE",P+63,"APPLE"
```

As with all speech commands, you may use variables. Try:

```
X=20

& PX,"THE","APPLE",P0
```

or

```
FOR A = 0 TO 10 : & PX, "APPLE" : NEXT A
```

Note: If you are using a variable beginning with any of the letters P, V, D, C, E, or S, they may be misinterpreted as command letters. For example, if you set the variable PX to 3 and then type:

```
& PX
```

you might expect the SuperSprite top say "APPLE" (word number 3). However, what will happen is that the pitch will be set to the present value of the variable X. This is because the speech programs look at the first character after the ampersand or after each comma in a speech command call, and if the first character is a P, V, D, C, E, or S, then it is assumed to be a command. If the first character is a quote, then it is assumed to be a label. Otherwise it will be processed as a variable or expression. So, if you want to use a variable beginning with P, V, D, C, E, or S you must do it as follows:

```
& 0 + PX
```

This will be assumed to be an expression and word number PX will be spoken since 0 + PX = PX.

16.2.2. Volume

The V command modifies volume similarly to the way the pitch command modifies pitch. However, the number following V should be in the range -15 to +15. Like the P command, an error message ("ILLEGAL QUANTITY") will not be generated unless the number is greater than 255 or less than -255. V0 resets the volume to its normal level.

As an example, type:

```
& V-5,"I","LIKE",V0,"THE",V+5,"APPLE"

&V0
```

16.2.3. Delay

The D command increases the length of the delay between words. The number following D should be in the range 0 to 255. D0 is the normal operating mode. As an example, type:

```
& D100,"I","LIKE","THE","APPLE"
```

or for a pause only after "I":

```
& D100,"I",D0,"LIKE","THE","APPLE"
```

Remember the D command will take effect in the pause following the word it precedes.

16.2.4. Compressed and Expanded

The compressed mode (C command) will cause the SuperSprite to speak twice as fast and the expanded mode (E command) returns the SuperSprite to the normal speaking rate. These commands are similar to their counterparts in TEXTALKER. Type:

```
& C,0,1,2,3
```

```
& E,0,1,2,3
```

16.2.5. Adding an "S"

The final command is the S command which you may use to add an "S" sound to any word or phrase. Remember that commands precede the words they modify, so the "S" command will precede the word to be made plural. Also, the S command only affects the first word to follow it. It does not remain in effect as do all the other commands. Example:

```
& "I","LIKE",S,"APPLE"
```

will say "I LIKE APPLES"

16.3. The Short Program

The short program is only 226 ($E2) bytes long, and 61 bytes are only used for initializing the program and are then free. Therefore you may load the short program at $2C3 (707) and, after it is run, it will only use the memory from $300 to $3A5 (768 to 933), meaning you don't have to set HIMEM:. The drawback to the short program is that it has none of the options of the Long program (P, V, D, C, E, S). Also, it doesn't use the ampersand, and it can't use labels. You may only call the short program with the number of the word to be spoken. For example:

```
CALL ####,0,1,2
```

Therefore, make sure you note the BASIC call address when you create a Short program. If you forget, Appendix C explains how to calculate this address.

Please note that the Short program does not check for illegal entries. If you give the SuperSprite an invalid word number, press RESET immediately.

16.4. No Program

This option is only for advanced users. It simply saves the LPC data and, if you like, the label table. It is usually used when memory size and disk space are a problem. Typically, one version of a speech program is loaded into memory, and then overlayed on the speech data already in memory. It may be also used by assembly language programmers who wish to design their own speech drivers.

Appendix A: Echo Words Vocabulary List

Word	Length	Word	Length	Word	Length
DISK DRIVE	150	DISKETTE	96	DIVIDE	144
DIVISION	128	DO	93	DOES	89
DOG	97	DOING	124	DOLLAR	115
DOUBLE	95	DOWN	125	DRAW	112
DRAWING	167	DURING	120		
E	44	EACH	77	EARLY	142
EARTH	65	EAT	59	ECHO II	148
ECHO WORDS	218	ECONOMIC	141	EIGHTEEN	160
EIGHTY	102	EITHER	106	ELECTRONICS	167
ELEVEN	119	ELSE	83	END	80
ENGLISH	143	ENOUGH	91	ENTER	113
ERROR	108	ESCAPE	107	EVEN	108
EVER	99	EVERY	138	EXACTLY	190
EXAMPLE	169	EXPERIENCE	163		
F	56	FACE	85	FACT	86
FAMILY	141	FAR	117	FAST	99
FATHER	107	FEBRUARY	180	FEDERAL	132
FEEL	120	FEET	77	FELT	93
FEMALE	138	FEW	108	FIELD	138
FIFTEEN	148	FIFTH	84	FIFTY	131
FIGURE	130	FIND	116	FINE	144
FINISH	98	FINISHED	105	FIRE	126
FIRST	93	FISH	81	FIT	78
FOLLOWING	177	FOOD	111	FORM	130
FORTY	127	FOUND	125	FOURTEEN	181
FOURTH	93	FRACTION	120	FREE	118

Word	Length	Word	Length	Word	Length
BETWEEN	160	BIG	95	BILLION	136
BLACK	80	BLUE	108	BOARD	128
BODY	139	BOOK	69	BOTH	75
BOTTOM	104	BOY	129	BROUGHT	86
BROWN	126	BUSINESS	118	BUT	69
BY	116				
C	107	CALL	109	CALLED	117
CALLING	120	CAME	102	CAN	101
CANNOT	104	CAR	92	CARD	108
CASE	78	CASSETTE	101	CENT	94
CENTER	110	CERTAIN	112	CHANGE	118
CHECK	65	CHILDREN	147	CHOICE	82
CHURCH	78	CIRCLE	132	CITY	117
CLEAR	121	CLOCK	94	CLOSE	113
CLOSED	145	CODE	118	COLD	127
COLLEGE	136	COLOR	102	COLUMN	138
COME	74	COMMA	92	COMMAND	150
COMPANY	130	COMPLETE	129	COMPLETED	133
COMPUTER	162	CONNECTED	128	CONSOLE	150
CONTINUE	191	CONTROLLER	180	CORRECT	99
COULD	92	COUNTRY	133	COURSE	101
CUT	70	CYAN	156		
D	69	DARK	70	DATA	97
DAY	80	DEATH	77	DECEMBER	132
DEGREE	129	DELAY	164	DEVELOPMENT	166
DEVICE	124	DID	86	DIFFERENT	135

Appendix A: Echo Words Vocabulary List

Word	Length	Word	Length	Word	Length
DISK DRIVE	150	DISKETTE	96	DIVIDE	144
DIVISION	128	DO	93	DOES	89
DOG	97	DOING	124	DOLLAR	115
DOUBLE	95	DOWN	125	DRAW	112
DRAWING	167	DURING	120		
E	44	EACH	77	EARLY	142
EARTH	65	EAT	59	ECHO II	148
ECHO WORDS	218	ECONOMIC	141	EIGHTEEN	160
EIGHTY	102	EITHER	106	ELECTRONICS	167
ELEVEN	119	ELSE	83	END	80
ENGLISH	143	ENOUGH	91	ENTER	113
ERROR	108	ESCAPE	107	EVEN	108
EVER	99	EVERY	138	EXACTLY	190
EXAMPLE	169	EXPERIENCE	163		
F	56	FACE	85	FACT	86
FAMILY	141	FAR	117	FAST	99
FATHER	107	FEBRUARY	180	FEDERAL	132
FEEL	120	FEET	77	FELT	93
FEMALE	138	FEW	108	FIELD	138
FIFTEEN	148	FIFTH	84	FIFTY	131
FIGURE	130	FIND	116	FINE	144
FINISH	98	FINISHED	105	FIRE	126
FIRST	93	FISH	81	FIT	78
FOLLOWING	177	FOOD	111	FORM	130
FORTY	127	FOUND	125	FOURTEEN	181
FOURTH	93	FRACTION	120	FREE	118

Appendix A: Echo Words Vocabulary List

Word	Length	Word	Length	Word	Length
FRIDAY	159	FROM	89	FRONT	99
FUTURE	119				
G	134	GAME	161	GAVE	180
GENERAL	171	GET	97	GETTING	154
GIVE	155	GIVEN	155	GO	143
GOD	155	GOING	136	GOOD	130
GOOD WORK	186	GOODBYE	171	GOT	78
GOVERNMENT	180	GRAY	180	GREAT	106
GREEN	168	GROUND	186	GROUP	125
GROW	143	GUESS	64		
H	82	HAD	147	HALF	98
HAND	188	HARD	159	HAS	147
HAVE	109	HAVING	156	HE	147
HE IS	230	HEAD	137	HEARD	154
HELD	150	HELLO	144	HELP	107
HER	131	HERE	145	HIGH	137
HIGHER	147	HIM	131	HIMSELF	178
HIS	134	HISTORY	162	HIT	69
HOME	147	HOUR	133	HOUSE	107
HOW	141	HOWEVER	166	HUMAN	166
HUNDRED	198	HURRY	141		
I	133	I AM	239	I WIN	201
IDEA	135	IDENTIFICATION	279	IF	90
IMPORTANT	195	IN	78	INCH	97
INCHES	166	INDIVIDUAL	224	INFORMATION	264

Appendix A: Echo Words Vocabulary List

Word	Length	Word	Length	Word	Length
INSIDE	217	INSTRUCTION	216	INTEGER	134
INTEREST	168	INTO	169	IS	139
IT	70	IT IS	167	ITSELF	145
J	109	JANUARY	163	JOB	134
JOYSTICK	168	JULY	125	JUNE	131
JUST	80				
K	80	KEEP	89	KEY	116
KEYBOARD	158	KIND	169	KNOWN	170
L	112	LAND	140	LANGUAGE	187
LARGE	137	LARGER	166	LARGEST	130
LASER	148	LAST	96	LATER	125
LAW	114	LEARN	128	LEARNED	96
LEAST	83	LEAVE	150	LEFT	102
LESS	96	LET	88	LETTER	95
LIFE	114	LIGHT	95	LIKE	91
LINE	146	LITTLE	113	LIVE	128
LIVED	129	LIVING	140	LOAD	116
LOADING	167	LOCAL	134	LONG	133
LOOK	103	LOOKED	93	LOOKING	132
LOW	110	LOWER	143		
M	90	MACHINE	149	MADE	130
MAGENTA	155	MAJOR	134	MAKE	97
MAKING	150	MALE	117	MAN	119
MANY	105	MARCH	109	MATTER	101
MAY	90	ME	108	MEAN	129

Word	Length	Word	Length	Word	Length
MEDIUM	214	MEMBER	107	MEMORY	150
MEN	87	MESSAGE	135	MIDDLE	107
MIGHT	92	MILE	106	MILLION	172
MIND	124	MINUTE	97	MISS	62
MODULE	150	MOMENT	137	MONDAY	148
MONEY	117	MONITOR	158	MORE	95
MORNING	162	MOST	101	MOTHER	99
MOVE	136	MR	132	MRS	123
MS	141	MUCH	75	MULTIPLICATION	206
MULTIPLY	169	MUST	91	MY	142
N	100	NAME	121	NATIONAL	139
NEAR	124	NEAT	91	NEED	124
NEGATIVE	124	NEVER	96	NEW	117
NEXT	101	NICE TRY	151	NIGHT	87
NINETEEN	188	NINETY	151	NO	98
NOT	78	NOTHING	149	NOVEMBER	184
NOW	149	NUMBER	118		
O	102	O'CLOCK	123	OCTOBER	170
OF	65	OFF	81	OFFICE	89
OFTEN	120	OLD	111	ON	96
ONCE	101	ONLY	156	OPEN	124
OR	78	ORANGE	149	ORDER	189
OTHER	117	OUT	79	OVER	124
OWN	126				
P	98	PADDLE	129	PAGE	121

Appendix A: Echo Words Vocabulary List

Word	Length	Word	Length	Word	Length
PAPER	107	PART	71	PARTNER	115
PAST	98	PEOPLE	93	PER	67
PERHAPS	113	PERIOD	109	PERSONAL	108
PHOTON	153	PICTURE	111	PINK	77
PITCH	92	PLACE	89	PLANT	93
PLAY	96	PLEASE	113	POINT	98
POLITICAL	138	POSITION	121	POSITIVE	156
POSSIBLE	125	POWER	119	PREESENT	113
PRESENT	133	PRESIDENT	132	PRESS	77
PRINT	71	PRINTER	103	PROBABLY	148
PROBLEM	128	PROGRAM	178	PROUD	118
PUBLIC	98	PURPLE	85	PUT	84
PUTTING	111				
Q	77	QUARTER	115	QUESTION	122
QUITE	93				
RAM	107	RANDOM	163	RATHER	110
RAY	121	READY	111	READY TO START	210
REAL	116	REALLY	132	REASON	148
RECORDER	134	RED	112	REED	125
REFER	119	REMEMBER	148	RESET	110
REST	102	RESULT	121	RETURN	140
REWIND	174	RIGHT	101	RINGING	158
ROM	146	ROOM	136	ROUND	141
ROW	115	RUN	115		
S	65	SAID	118	SAME	117

Appendix A: Echo Words Vocabulary List

Word	Length	Word	Length	Word	Length
SATURDAY	132	SAVE	103	SAVING	131
SAW	89	SAY	97	SAYS	91
SCHOOL	133	SCREEN	143	SECOND	103
SEEM	105	SEEMED	119	SEEN	111
SELECT	106	SELECT ONE OF THE FOLLOWING			360
SENSE	73	SENTENCE	133	SEPTEMBER	155
SERVICE	99	SET	74	SEVENTEEN	201
SEVENTY	169	SEVERAL	128	SHALL	107
SHAPE	82	SHE	120	SHE IS	167
SHIFT	98	SHORT	108	SHORTER	135
SHOULD	104	SHOW	103	SHOWN	137
SIDE	120	SINCE	85	SIXTEEN	173
SIXTY	130	SLOT	117	SLOW	131
SMALL	132	SMALLER	130	SMALLEST	126
SO	121	SOCIAL	127	SOCIETY	171
SOME	129	SOMETHING	133	SOMETIME	162
SOON	141	SORRY	129	SOUND	147
SOUTH	98	SPACE	125	SPEAK	106
SPEAKEASY	168	SPEAKER	116	SPECIAL	133
SPEECH	95	SPELL	127	SPORT	129
SQUARE	129	START	109	STARTED	139
STATE	91	STEP	86	STILL	109
STOP	101	STORY	163	STREET	100
STUDY	111	SUBTRACT	179	SUBTRACTION	182
SUCH	91	SUM	118	SUN	106
SUNDAY	127	SUPPOSED	162	SURE	131
SWITCH	117	SYNTHESIZER	178	SYSTEM	129

Appendix A: Echo Words Vocabulary List

Word	Length	Word	Length	Word	Length
T	118	TABLE	109	TAKE	82
TAKEN	110	TAPE	74	TELEPHONE	164
TELEVISION	198	TELL	107	TEMPERATURE	164
TEN	106	TEXTALKER	156	THAN	116
THANK YOU	150	THAT	88	THAT IS CORRECT	206
THAT IS RIGHT	154	THE	85	THEIR	96
THEM	111	THEMSELVES	180	THEN	114
THERE	99	THESE	119	THEY	133
THEY ARE	180	THING	114	THINK	77
THIRD	108	THIRTEEN	166	THIRTY	168
THIS	91	THOSE	144	THOUGH	102
THURSDAY	136	THUS	111	TIME	91
TODAY	96	TOGETHER	157	TOLD	123
TONE	100	TOOK	70	TOP	82
TORPEDO	164	TOWARD	124	TREE	117
TRIANGLE	120	TRUE	117	TRY	86
TRY AGAIN	151	TUESDAY	171	TURN	87
TURNED	127	TWELVE	156	TWENTY	114
TYPE	84				
U	113	UNDER	124	UNDERSTAND	186
UNITED STATES	228	UNTIL	161	UP	50
UPON	138	UPPER	88	US	64
USE	142	USED	156	USING	176
USUALLY	189				
V	140	VERY	149	VOCABULARY	220
VOICE	125	VOLUME	169		

Appendix A: Echo Words Vocabulary List

Word	Length	Word	Length	Word	Length
W	153	WAIT	76	WANT	115
WANTED	157	WAR	126	WAS	110
WATER	129	WAY	110	WE	127
WE ARE	174	WEDNESDAY	147	WEEK	93
WELCOME	156	WELL	138	WENT	106
WERE	103	WEST	96	WHAT	75
WHAT WAS THAT	159	WHEN	132	WHERE	111
WHETHER	101	WHICH	93	WHILE	134
WHITE	74	WHO	107	WHOLE	112
WHOSE	141	WHY	126	WILL	112
WIN	128	WIND	132	WITH	93
WITHIN	129	WITHOUT	108	WORD	135
WORK	91	WORKING	146	WORLD	128
WOULD	124				
X	57				
Y	126	YEAR	114	YELLOW	137
YES	93	YET	115	YOU ARE	167
YOU WIN	140	YOUNG	110	YOUR	138
Z	124				

18. Appendix B: Applesoft Memory Usage

Listed below you will find a memory map. If it looks confusing, don't worry about it. Just follow these rules:

If you are using TEXTALKER in conjunction with Echo Words, make sure the speech program ends at $76FF, and BRUN TEXTALKER first, then the speech program. Otherwise, make sure the speech program ends below DOS ($95FF). In both cases, set HIMEM: to the lowest address of the speech program after running the speech program. You may also load the RAM program at $2A2 or the Short program at $2C3. If you do this, you don't need to set HIMEM:.

$D000-FFFF (53248-65535)	ROM. Applesoft and the Monitor. If you have a RAM card, the extra memory is bank-switched here.
$C000-CFFF (49152-53247)	I/O addresses. Speaker, keyboard-strobe, etc.
$9600-BFFF (38400-49151) (HIMEM:38400)	DOS. Unless specifically set by you, DOS will set HIMEM: to this point upon booting or "FP". TEXTALKER will also set HIMEM:, see the TEXTALKER manual.
(Applesoft strings)	Applesoft will not store anything above HIMEM:. Applesoft stores strings from HIMEM: on down. Type the following to find the lowest point of string storage: ? PEEK(111)+PEEK(112)*256
$4000-5FFF (16384-24575)	High resolution graphics page 2. This section of memory stores the image you see on the screen.
$2000-3FFF (8192-16383)	High resolution graphics page 1.
(array pointers and numeric variables)	These numbers are stored just above LOMEM:. The end of this storage is: ? PEEK(109)+PEEK(110)*256
(LOMEM:)	The end of your Applesoft program, unless changed by you.
$800- (2048-)	Start of your Applesoft program.
$400-7FF (1024-2047)	Text screen memory.
$3D0-3FF (976-1023)	Reserved for DOS and the monitor.

$300-3CF	(768-975)	FREE SPACE. You may store the Short program or RAM program here.
$200-2FF	(512-767)	Input buffer. Inputs are temporarily stored here. This space may be used by the initialization routines of the Short and RAM programs.
$0-1FF	(0-511)	Reserved for DOS, Applesoft, and the Monitor.

19. Appendix C: Useful Speech Program Addresses

If you've forgotten where a speech program loads into memory, or what the BASIC call address is, this appendix will help you.

If you are using DOS 3.3 on a 48K or greater Apple, you may find the starting address of a speech program by first BRUNning the program and then typing:

```
PRINT PEEK(43634) + PEEK(43635)*256
```

The following table gives the addresses (relative to the start of the program) of various sections of the speech programs.

	Long		Short		RAM	
Initialization	start-$68	(104)	start-$3C	(60)	start-$5D	(93)
Ampersand vector	$69	(105)	----------		$D8	(216)
BASIC call address	$70	(112)	$3D	(61)	$E2	(226)
Animation vector	$77	(119)	----------		* $D067	(53351)
Pointers:						
Label table	$4C8	(1224)	----------		* $D4D3	(54387)
LPC data	$4CA	(1226)	** LO- $53	(83)	* $D4D5	(54389)
			$64	(100)		
			HI- $59	(89)		
			$6C	(108)		

* Absolute address in RAM card.

** The short program has two pointers to the LPC data.

20. Appendix D: How to Overlay LPC Data

If you are an advanced programmer using a long list of words with your program and they do not fit into memory all at one time, this appendix is for you.

What you will need to do is to divide your word list into more than one group. Then you'll load into memory each group of words as necessary. The procedures to do this vary slightly depending on whether or not you are using labels. Both methods are given below.

If you are not using labels

1. Save your first group of words as either the Long program or Short program. Write done the starting address of the LPC data.

2. Type in your next group of words and save these with the "No program" option. The starting address should be the address you noted in step 1.

3. In your Applesoft program, BRUN the speech program. Whenever you need a new group of words, simply BLOAD them from the disk.

If you are using labels

1. Save your first group of words as the Long program. Write down the starting address of the label table.

2. Type in your next group of words and save these with the "No program" option. The starting address should be the address you noted in step 1. Write down the starting address of the LPC codes for this group of words.

3. In your Applesoft program, BRUN the speech program. Whenever you need a new group of words, BLOAD the group from your disk and then you'll need to do a couple of POKEs.

   First, calculate the address of the LPC pointer using Appendix C. Assuming LP equals the address of the LPC pointer and L equals the address you noted in step 2, insert the following statements into your program after the BLOAD statement.

   ```
   POKE LP, (L+2)-INT((L+2)/256)*256

   POKE LP+1, INT((L+2)/256)
   ```

   Now you may speak the words you've just loaded into memory.

21. Appendix E: How to Use the Animation Vector

Assembly language programmers may do animation while the SuperSprite is speaking! You may have noticed this being done at the opening of the demo when the asterisks are drawn around the screen while the SuperSprite is speaking. While speaking, both the Long program and the RAM program periodically jump to a point in memory we call the animation vector. The address of the vector is given in the table in Appendix C.

Normally the vector contains 3 RTS statements. You may insert a JMP command to your own routine here. To try an example you can insert a JMP to the scroll routine (4C 70 FC). Now, every time the SuperSprite says a word, the screen will scroll while it speaks; not very useful but it shows how the vector works. The only limitation is that if your routine uses too much time the SuperSprite will "crash". Your animation routine should take less than 19,500 clock cycles, or approximately 4,875 instructions.

22. Appendix F: The Format of the LPC Data and Label Tables

LPC data is stored in the following format:

(all addresses are in the standard low/high format)

Bytes 0 and 1	Number of entries in the table. Pointer to
2 and 3	the start of the LPC code for the first entry. All the pointers are relative offsets from this point.
4 and 5	Pointer to the next entry.
.	
.	
.	
n and n+1	Pointer to the last entry.
n+2 and n+3	Two zero bytes.
n+4	Start of LPC data for the first entry.

Here is an example, using an imaginary table:

00 02	06 00	09 00
number of words	pointer to word #0	pointer to word #1
00 00	45 77 5F	63 ...
zero bytes	start of word #0	start of word #1

The label table consists of the labels in the order they were typed. Each character of the label is stored with the high bit off, except for the last character which has the high bit set. For example, the labels "I", "LIKE", "THE" would be stored as follows:

C9 4C 49 4B C5 54 48 C5

I L I K E T H E

Appendix G: Writing Your Own Speech Program

You may write your own version of the Short program for use from assembly language. A listing of the Short program, with comments on how to modify it for assembly language use, follows this paragraph. You can create your speech files using the "No program option and use them with your own speech driver. Here is a general outline of the program:

LINES	47-75	Search to find the SuperSprite, if not found, look for Echo][in alternate slots.
	85	This is where BASIC calls the program.
	92	You may call the program here from assembly language. The word number should be in the X register.
	93-96	Save the zero page variables on the stack.
	98-125	Multiply the word number by 2 to use as a pointer to the table of offsets (see Appendix F). Add this value to the start of the table to obtain the starting address of this word and the starting address of the next word (which is also the end of this word).
	128-130	Make sure the SuperSprite has stopped speaking any prior words. Always do this.
	132-137	Initialize the TMS5220. You must always do this prior to sending the SuperSprite new data.
	140-151	Send all the LPC data.
	154	Reset the SuperSprite (or Echo][, if SuperSprite not present). You must always reset the TMS5220 by sending it 9 $FF's at the end of a stream of LPC data.

23. Program Listing

```
 1 *********************************
 2 *                               *
 3 *       SHORT VERSION OF        *
 4 *    FIXED SPEECH ROUTINES.     *
 5 *                               *
 6 *    WRITTEN BY MIKE KORY       *
 7 *                               *
 8 *    COPYRIGHT 1983             *
 9 *    STREET ELECTRONICS CORP.   *
10 *                               *
11 *********************************
12                          ;
13 BEG        EQU $0        ;BEGINNING OF LPC CODE FOR WORD
14 END        EQU $2        ;END OF LPC CODE FOR WORD
15 TEMP       EQU $4        ;TEMPORARY POINTER (NOTE THAT ONLY ONE
16                          ; 0-PAGE POINTER IS ACTUALLY USED AT A TIME
17                          ; SO A SLIGHTLY LONGER PROGRAM CAN BE WRITTEN
18                          ; THAT DOESN'T USE AS MUCH 0-PAGE)
19                          ;
20 COMMABYT   EQU $E74C     ; FP ROM ROUTINE
21 CHARGOT    EQU $B7       ; FP ROUTINE TO GET CHARACTER AT TEXT POINTER
22                          ;
23 SLOT1      EQU $90
24 SLOT2      EQU $A0
25 SLOT3      EQU $B0       ;LOOK FOR ECHO ][ IN SLOTS
26 SLOT4      EQU $C0       ;1 THROUGH 6
27 SLOT5      EQU $D0
28 SLOT6      EQU $E0
29 SLOT7      EQU $F2       ;SUPERSPRITE IN SLOT 7
30                          ;
31 BASEADDR   EQU $C000     ;BASE ADDRESS OF I/O SPACE
32 LPCTABLE   EQU $2000
33                          ;
34            ORG $1000
35                          ;
36 *********************************
37 * FIND SUPERSPRITE OR ECHO AND  *
38 *     MODIFY THE PROGRAM.       *
39 *                               *
40 * SLOTS ARE CHECKED IN THE      *
41 * ORDER SHOWN IN ORDER TO       *
42 * AVOID FINDING A CARD WHICH    *
43 * LOOKS LIKE ECHO (SUCH AS A    *
44 * RAM CARD                      *
45 *********************************
46                          ;
47 FINDECHO
48            LDA #SLOT7    ;CHECK FOR SUPERSPRITE FIRST
49            JSR CONFIG
50            LDA #SLOT4    ;NOW LOOK FOR ECHO ][
51            JSR CONFIG
52            LDA #SLOT5
53            JSR CONFIG
```

```
 54            LDA #SLOT3
 55            JSR CONFIG
 56            LDA #SLOT2
 57            JSR CONFIG
 58            LDA #SLOT1
 59            JSR CONFIG
 60            LDA #SLOT6
 61            JSR CONFIG
 62            RTS             ;NOT FOUND
 63 CONFIG
 64            STA ECHO1
 65            STA ECHO2
 66            STA ECHO3
 67            STA ECHO4
 68            JSR READECHO
 69            AND #$9F
 70            CMP #$1F
 71            BEQ FOUNDIT     ;FOUND SUPERSPRITE OR ECHO
 72            RTS
 73 FOUNDIT    PLA             ;POP RTS OFF STACK
 74            PLA
 75            RTS
 76                            ;
 77 ********************************
 78 *     CALL HERE FROM BASIC     *
 79 *          CALL ###,X          *
 80 *   WHERE ### IS LOCATION OF   *
 81 *   THIS ROUTINE, AND X IS THE *
 82 *   DESIRED WORD NUMBER.       *
 83 ********************************
 84                            ;
 85 BASIC      JSR COMMABYT ;FP ROM ROUTINE TO GET A COMMAND AND A # IN X
 86                            ;
 87 ********************************
 88 *   JSR HERE FROM MACHINE LANG *
 89 *   WITH THE WORD NUMBER IN X  *
 90 ********************************
 91                            ;
 92 MACHINE    LDY #5          ;SAVE ZERO PAGE VARIABLES
 93 PHALOOP    LDA BEG,Y       ;(NEED TO USE Y AS INDEX SINCE X HAS WORD #)
 94            PHA
 95            DEY
 96            BPL PHALOOP
 97                            ;
 98 SAYWORD    LDA #$0         ;MULTIPLY WORD # BY 2
 99            STA TEMP+1
100            TXA             ;WORD # IS NOW IN A
101            ASL
102            ROL TEMP+1      ;NEED 2 BYTES SINCE WORD # RANGE IS 0-255
103            CLC
104            ADC #<LPCTABLE ;ADD 2 * WORD NUMBER TO START OF LPCTABLE
105            STA TEMP        ;     TO POINT TO LPC TABLE'S POINTERS
106            LDA TEMP+1
107            ADC #>LPCTABLE
108            STA TEMP+1
```

```
109                         ;
110             LDX #0      ;CALCULATE BEGINNING AND END OF WORD
111             LDY #0
112 LOADLOOP
113             CLC
114             LDA (TEMP),Y
115             ADC #<LPCTABLE
116             STA BEG,X
117             INY
118             INX
119             LDA (TEMP),Y
120             ADC #>LPCTABLE
121             STA BEG,X
122             INY
123             INX
124             CPX #4
125             BNE LOADLOOP
126                         ;
127 WAITSTOP
128             JSR READECHO ;MAKE SURE 5220 HAS STOPPED SPEAKING
129             AND #$80
130             BNE WAITSTOP
131                         ;
132 INITECHO
133             LDA #$60    ;INITIALIZE 5220
134             STA BASEADDR
135 ECHO1       EQU *-2     ;LABEL SO THAT FINDECHO ROUTINE CAN MODIFY CODE
136             LDA ($0,X)  ;TIME DELAY
137             LDA ($0,X)
138                         ;
139             LDY #$0     ;START SPEAKING
140 SPEECHLOOP
141             LDA (BEG),Y ;GET BYTE TO SENT TO ECHO
142             JSR SENDECHO ;SEND BYTE TO ECHO
143             INC BEG     ;INCREMENT POINTER
144             BNE END?
145             INC BEG+1
146 END?        LDA BEG+1   ;HAVE WE SENT ALL THE DATA FOR THIS WORD?
147             CMP END+1
148             BNE SPEECHLOOP
149             LDA BEG
150             CMP END
151             BNE SPEECHLOOP
152                         ;
153             LDY #9      ;SEND ECHO 9 $FF'S TO RESET THE 5220
154 RESET       LDA #$FF
155             JSR SENDECHO
156             DEY
157             BNE RESET
158                         ;
159                         ;IF BEING CALLED BY A MACHINE LANGUAGE PROGRAM
160                         ;-DELETE LINES 162 THROUGH 166
161                         ;
162             JSR CHARGOT ;GET NEXT CHAR IN BASIC LINE
163             CMP #','    ;IF IT'S A COMMA THEN THERE'S ANOTHER #
```

```
164             BNE ALLDONE
165             JSR COMMABYT ;GET NEXT VALUE
166             JMP SAYWORD  ;START OVER
167 ALLDONE     LDX #0       ;RESTORE ZERO PAGE VARIABLES FROM STACK
168 PLALOOP     PLA
169             STA BEG,X    ;VARIABLES ARE BEG PTR END & TEMP POINTERS
170             INX
171             CPX #6
172             BNE PLALOOP
173             RTS
174                          ;
175 ***********************************
176                          ;
177 SENDECHO
178             STA TEMP     ;SAVE VALUE TO SEND THE ECHO TEMPORARILY
179 SENDLOOP    JSR READECHO ;SEE IF 5220 IS READY TO RECEIVE A BYTE
180             AND #$40
181             BNE READY
182             JSR READECHO
183             AND #$80
184             BNE SENDLOOP
185             RTS          ;PROGRAM WILL GET HERE ONLY IF 5220 CRASHES
186                          ;- THIS SHOULDN'T EVER HAPPEN IF ONLY VALID
187                          ;  DATA IS SENT TO THE CHIP.
188 READY       LDA TEMP
189             STA BASEADDR
190 ECHO2       EQU *-2      ;LABEL SO FINDECHO CAN MODIFY CODE
191             RTS
192                          ;
193 READECHO
194             LDA BASEADDR ;READ THE 5220'S STATUS
195 ECHO3       EQU *-2
196             LDA ($0,X)   ;(TIME DELAYS)
197             LDA ($0,X)
198             LDA BASEADDR
199 ECHO4       EQU *-2
200             RTS
```