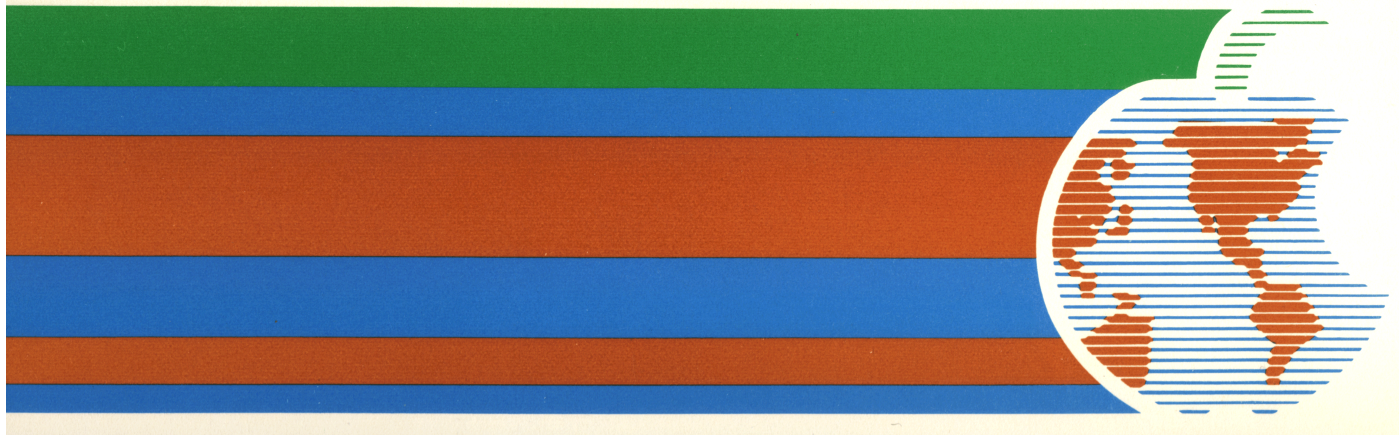
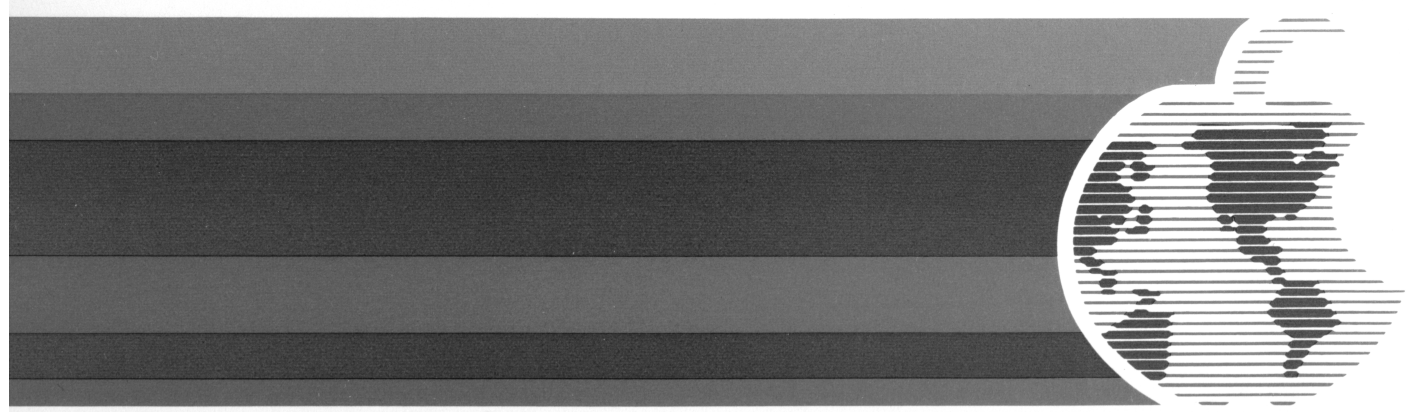


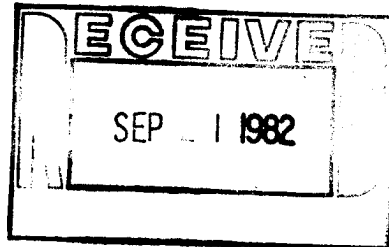
apple tech notes



apple tech notes







August 27, 1982

Dear IAC Committee Chairperson:


Thank you for all your help and participation concerning the IAC. Enclosed you will find your official copy of the new "APPLE TECH NOTES".

Please fill out the "Subscription Form" and send it back to the office. We will keep good records on all binders sent, sold, or given. The other reason for sending the form back is to get on the "Update" list. In this way your name will not be left off - DO NOT SEND ANY MONEY WITH THE FORM. Please indicate your position with the IAC on the back for the form.

BOARD OF DIRECTORS MEETING - SEPT. 30, OCT. 1 - 3, 1982

The Board of Directors will be having a special meeting, as indicated above. If you have any input, problems, or questions on any IAC subject you will like taken up by the Board, please drop the office a note. We will try and get an answer for you at the meeting.

Best Regards,


Ken Silverman
President

NOTICE

This information has been provided to the International Apple Core by Apple Computer, Inc. for informational purposes only. No representation is made by either the International Apple Core or Apple Computer, Inc. as to its accuracy or applicability for your particular purposes. For authoritative advice concerning Apple products, consult your local authorized Apple dealer, who can help you tailor this information to your specific needs.

In no event will the IAC or Apple Computer be liable for direct, indirect, incidental, or consequential damages resulting from this information or the use thereof, even if the IAC and/or Apple Computer have been advised of the possibility of such damages.

This information is copyrighted, all rights reserved. This document, or any part thereof, may not, in whole or part, be copied, photocopied, reproduced, translated or reduced to any electronic medium or machine readable form without prior consent, in writing, from Apple Computer, Inc.

Copyright © 1982 - Apple Computer, Inc.

As changes and new material are released, the IAC will make them available, on a quarterly basis, to purchasers of Apple Tech Notes who wish to subscribe to the service. Please fill out the form below and mail it to:

**International Apple Core
Apple Tech Notes Division
910 A George Street
Santa Clara, CA 95050**

UPDATES

When you subscribe to the "Apple Tech Notes" Updates, the changes and additions are flagged by the use of a vertical change bar.

What are change bars? They are those vertical lines in the right margin of some of the notes. They indicate where information has been added or changed from the last issue of that note. the index pages (XXXX.000) will probably always have the most change bars.

 International Apple Core - "APPLE ORCHARD" - Subscription Information

The APPLE ORCHARD is the premier magazine for users of Apple microcomputers. It is published by the International Apple Core, a federation of Apple Users' Groups all over the world, although its circulation is not limited to group members only. The readership and contents spans the full range of involvement from the newcomers to experts and the publication is edited to encourage the curiosity and creativity of readers at all skill levels.

Each issue is a mixture of features covering uses for the computer, simple and complex programs in the BASIC and Pascal languages, and analysis of issues affecting the level of benefit and enjoyment derived by the reader.

Regular departments include "Forbidden Fruit," the most comprehensive discussion of new products for the Apple community in any periodical; "Planting a Seed," a series of thought-provoking essays on the human aspects of computers; unbiased reviews of new, old, and improved products and services, and other regular columns.

USE THE SUBSCRIPTION FORM BELOW

International Apple Core, 910 A George St., Santa Clara, CA 95050

The International Apple Core makes individual subscriptions to "The Apple Orchard" available:

NAME _____

STREET _____

CITY _____ STATE _____ ZIP _____

COUNTRY _____

Annual Subscription Rate: \$15.00 for 6 Issues

First Class Postage: \$7.50 additional (required for Canada, Mexico, APO, and FPO addresses)

Overseas and other foreign air mail postage (required): \$15.00 additional

TOTAL REMITTANCE ENCLOSED: \$(USA) _____

Make check or money order payable to "International Apple Core" and return with this form to:

**International Apple Core
 910 A George St.
 Santa Clara, CA 95050**

TOTAL REMITTANCE ENCLOSED: \$(USA) _____

APPLE COMPUTER TECHNICAL NOTES

Issued 2 Jul 82

General Information

Page 1000.000.01

INDEX TO GENERAL INFORMATION

ALPHABETIC LISTING

Doc	Title
001	Index to subjects
002	Introduction to Tech Notes

NUMERIC LISTING

Doc	Issue Date	Pre Date	Pages
001	JUL 2 82	MAY 19 82	3
002	OCT 28 81	-	1

INDEX TO SUBJECTS

The following two lists are all for the currently defined Tech Note categories. This index can be used to refer to the proper section of Tech Notes to find an answer. For example, a question about Apple /// Emulation Mode would be answered in section 1600.

ALPHABETIC LISTING

- 1600 Apple /// Emulation Mode
- 1500 Apple /// Hardware
- 1700 Apple /// Interfacing
- 1800 Apple Adventure
- 1900 Apple Bowl
- 1300 Apple II Hardware
- 1400 Apple II Interfacing
- 2200 Apple Plot
- 2300 Apple Post
- 9600 Apple PROMs
- 2400 Apple Stellar Invaders
- 2500 Apple Writer
- 2550 Apple Writer ///
- 2600 Applesoft
- 2700 Applesoft Firmware Card
- 2800 Artist Designer
- 2900 Auto-Start ROM
- 9300 Bulletin Board Systems
- 3100 Business Basic
- 3130 Business Graphics ///
- 3160 Business Graphics II
- 3200 Centronics Printer Interface
- 3300 Communications Interface
- 3700 DOS
- 3800 DOS Tool Kit
- 4100 Elementary, My Dear Apple
- 4200 Formulex
- 4300 FORTRAN
- 1000 General Information
- 4600 Goodspell
- 4700 Graphics Tablet
- 4800 Hand Holding Basic
- 4900 High Speed Serial Interface
- 5100 Integer Basic
- 5200 Integer Basic Firmware Card
- 5500 Language Card
- 5600 Microchess 2.0
- 6200 Parallel Printer Interface
- 6100 Pascal
- 6150 Pascal ///

(Continued)

APPLE COMPUTER TECHNICAL NOTES

Page 1000.001.02

General Information

Issued 2 Jul 82

6000 Pascal Animation Tools
6400 Pilot
6600 Plan80
6650 Profile
6700 Programmer's Aid #1
6800 Psort
7100 The Shell Games
7200 Silentype
7250 Softcard ///
7275 Spelling Strategy
7300 Super Serial Card
7400 Supermap
7500 Tax Planner
7650 Universal Parallel Card
9900 Vendor List
7800 Visicalc ///
8000 VT-100 Emulator

NUMERIC LISTING

1000 General Information
1300 Apple II Hardware
1400 Apple II Interfacing
1500 Apple /// Hardware
1600 Apple /// Emulation Mode
1700 Apple /// Interfacing
1800 Apple Adventure
1900 Apple Bowl
2200 Apple Plot
2300 Apple Post
2400 Apple Stellar Invaders
2500 Apple Writer
2550 Apple Writer ///
2600 Applesoft
2700 Applesoft Firmware Card
2800 Artist Designer
2900 Auto-Start ROM
3100 Business Basic
3130 Business Graphics ///
3160 Business Graphics II
3200 Centronics Printer Interface
3300 Communications Interface
3700 DOS
3800 DOS Tool Kit
4100 Elementary, My Dear Apple
4200 Formulex
4300 FORTRAN
4600 Goodspell
4700 Graphics Tablet

(Continued)

APPLE TECH NOTES

Copyright (C) 1982 by Apple Computer, Inc.

"AppleACTNJuly82_1000-001-02" 106 KB 1962-10-11 dpi: 300h x 300v pix: 1994h x 2991v

APPLE COMPUTER TECHNICAL NOTES

Issued 2 Jul 82

General Information

Page 1000.001.03

4800 Hand Holding Basic
4900 High Speed Serial Interface
5100 Integer Basic
5200 Integer Basic Firmware Card
5500 Language Card
5600 Microchess 2.0
6000 Pascal Animation Tools
6100 Pascal
6150 Pascal ///
6200 Parallel Printer Interface
6400 Pilot
6600 Plan80
6650 Profile
6700 Programmer's Aid #1
6800 Psort
7100 The Shell Games
7200 Silentype
7250 Softcard ///
7275 Spelling Strategy
7300 Super Serial Card
7400 Supermap
7500 Tax Planner
7650 Universal Parallel Card
7800 Visicalc ///
8000 VT-100 Emulator
9300 Bulletin Board Systems
9600 Apple PROMs
9900 Vendor List

INTRODUCTION TO TECH NOTES

The Apple Answer Book has proved to be a very useful tool. It puts the answers to the most commonly asked questions at the hand of the people who get asked most often. The saga continued with the Apple Answer Book 2. It contains more information but in a format that's harder to use. The time has come for a book that needn't be replaced every six months. We developed "Tech Notes" as an easily expanded and updated answer to this need.

Tech Notes is organized on two levels. Each major subject or product is given its own number. Then each actual document has its own number. You are presently reading the second document under the major subject number 1000. The page number reflects this structure. This is page 1000.002.01. The 1000 refers to the major subject "General Information". The 002 refers to this document, "Introduction to Tech Notes". The last two digits refer to which page of the document this sheet is.

There is an index to all of the major subjects in document 1000.001. Each entry in this index will have an index to the individual documents in the subject in document number 000. Document 1000.000 contains two entries, the overall major subject index and this document.

When we add a document we will also send out replacements for any affected indexes. Any changes to the information to a document will result in mailing out a copy of the affected document with the changed material flagged by a "change bar" in the right margin of the changed lines. We will include information on how to keep your copy up to date.

APPLE COMPUTER TECHNICAL NOTES

Issued 24 May 82

Apple II Hardware

Page 1300.000.01

INDEX TO APPLE II HARDWARE

ALPHABETIC LISTING

Doc	Title
001	The Apple II Keyboard
003	Apple II vs Apple II Plus
005	Character Generator ROM
990	Errata - Apple II Reference Manual
002	The Mini-Assembler
006	Power Supply Input Frequency Limitations
004	Stopping the Blinking Cursor

NUMERIC LISTING

Doc	Issue Date	Previous Date	Pages
000	MAY 24 82	MAY 12 82	1
001	APR 26 82	OCT 1 81	1
002	SEP 28 81	-	1
003	OCT 1 81	-	2
004	SEP 28 81	-	1
005	MAY 24 82	-	1
006	MAY 24 82	-	1
990	APR 26 82	SEP 28 81	6

THE APPLE II KEYBOARD

The keyboard on the Apple II has an upper-case only encoder. It can produce ASCII characters 0 through 95 excluding `_`, `\`, `[`, `ctrl-_`, `ctrl-[`, and `ctrl-\`. The Apple /// uses the very same encoder. However, the Apple /// also has a second keyboard port which allows it to independently sense the shift, control, and alpha lock keys, as well as a number of other functions.

It is possible to modify an existing Apple II to allow it to sense whether or not the shift keys are depressed. This modification is supported by system software such as Pascal 1.1 and Pilot, as well as some word processing software. The core of the modification is a wire running from the shift keys to button input 2 of the game I/O port, which is unused by the paddles. This modification is not recommended for in-warranty Apples and should be only performed by an authorized service technician.

Since the Apple can now sense whether or not the shift key is being pressed, it can now distinguish 51 new characters (`ctrl-shift-A` thru `Z` excluding `M`, `N`, and `P`, `shift-A` thru `Z`, excluding `M`, `N`, and `P`, `shift-zero`, `shift-return`, `shift-ESC`, `shift-left arrow`, and `shift-right arrow`). This raises the total number of distinguishable characters to 141, which easily encompasses the 128 member ASCII character set.

It isn't unusual to rearrange the interpretation of these characters in software to facilitate the most natural use of the keyboard, so that it resembles a typewriter.

Apple has employed several types of keyboards on the Apple II since its introduction, so the modification can take on one of two appearances. On the older style keyboard, a wire should be run between pin 2 of the 74LS00 and pin 4 of the keyboard connector on the keyboard.

On the newer style keyboard with the "piggy-back" electronic assembly, the wire should be run between pin 9 of the 74LS00 closest to the keyboard connector and pin 4 of the keyboard connector.

In either case, a second wire should connect the bottom of pin 4 at location A7 to pin 7 at location J14 on the motherboard. Check carefully that the wire isn't connected to pin 7 of H14. The keyboard cable acts as a connection between these two wires.

A quicker but less flexible way to connect the "piggy-back" keyboards is to wire from pin 24 of the connector between the keyboard and the encoder board and pin 1 of IC H14. H14 is the IC directly in front of the Game I/O connector. Check carefully that the wire isn't connected to pin 7 of H14.

THE MINI-ASSEMBLER

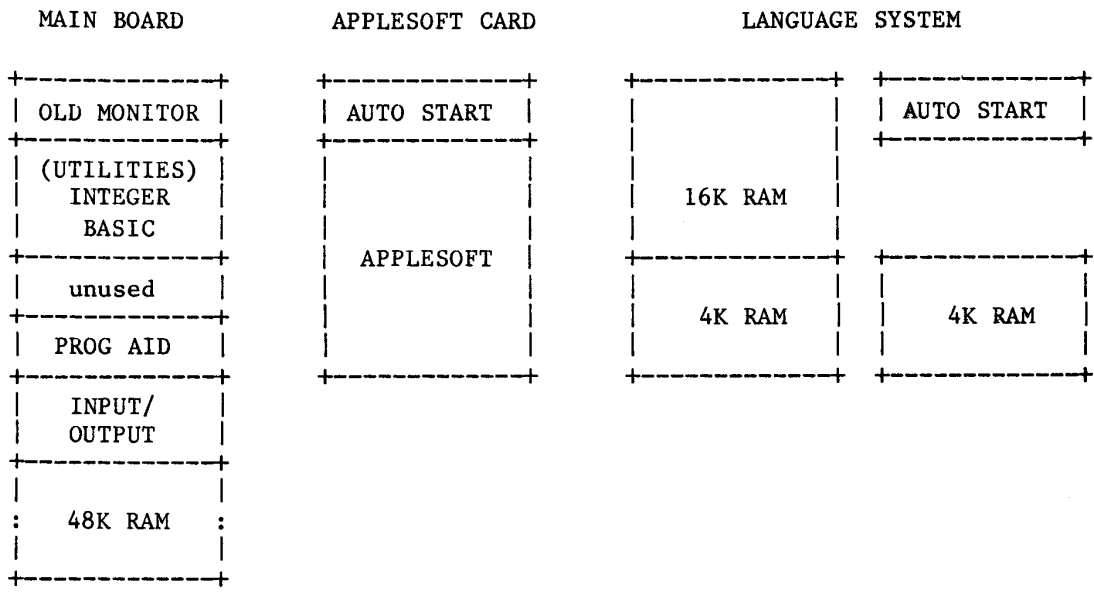
The Mini-Assembler is a convenient development tool for small programs. For major programming jobs it is suggested that you get a full sized assembler like EDASM in the DOS TOOL KIT. To use the Mini-Assembler you must first be in Integer BASIC. If you have an Apple II Plus then you will need a firmware card or a Language card. There are instructions for the Mini-Assembler on page 66 of the Apple II Reference Mnaual.

```
INT
CALL -151
F666G
```

APPLE II VS APPLE II PLUS

The only difference between the Apple II and the Apple II Plus is that the Apple II has Integer BASIC and the "old" monitor ROM while the Apple II Plus has Applesoft BASIC and the Autostart monitor ROM. Most of the game programs available today are written in Integer BASIC and most of the business, scientific, and industrial programs require Applesoft, so the selection depends on your application. Apple offers firmware cards that will supply Applesoft to Apple II owners and Integer Basic to Apple II Plus owners.

THE APPLE II



(Continued)

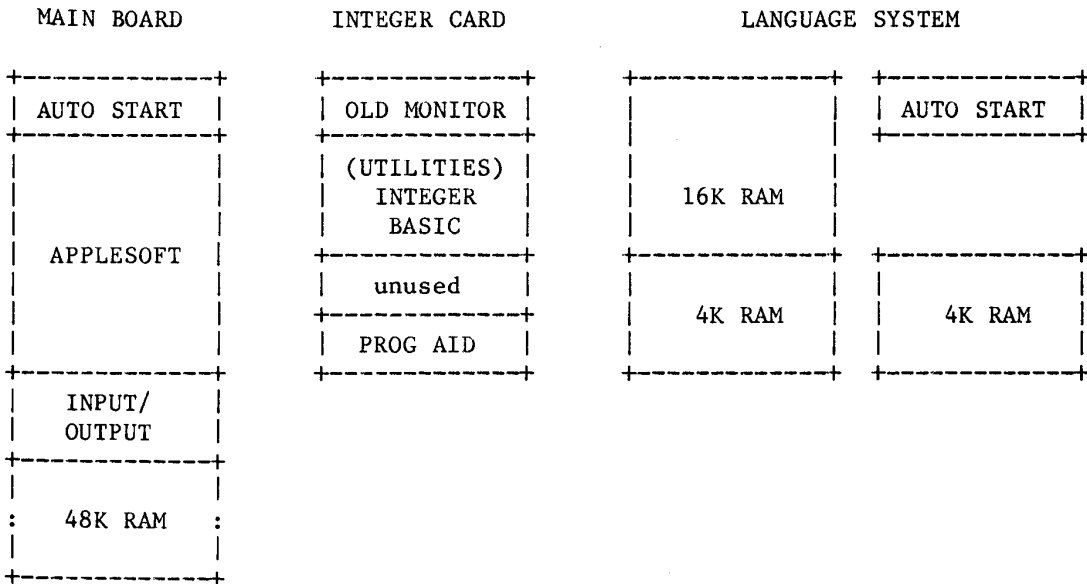
APPLE COMPUTER TECHNICAL NOTES

Page 1300.003.02

Apple II Hardware

Issued 1 Oct 81

THE APPLE II PLUS



Issued 28 Sep 81

Apple II Hardware

Page 1300.004.01

STOPPING THE BLINKING CURSOR

The blinking cursor on the Apple's screen is generated by hardware. You can stop the blinking and have a plain inverse cursor by lifting pin 13 of IC number B-13 and connect it to pin 7 of the same IC. Connecting pin 13 to pin 14 will remove the cursor entirely.

CHARACTER GENERATOR ROM

The character generator ROM in revision 7 and later Apples can be replaced with a user supplied 2716 PROM. This allows the user to program in special characters or lower case characters as required. This note describes how the characters are mapped in the ROM.

Characters are stored using eight bytes per character and are arranged in the ROM in the order shown in Table 7 on page 15 of the Apple II Reference Manual. Lower case characters, if desired, should be mapped in place of the numbers and punctuation in columns \$E0 and \$F0. The starting address for each character is the address from Table 7 multiplied by eight.

Each character is made up of eight bytes. Each byte represents one row of dots. The most significant bit of each byte is ignored by the hardware. The lowest addressed byte of each character is the top most row of dots of the displayed character. The first and last bits of each row of dots are usually set to zero to supply a two dot space between characters. The bottom row of dots is usually left set to zero to allow a one row space between lines.

Some lower case character sets use the bottom row for descenders (the letter "g", for example). This can cause overlap when the descender is directly above an upper case letter like "B". The following diagram shows how the characters are built. The three digit numbers are the hexadecimal ROM address for each byte and the two digit hexadecimal number is the pattern of on and off bits.

\$208	\$08	...*..	\$708	\$00	\$730	\$08	...*..
\$209	\$14	..*.*..	\$709	\$00	\$731	\$14	..*.*..
\$20A	\$22	.*...*	\$70A	\$1C	..***..	\$732	\$10	..*....
\$20B	\$22	* * *	\$70B	\$02*	\$733	\$10	..*....
\$20C	\$3E	*****	\$70C	\$1E	..*****	\$734	\$3E	*****
\$20D	\$22	.*...*	\$70D	\$22	.*...*	\$735	\$10	..*....
\$20E	\$22	.*...*	\$70E	\$1E	..*****	\$736	\$10	..*....
\$20F	\$00	\$70F	\$00	\$737	\$00
\$210	\$3C	*****	\$710	\$20	.*...*	\$738	\$00
\$211	\$22	.*...*	\$711	\$20	.*...*	\$739	\$00
\$212	\$22	.*...*	\$712	\$3C	*****	\$73A	\$1C	..***..
\$213	\$3C	*****	\$713	\$22	.*...*	\$73B	\$22	.*...*
\$214	\$22	.*...*	\$714	\$22	.*...*	\$73C	\$22	.*...*
\$215	\$22	.*...*	\$715	\$22	.*...*	\$73D	\$1E	..*****
\$216	\$3C	*****	\$716	\$3C	*****	\$73E	\$02*
\$217	\$00	\$717	\$00	\$73F	\$1C	..***..

Issued 24 May 82

Apple II Hardware

Page 1300.006.01

POWER SUPPLY INPUT FREQUENCY LIMITATIONS

The Apple II power supply is a switching power supply. It is designed to accept 107 to 135 volts from DC to 60 Hz. It will also work at up to 400 Hz but there is a danger that the circuit that protects the supply from short circuits will not work.

 Issued 26 Apr 82

Apple II Hardware

Page 1300.990.01

ERRATA - APPLE II REFERENCE MANUAL

A2L0001A
 030-004-01

Page 4

Due to continuing cost reductions on 16K RAMs, current revisions of the Apple II will accept only 16K RAMs.

Page 7

Table 2, the backspace key (<-) and the forward copy key (->) are reversed.

->	\$95	\$95	\$95	\$95
<-	\$88	\$88	\$88	\$88

Page 9

2nd Paragraph, the pins carrying the video signals are referred to as being on the left side of the board. They are on the RIGHT.

Page 10

The photograph refers to a Revision 6 Apple. Revision 7 and later Apples will look slightly different.

Page 10

The Eurapple modification is not complete and we do not support or recommend modification of Apples for European television signals.

Page 11

The photograph refers to a Revision 6 Apple. Revision 7 and later Apples will look slightly different.

Page 23

The photograph refers to a Revision 6 Apple. Revision 7 and later Apples will look slightly different.

Page 25

First paragraph, line 4; The address is actually \$C040 instead of \$C04F.

Page 31

Paragraph 3, line 3, "the leftmost column" should read "the rightmost column"

(Continued)

Page 31

Table 11 should read:

LEFT EDGE	32	\$20	0/ 0/39	\$0/\$ 0/\$27
WIDTH	33	\$21	0/39/39	\$0/\$27/\$27
TOP EDGE	34	\$22	0/ 0/23	\$0/\$ 0/\$17
BOTTOM EDGE	35	\$23	0/24/24	\$0/\$18/\$18

Page 35

ESC E "When COUT detects this" should read "When RDKEY detects this"

Page 36

The Autostart ROM initializes the annunciators 0 and 1 to OFF and annunciators 2 and 3 to ON.

Page 37

Paragraph 5 refers to to using call -1169 to set \$3F4 to XOR of \$3F3 in autostart reset vector. This may garbage the diskette in drive 1 if used on a non-autostart system.

Page 47

The line of monitor command just under the first paragraph should read

```
*0:FF FF AD 30 C0 88 D0 04 C6 01 F0 08
*:CA D0 F6 A6 00 4C 02 00 60
```

Page 70

Paragraph 2, the page 3 memory usage chart is actually on page 65 of the manual instead of page 62.

Page 70

RAM Configuration Blocks are not included on Revision 7 and later Apple boards.

Page 74

The Zero Page memory maps are incomplete. Applesoft also uses \$D6 and Applesoft HI-RES uses \$19 to \$1D.

(Continued)

Page 79 Table 22

The line for \$C060 should be

	\$0	\$1	\$2	\$3	\$4	\$5	\$6	\$7
\$C060	cin	pb0	pb1	pb2	gc0	gc1	gc2	gc3

Page 81

Paragraph 3 recommends IOSAVE and IORESTORE. These routines must be used with caution because if any other routine in the system uses them, they will overwrite your information. The 6502 stack is a better place to save the registers.

Page 84

Expansion ROM, paragraph 3, This flip-flop should be turned on by the I/O SELECT signal, not the DEV SEL signal.

Page 89

The photograph refers to a Revision 6 Apple. Revision 7 and later Apples will look slightly different.

Page 89

The pointer to the USER 1 jumper is wrong. See the photograph on page 99 for the correct location.

Page 90 Paragraph 5

RDY, RES, IRQ, NMI lines are held high by a 1000 ohm resistor, NOT 3300 ohm.

Page 91

Data from 6502 (read) and Data to 6502 (write) are reversed. They should be:

Data from 6502 (write)
Data to 6502 (read)

Page 96

Paragraph 4, line 5, the 74LS283 is at location E14.

(Continued)

Page 100

The Apple's new built-in keyboard is built around a AY-5-3600 keyboard encoder. The inputs to this ROM, on pins 17 through 26 and 36 through 40 are connected to the matrix of keyswitches on the keyboard. The outputs of this ROM are buffered by a 74LS04 and are connected to the Apple keyboard connector.

The keyboard decoder rapidly scans through the array of keys on the keyboard, looking for one which has been pressed. This scanning action is controlled by the free running oscillator made up of three sections of a 74LS00 at location B3 on the separate encoder board. The speed of this oscillation is controlled by C7, R7 and R8 on the encoder board.

Page 104

The +12 and -5 volt levels are documented on page 92 as +11.8 and -5.2. The levels will vary from Apple to Apple.

Page 107 Pin 19, SYNC, is connected only on Apples manufactured for sales overseas.

Page 107

Pin 21, RDY, is pulled high with a 1000 ohm resistor to +5 volts.

Page 107

Pin 22, DMA, is held high by a 1000 ohm resistor to +5 volts. This signal will stop the 6502 clock. It should not be held low for more than two clock cycles or the 6502 internal registers may be lost.

Page 108

Pin 28, INT IN, is the second item on the page and is mislabeled 26.

Page 108

Pin 32, INH, is pulled high by a 1000 ohm resistor.

Page 108

Pin 35, COLOR REF, is connected only on Apples manufactured for sales overseas.

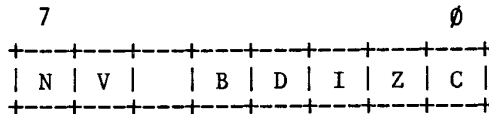
Page 119 Figure 3 should be labeled:

ROTATE ONE BIT RIGHT (MEMORY OR ACCUMULATOR) M or A.

(Continued)

Page 120

The Processor status word should be



PROCESSOR STATUS WORD, "P"

^
This bit is undefined.

Page 121

Note 1 should read "Bits 6 and 7 are transferred to the Status Register. If the result of A AND M is zero, then Z=1; otherwise Z=0."

Page 127-128

The unimplemented opcodes are shown as NOPs, which is wrong. \$EA is the only code defined as NOP. The others should not be used as they perform undefined operations.

Page 128

Op-code \$AD is a LDA, Absolute

Page 137

The addresses starting at line 100 should be:

```

CLRAN0 EQU $C058
SETAN0 EQU $C059
CLRAN1 EQU $C05A
SETAN1 EQU $C05B
CLRAN2 EQU $C05C
SETAN2 EQU $C05D
CLRAN3 EQU $C05E
SETAN3 EQU $C05F
    
```

Page 143

Starting at address \$FA6F the comments should read:

```

FA6F LDA CLRAN0 ;AN0 = TTL LO
FA72 LDA CLRAN1 ;AN1 = TTL LO
FA75 LDA SETAN2 ;AN2 = TTL HI
FA78 LDA SETAN3 ;AN3 = TTL HI
    
```

(Continued)

Page 165

The comment after address \$FCAC should read

$1.0204 \text{ USEC} * (13+27/2*A+5/2*A*A)$

Pages 172-176

These tables were cut up to fit the pages so they are no longer in numeric or alphabetic order.

 Issued 12 May 82

Apple II Interfacing

Page 1400.000.01

INDEX TO APPLE II INTERFACING

ALPHABETIC LISTING

Doc	Title
002	Cassette Tape Loading Problems
001	The Apple II Cassette Interface

NUMERIC LISTING

Doc	Issue Date	Previous Date	Pages
000	MAY 12 82	SEP 30 81	1
001	SEP 25 81	-	4
002	SEP 21 81	-	1

THE APPLE II CASSETTE INTERFACE

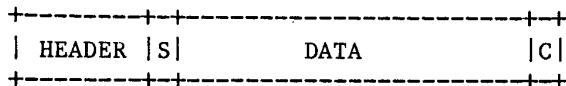
INTRODUCTION:

This note is to explain the cassette interface built into the Apple II and Apple II plus. Included is information on the format and use of the read and write subroutines. It is assumed in this note that the cassette recorder is in the proper mode, play or record, when the read and write routines are executed. Also note that the timing is approximate and may vary from one Apple to another.

RECORDS:

A record is a block of binary data. This data may be a BASIC or APPLESOFT program, a machine language program, or just binary data. Records representing BASIC or APPLESOFT programs are really two records, the length of the program and the actual program. A record consists of a header, sync bit, the actual data, and a checksum byte for error detection.

MONITOR RECORD FORMAT



BASIC PROGRAM RECORD FORMAT



KEY: S = SYNC BIT
 C = CHECKSUM BYTE
 LB = BASIC PROGRAM LENGTH

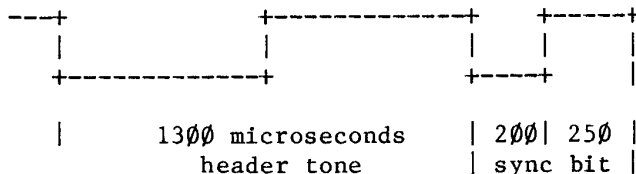
THE HEADER:

The header consists of 10 seconds of 770 Hz tone, (1 cycle equals 1300 microseconds). This is enough time for the cassette motor to get up to speed and the plastic tape leader to go by. There is also a shortened header between the BASIC length bytes and the BASIC program itself. This header is generated by a subroutine called HEADR. The value of the accumulator on entry controls the length of the header tone. This can vary from 0.2 seconds to 40 seconds. On entry the X register should be 0 and

(Continued)

the carry flag should be set. HEADR also generates a sync bit at the end of the tone. HEADR resides at hexadecimal address \$FCC9, or decimal address -882.

THE LAST CYCLE OF HEADER TONE AND SYNC BIT



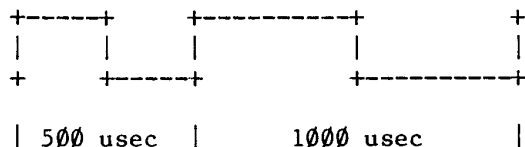
THE SYNC BIT:

The sync bit is one half cycle of 2500 Hz, (200 microseconds) and one half cycle of 2000 Hz, (250 microseconds). It is used to signal the end of the header tone and the start of the data. It is generated by HEADR.

THE DATA:

The data is recorded on the tape with a low starting address and a high ending address. Each byte of data is shifted out most significant bit first, least significant bit last. A zero bit is made up of one cycle of 2 kHz, (250 microseconds per half cycle) and a one bit is one cycle of 1 kHz, (500 microseconds per half cycle). This works out to 2000 baud for zeros only and 1000 baud for ones, or an average of 1500 baud.

A ZERO BIT AND A ONE BIT



THE CHECKSUM:

All during reading or writing each data byte is EXCLUSIVE OR-ed with a checksum byte. This byte is written on the tape at the end of the data block. If the checksum computed during a read agrees with the checksum that was written out then the data is probably good. This method will detect an odd number of errors for any of the eight bits of the byte.

(Continued)

WRITING DATA:

The cassette output circuitry is quite simple. It is a flip-flop connected through a voltage divider to the jack on the back panel of the Apple. Any time the address \$C020 is accessed this flip-flop changes state. Accessing the flip-flop once every 500 microseconds generates a 1000 Hz tone.

READING DATA:

The cassette input circuit is more complicated. It consists of a 741 operational amplifier configured as a zero crossing detector. That means that whenever the voltage at the input jack goes from positive to negative (or negative to positive) the output of the amplifier switches from a 1 to a 0 (or 0 to 1). The detector is accessed by any read to address \$C060. The sign bit (most significant bit) of the byte read reflects the detector status. The read routines continually EXCLUSIVE ORs this bit with the value most recently read to detect a change in state. The amount of time required to change state indicates the incoming frequency which then is used to determine if a one or a zero has been received. After detecting the first zero crossing at the start of the header, the read routine uses HEADR to generate a 3.5 delay then waits for the sync bit. It then reads the data and puts it in the specified memory range.

USING THE CASSETTE INTERFACE:

To either read or write all you need do is specify an address range and execute the read or write subroutine. The address range is stored in four bytes, two for the first address to be saved and two for the last to be saved. In both cases the least significant byte is first.

FROM THE MONITOR:

If the start is \$800 and the end is \$9FF then

```
800.9FFW will write the data to the cassette and
800.9FFR will retrieve it.
```

(Continued)

FROM MACHINE LANGUAGE:

Again if the start is \$800 and the end is \$9FF then store the address range,

```
LDA #$00
STA $3C    starting address low
LDA #$08
STA $3D    starting address high
LDA #$FF
STA $3E    ending address low
LDA #$09
STA $3F    ending address high
JSR $FEDC  write to block to tape
```

The JSR \$FEDC will write the data to the cassette or JSR \$FEFD can be used to read from the cassette.

FROM BASIC:

First set up the address range. If S = the start and E = the end then from integer BASIC,

```
POKE 60,S MOD 256
POKE 61,S / 256
POKE 62,E MOD 256
POKE 63,E / 256
```

or from APPLESOFT,

```
POKE 60,S - INT(S / 256) * 256
POKE 61,S / 256
POKE 62,E - INT(E / 256) * 256
POKE 63,E / 256
```

Then to write out to cassette CALL -307 or to read in from the cassette CALL -259.

CASSETTE TAPE LOADING PROBLEMS

There are several possible reasons for a cassette tape load to fail. The following list is presented in the order of the most probable cause.

1. Try adjusting the volume and tone controls on the recorder.
2. Try moving the recorder at least 18 inches away from the TV.
3. Try another cassette to check if yours is defective.
4. Try another recorder to see if yours is compatible and aligned.
5. Try another Apple to see if yours is OK.
6. Try cleaning the recorder head.
7. Try unplugging the cable to the MIC jack.

INDEX TO APPLE /// HARDWARE

ALPHABETIC LISTING

Doc	Title
003	Apple /// and Game Paddles
001	Apple /// Memory Diagnostic Display
002	Apple /// Video
990	Errata - Apple /// Owner's Guide
004	Joystick Input - Apple II game Incompatibility
005	Joystick Input - Apple II Game Controller Incompatibility

NUMERIC LISTING

Doc	Issue Date	Previous Date	Pages
000	MAY 12 82	SEP 30 81	1
001	SEP 25 81	-	1
002	SEP 18 81	-	1
003	OCT 21 81	-	1
004	MAY 24 82	-	1
005	MAY 24 82	-	1
990	APR 26 82	SEP 28 81	1

APPLE /// MEMORY DIAGNOSTIC DISPLAY

The Apple /// performs a memory check whenever it's turned on. Normally the display doesn't stay on the screen long enough to be noticed. However, if you have a memory error you will get an array of periods with an inverse "1" to indicate the faulty bit. The memory diagnostic can also be invoked by <control> <open-apple> <Reset> and >F6E6G

Row	Bits
7	b7 b6 b5 b4 b3 b2 b1 b0
6	b7 b6 b5 b4 b3 b2 b1 b0
5	b7 b6 b5 b4 b3 b2 b1 b0
4	b7 b6 b5 b4 b3 b2 b1 b0
3	b7 b6 b5 b4 b3 b2 b1 b0
2	b7 b6 b5 b4 b3 b2 b1 b0
1	b7 b6 b5 b4 b3 b2 b1 b0
0	b7 b6 b5 b4 b3 b2 b1 b0

This information can be decoded with the following table to indicate which chip needs replacement.

MAP OF THE MEMORY BOARD

Board Ref	Row	Chips	Row	Chips
D	1	b7 b6 b5 b4 b3 b2 b1 b0	2	b7 b6 b5 b4 b3 b2 b1 b0
C	3	b7 b6 b5 b4 b3 b2 b1 b0	0	b7 b6 b5 b4 b3 b2 b1 b0
B	4	b7 b6 b5 b4 b3 b2 b1 b0	5	b7 b6 b5 b4 b3 b2 b1 b0
B	6	b7 b6 b5 b4 b3 b2 b1 b0	7	b7 b6 b5 b4 b3 b2 b1 b0

For example, if your display looked like this;

```

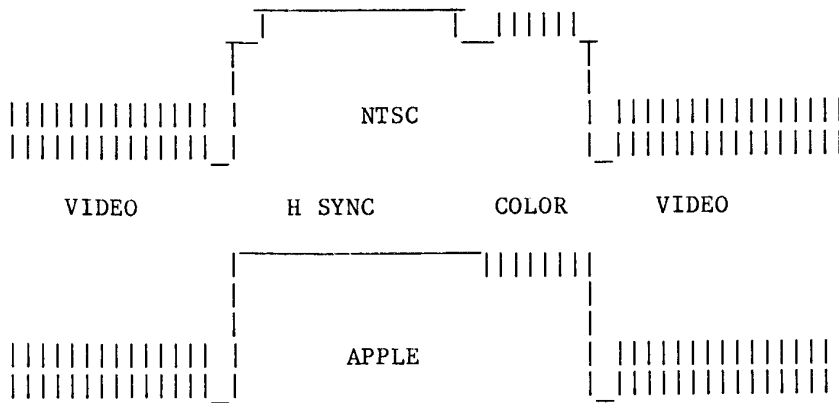
. . . . .
. . . 1 . . .
. . . . .
. . . . .
. . . . .
. . . . .
. . . . .
. . . . .
. . . . .

```

you would need to replace the fourth chip from the left end of row "B". This row appears double width because it has 32K RAMs.

APPLE /// VIDEO

The Apple ///'s horizontal sync consists of 8usec of sync and 4usec of color burst. There are no porches or breezeways. There is a certain amount of 3.58 MHz during a video line due to the switching of the pixels.



JOYSTICK INPUT - APPLE II GAME INCOMPATIBILITY

Some machine language games that use the paddle inputs will not work on the Apple ///. The hardware for reading the analog inputs is different and some software will not be able to take this into account.

Also, the joystick ports on the back of the Apple /// are arranged such that some games won't work properly in Emulation mode. Most joystick oriented games will use PDL (0) and PDL (1) for X-Y control. The ports on the Apple /// put these signals on different connectors. Here are the Apple II Emulation Mode equivalents of the signals available on ports A and B.

pin	Port A	Port B
4	PDL (0)	PDL (1)
5	PB1	PB2
8	PDL (2)	PDL (3)
9	PB3	

Issued 24 May 82

Apple /// Hardware

Page 1500.005.01

JOYSTICK INPUT - APPLE II GAME CONTROLLER INCOMPATIBILITY

Apple II paddles will not work when directly wired to the Apple /// joystick ports. Please refer to the Apple /// Owner's Guide, pages 128 through 130 for the proper schematic. The preferred value for the variable resistors is 2000 ohms.

Issued 26 Apr 82

Apple /// Hardware

Page 1500.990.01

ERRATA - APPLE /// OWNER'S GUIDE
A3L0001
030-0121-00

|

Page 80

Pin number 9 of port A is really SW3.

 Issued 12 May 82

Apple /// Emulation Mode

Page 1600.000.01

INDEX TO APPLE /// EMULATION MODE

ALPHABETIC LISTING

Doc	Title
002	Apple /// and Apple Plot
003	Apple /// Disk Drives and Emulation Mode
001	Printing from Emulation Mode

NUMERIC LISTING

Doc	Issue Date	Previous Date	Pages
000	MAY 12 82	SEP 30 81	1
001	SEP 30 81	-	1
002	SEP 25 81	-	1
003	APR 26 82	SEP 30 81	1

PRINTING FROM EMULATION MODE

The first version of Apple II emulation mode for the Apple /// simulates a Communications card in slot 7 that uses the port C RS-232 hardware. This simulation doesn't drive printers very well. The machine language routine below will properly drive any printer from emulation mode that can be driven from Business Basic.

CALL 768 will initialize the routine and connect it to either Basic through DOS. PRINT CHR\$(4);"PR#0" will return output to the screen.

The replacement value for x, y, and z are from the Apple /// Standard Device Drivers Manual on pages 82 and 83:

x = the low digit from the table on page 83
 y = the high digit from the table on page 83
 z = the low digit from the table on page 82

```

300:A9 19          LDA #$19          ;SET UP I/O VECTORS
302:85 36          STA $36
304:A9 03          LDA #$03
306:85 37          STA $37
308:20 EA 03      JSR $03EA
30B:8D F1 C0      STA $C0F1          ;RESET ACIA
30E:A9 2B          LDA #$2B          ;$xB, odd parity
310:8D F2 C0      STA $C0F2
313:A9 28          LDA #$28          ;$yz, 7 bits, 1200 baud
315:8D F3 C0      STA $C0F3
318:60           RTS
319:48           ENTER PHA
31A:AD F1 C0      LOOP  LDA $C0F1
31D:49 10          EOR #$10
31F:29 70          AND  #$70
321:D0 F7          BNE  LOOP
323:68           PLA
324:8D F0 C0      STA $C0F0
327:C9 8D          CMP  #$8D
329:D0 07          BNE  OUT
32B:48           PHA
32C:A9 8A          LDA  #$8A
32E:20 17 03      JSR  ENTER
331:68           PLA
332:60           OUT   RTS
    
```

Issued 30 Sep 81

Apple /// Emulation Mode

Page 1600.002.01

APPLE /// DISK DRIVES AND EMULATION MODE

The Apple II emulation mode on the Apple /// is configured so that the internal drive looks like Slot 6, Drive 1 and the first external drive looks like Slot 6, Drive 2. Emulation mode can't access the third nor the fourth external drive.

Issued 26 Apr 82

Apple /// Emulation Mode

Page 1600.003.01

APPLE /// DISK DRIVES AND EMULATION MODE

The Apple II emulation mode on the Apple /// is configured so that the internal drive looks like Slot 6, Drive 1 and the first external drive looks like Slot 6, Drive 2. Emulation mode can't access the third or the fourth external drive. |

INDEX TO APPLE /// INTERFACING

ALPHABETIC LISTING

Doc	Title
001	Interfacing the Apple /// to Serial Printers
002	Interrupts and RESET in the Apple /// Peripheral Slots
003	Problems with the Apple II Parallel Printer Interface
005	.RS232 - Number of Stop Bits
004	Using the Super Serial Card in the Apple ///

NUMERIC LISTING

Doc	Issue Date	Previous Date	Pages
000	MAY 12 82	SEP 30 81	1
001	APR 26 82	OCT 29 81	4
002	SEP 25 81	-	1
003	NOV 17 81	-	1
004	NOV 17 81	-	1
005	NOV 17 81	-	1

INTERFACING THE APPLE /// TO SERIAL PRINTERS

The Apple /// is a terminal type RS-232 device as are most serial printers. This means that some of the wires in the cable between the Apple /// and the printer need to be changed for proper communications and handshaking to occur. There is a short cable adapter included with the Apple /// for connection to a Qume Sprint 5 printer (Apple Part Number 590-0029). However, other printers may have other requirements.

The Table 1 and Table 2 on the following pages give the cable adapter wiring diagrams for some popular printers. Scan Table 1 until you find the brand and model of your printer. The number in the "DIAGRAM" column indicates which wiring diagram in Table 2 you need. The "NOTES" column in Table 1 gives additional information for proper interfacing. Wire up the adapter as shown and install it between the Apple /// and the printer.

Included below is the specification for the Apple /// RS-232 interface in case your printer isn't mentioned.

Pin	Name	Function	Notes
1	GND	Frame ground	
2	TxD	Transmit Data	Data output to printer
3	RxD	Receive Data	Not used by printer driver
4	-RTS	Request To Send	Handshake output to printer
5	-CTS	Clear To Send	Do not use
6	-DSR	Data Set Ready	Handshake input from printer
7	GND	Signal ground	
8	-DCD	Data Carrier Detect	Handshake input from printer
20	-DTR	Data Terminal Ready	Handshake output to printer

The "-" before the name of a signal indicates that it is a negative true signal. These signals are considered active if the signal is above +3 volts at the RS-232 connector and inactive if less than -3 volts.

Use of -CTS is discouraged because it directly controls the Apple's data transmitter and may stop transmission in the middle of a character. The Apple /// printer driver sets up -RTS and -DTR to alert the printer and checks both -DSR and -DCD for printer ready information before transmitting any characters. All handshake input lines are pulled to an enabled condition on the Apple /// if the printer doesn't use them.

The first version of Apple II Emulation mode that was shipped did not check -DSR or -DCD before transmitting.

(Continued)

APPLE COMPUTER TECHNICAL NOTES

Page 1700.001.02

Apple /// Interfacing

Issued 26 Apr 82

TABLE 1

PRINTER	DIAGRAM	NOTES
ANADEX DP-9500	5	CHECK PARITY, BAUD, STOP BITS, ETC
DP-9501	5	CHECK PARITY, BAUD, STOP BITS, ETC
ANDERSON JACOBSON 832	A	
833	A	
841	A	INSTALL JUMPER AT LOCATION 'F'
AXIOM EX-801	8	
EX-820	8	
CENTRONICS 737	3	
C-ITOH	8	CHECK PRINTER OPTION JUMPERS
COMPRINT 912	8	CHECK PRINTER OPTION JUMPERS
DEC LA34	8	SET DRIVER TO 300 BAUD
LA120	A	
DIABLO 630	3	
EPSON MX-70	2	USE AT 9600 BAUD
MX-80	2	USE AT 9600 BAUD
HEATHKIT H-14	1	
IDS 125	7	
225	7	
440	7	
445	7	
460	7	
560	7	
NEC 5510	6	
5515	6	
5520	6	
5525	6	
OKIDATA 82A,	2	Set switches
OKIDATA 83A,	2	Set switches
		Dip Switches: 87654321
		Front 10010000 1=ON
		Back 101011 0=OFF
		Jumper Blocks: Position A
QUME SPRINT 5	A	

(Continued)

APPLE TECH NOTES

Copyright (C) 1982 by Apple Computer, Inc.

"Apple/ACTN/July82_1700-001-02" 87 KB 1962-10-11 dpi: 300h x 300v pix: 2000h x 2991v

TELETYPE 43 8
 TI 810 3

TABLE 2 - Cabling Diagrams

1. Apple /// Printer
 1 ----- 1
 2 ----- 3
 6 ----- 4
 7 ----- 7

2. Apple /// Printer
 1 ----- 1
 2 ----- 3
 6 ----- 11
 7 ----- 7

3. Apple /// Printer
 1 ----- 1
 2 ----- 3
 4 ----- 6 & 8
 6 ----- 11
 7 ----- 7

4. Apple /// Printer
 1 ----- 1
 2 ----- 3
 6 ----- 14
 7 ----- 7

5. Apple /// Printer
 1 ----- 1
 2 ----- 3
 6 ----- 19
 7 ----- 7

6. Apple /// Printer
 1 ----- 1
 2 ----- 3
 4 ----- 5
 6 ----- 19
 7 ----- 7
 20 ----- 6 & 8

(Continued)

7. Apple /// Printer
 2 ----- 3
 6 ----- 20
 7 ----- 7

8. Apple /// Printer
 1 ----- 1
 2 ----- 3
 6 ----- 20
 7 ----- 7

A. APPLE SUPPLIED CABLE
 Apple /// Printer
 1 ----- 1
 7 ----- 7
 2 ----- 3
 3 ----- 2
 4 & 5 ----- 8
 8 ----- 4 & 5
 6 ----- 20
 20 ----- 6

Issued 25 Sep 81

Apple /// Interfacing

Page 1700.002.01

INTERRUPTS AND RESET ON THE APPLE /// PERIPHERAL SLOTS

The Apple /// slots can not respond to IRQ or RESET from the peripheral cards in emulation mode. There is no way around it because the hardware is different. NMI is not on the same pin as in the Apple II and will act like RESET on the Apple II.

PROBLEMS WITH THE APPLE II PARALLEL PRINTER INTERFACE

There is a slight timing problem when the Apple II Parallel Printer Interface is used in an Apple ///. The best way to solve this is to get a Universal Parallel Interface which will work in both Apple /// mode and Apple II emulation mode. The Apple II Parallel Printer Interface can be made to work with the addition of a 0.001 microfarad capacitor between pins 4 and 7 of chip B2, a 74LS74.

USING THE SUPER SERIAL CARD IN THE APPLE ///

The Super Serial Card can be used in the Apple /// with the appropriate driver. But, since the Apple /// is an interrupt driven system, switches S2-6 and S2-7 must be ON to enable interrupts from the card. The Apple /// will hang when the driver is opened if this is not done.

Issued 17 Nov 81

Apple /// Interfacing

Page 1700.005.01

.RS232 - NUMBER OF STOP BITS

There is no way to control the number of stop bits sent out by the .RS232 driver. It will send out one stop bit except at 110 baud when it sends out two.

Issued 12 May 82

Apple Adventure

Page 1800.000.01

INDEX TO APPLE ADVENTURE

ALPHABETIC LISTING

Doc	Title
001	Booting With DOS 3.3

NUMERIC LISTING

Doc	Issue Date	Previous Date	Pages
000	MAY 12 82	OCT 28 81	1
001	OCT 28 81	-	1

Issued 28 Oct 81

Apple Adventure

Page 1800.001.01

BOOTING WITH DOS 3.3

Apple Adventure is shipped in a protected DOS 3.2 format. If you want to save a game to continue play later, you will need a DOS 3.2 formatted diskette. Please ask your dealer to INITIALize a diskette with DOS 3.2.

You can initialize more 3.2 diskettes by booting with BASICS and the DOS 3.2 diskette that the dealer initialized for you, inserting a blank diskette in the drive, and typing INIT HELLO.

The MUFFIN program won't work on Adventure because of its software protection scheme.

Issued 12 May 82

Apple Bowl

Page 1900.000.01

INDEX TO APPLE BOWL

ALPHABETIC LISTING

Doc	Title
001	>32768 ERROR

NUMERIC LISTING

Doc	Issue Date	Previous Date	Pages
000	MAY 12 82	OCT 28 81	1
001	OCT 28 81	-	1

Issued 28 Oct 81

Apple Bowl

Page 1900.001.01

>32768 ERROR

If the Apple's memory initializes to \$80 in the variable storage area then the program will stop at line 1620 with a '>32768 ERROR'. Attempting to PRINT K(10) will give the same error.

Try running another program first to initialize that area of memory.

Issued 12 May 82

Apple Plot

Page 2200.000.01

INDEX TO APPLE PLOT

ALPHABETIC LISTING

Doc	Title
990	Errata - Apple Plot Manual

NUMERIC LISTING

Doc	Issue Date	Previous Date	Pages
000	MAY 12 82	SEP 30 81	1
990	APR 26 82	SEP 25 81	2

Issued 26 Apr 82

Apple Plot

Page 2200.990.01

ERRATA - APPLE PLOT MANUAL
A2L0037
030-0116-00

Page 9

The data for day 17 should be "63 1/2" instead of "63 11/2".

Page 23

Paragraph 5, line 1, "Apple Plot will plot one bat" should be "one bar".

Page 26

To overlay graphs the user must toggle the overlay graph format, recall data from the diskette, and DISPLAY the graph to get the third, fourth, etc. plots to overlay the first and second plots.

Page 36 and 38

Using either the DX or DY commands from the edit/entry section of Apple Plot will result in deletion of both the X and Y coordinates, as opposed to just the X coordinate or the Y coordinate depending on the command used.

Page 43

Paragraph 3 line 9 reads "...be, aand..." and should read "...be, and..."

Page 46

Silentype paragraph. The word backplane is nebulous and inaccurate if certain assumptions are made. Slot 1 is on the motherboard. Use the term "motherboard" instead of backplane.

Page 47

Paragraph 3 line 1 reads "...type 1 to select the Qume..." and should read "...type 2 to select the Qume..."

Page 53

The Visicalc instructions should contain a warning that columns containing labels (or non numeric information) will become points of zero value in the data of the Apple Plot file.

Page 53 and 54

The description for doing a Visicalc file conversion is unclear. Even when a user creates a file using /PD as described, he cannot have a full 100 data points. The reason for this is that Apple Plot's Visicalc file

(Continued)

APPLE TECH NOTES

Copyright (C) 1982 by Apple Computer, Inc.

"AppleIACTNJuly82_2200-990-01" 129 KB 1962-10-11 dpi: 300h x 300v pix: 1969h x 2991v

converter is reading in strings, these may be less than 255 characters long and will be truncated after 239 characters have been read. The result is that if a default column width of 9 is used then the maximum number of points that can be converted is about 26.

In general the Apple Plot manual should have stated that error messages in the program are Applesoft error codes so that users can reference the proper manual to find out what went wrong.

APPLE COMPUTER TECHNICAL NOTES

Issued 24 May 82

Apple Post

Page 2300.000.01

INDEX TO APPLE POST

ALPHABETIC LISTING

Doc	Title
001	Page Length on Printout

NUMERIC LISTING

Doc	Issue Date	Previous Date	Pages
000	MAY 24 82	-	1
001	MAY 24 82	-	1

Issued 24 May 82

Apple Post

Page 2300.001.01

PAGE LENGTH ON PRINTOUT

The page length parameter in the configuration menu is never used by Apple Post. It will always assume 60 lines per page.

 Issued 12 May 82

Apple Stellar Invaders

Page 2400.000.01

INDEX TO APPLE STELLAR INVADERS

ALPHABETIC LISTING

Doc	Title
001	Booting Stellar Invaders Under DOS 3.3
002	Stellar Invaders and 80 Column Video Cards

NUMERIC LISTING

Doc	Issue Date	Previous Date	Pages
000	MAY 12 82	SEP 30 81	1
001	SEP 25 81	-	1
002	SEP 17 81	-	1

Issued 25 Sep 81

Apple Stellar Invaders

Page 2400.001.01

BOOTING STELLAR INVADERS UNDER DOS 3.3

Apple Stellar Invaders is a Pascal diskette set up to boot on a 13 sector, DOS 3.2, Apple II. You must use the BASICS: diskette to get it to boot on a 16 sector, DOS 3.3, system. The BOOT13 program on the DOS 3.3 diskette won't work.

Issued 17 Sep 81

Apple Stellar Invaders

Page 2400.002.01

STELLAR INVADERS AND 80 COLUMN VIDEO CARDS

Apple Stellar Invaders will not work properly if you have a 80 column card like the Smarterm or a communication card in slot 3. It is a Pascal format diskette. It will turn on the Smarterm card and direct the text output to the normal Apple screen.

INDEX TO APPLE WRITER

ALPHABETIC LISTING

Doc	Title
010	Apple Writer and Carriage Returns
009	Apple Writer Modification for Lower Case Display
011	Cassette File Storage
002	Com Card Print Routine
007	Convert Apple Writer Files to DOS Text Files
008	Convert DOS Text Files to Apple Writer Files
001	Data Format
004	Parallel Printer Interface
006	Re-starting Apple Writer
003	Serial Card Handshake Routine
005	Transmitting Files Through the Communication Card

NUMERIC LISTING

Doc	Issue Date	Previous Date	Pages
000	MAY 24 82	MAY 12 82	1
001	SEP 28 81	-	1
002	SEP 28 81	-	1
003	SEP 28 81	-	1
004	SEP 28 81	-	1
005	SEP 28 81	-	1
006	SEP 28 81	-	1
007	SEP 25 81	-	2
008	SEP 28 81	-	1
009	SEP 21 81	-	2
010	SEP 30 81	-	1
011	MAY 24 82	-	4

DATA FORMAT

The characters in Apple Writer are not stored in either true ASCII or the Apple's version of ASCII.

			Most significant 4 bits						
	0	2	4	6	8	A	C	E	
ASCII	cc	n	UC	lc					
APPLE					cc	n	UC	lc	
APPLE WRITER	UC	n			cc		lc	n	

cc control character
 n numbers
 UC Upper case
 lc lower case

Apple Writer stores its information in memory starting at location \$1901. LOMEM (\$1900) and HIMEM (\$95FF) must contain \$83 which marks the limits of memory. While editing, everything above the cursor on the screen starts at LOMEM and is topped off by a \$60 and everything below the cursor on the screen is pushed up against HIMEM with another \$60 below it. So the file is defined by the two end of file markers (\$83) with a hole in the middle defined by the two cursors (\$60).

As the cursor on the screen is moved through a file, one character at a time is being moved from one \$60 marker in memory to the other. The distance between the two markers is the remaining free memory.

COM CARD PRINT ROUTINE

The COM CARD PRINTER DRIVER needs to be loaded and initialized each time you enter the PRINT module. Here is a short program that will load the printer driver, initialize it, and enter the print routine.

```
FP
10 PRINT CHR$(4);"BLOAD PRINT ROUTINE"
20 CALL 775
30 PRINT CHR$(4);"PR#0"
40 PRINT CHR$(4);"BRUN PRINTER"
SAVE PRINTER SETUP
```

And here's how to use it

Q	Exit Apple Writer
RUN PRINTER SETUP	Run the set-up program
L	Load the file (if necessary)
P	Print new document
J	Change the printer address
363	To the print routine
<RETURN>	Print the file

SERIAL CARD HANDSHAKE ROUTINE

The SERIAL CARD HANDSHAKE ROUTINE needs to be loaded and initialized each time you enter the PRINT module. Here is a short program that will load the print routine, initialize it, and enter the print mode.

```

FP
10 PRINT CHR$(4);"BLOAD PRINT ROUTINE"
20 CALL 944
30 PRINT CHR$(4);"PR#0"
40 PRINT CHR$(4);"BRUN PRINTER"
SAVE PRINTER SETUP
    
```

And here's how to use it

Q	Exit Apple Writer
RUN PRINTER SETUP	Run the set-up program
L	Load the file (if necessary)
P	Print new document
J	Change the printer address
3C5	To the print routine
<RETURN>	Print it

Issued 28 Sep 81

Apple Writer

Page 2500.004.01

PARALLEL PRINTER INTERFACE

There are instructions starting on page 62 of the Apple Writer manual on how to use Apple Writer with an Apple Parallel Printer Interface. However, they are a bit clumsy. Here is how to modify the HELLO program on your Apple Writer diskette so that it will initialize the printer automatically.

```
LOAD HELLO
1 PRINT CHR$(4);"PR#1"
2 PRINT CHR$(9);"132N"
3 PRINT CHR$(4);"PR#0"
UNLOCK HELLO
SAVE HELLO
LOCK HELLO
```

When you print a file be sure that option J, PRINTER ADDRESS, contains \$C102.

TRANSMITTING FILES THROUGH THE COMMUNICATIONS CARD

It's easy to send your Apple Writer files out to another computer with the Communications card. First, boot Apple Writer and load the file that you want to send. Select the "P" for print option. When you get to the PRINT menu, select "Q" for quit. Now get into terminal mode by typing:

```
IN#2 <RETURN>      assuming that you're in slot 2
control-A control-F  set up terminal mode
```

OK, now you're ready to sign on and get the other computer ready to accept the information. When the other computer is ready to go, type:

```
control-A control-X  To exit terminal mode
CALL 2051             To re-enter Apple Writer
P                    To print a new document
J                    Change printer address to
C205                 Aim it at the Com Card
```

Then proceed to print as usual and the information will get to the other computer. Quit Apple Writer and do the IN#2 sequence again to do your sign-off.

The Communications card doesn't send out linefeeds with its carriage returns, so this probably won't work with serial printers.

Issued 28 Sep 81

Apple Writer

Page 2500.006.01

RE-STARTING APPLE WRITER

You can re-start the Apple Writer Editor or Printer at any time without destroying the file in memory with CALL 2051 from either Basic or 803G from the monitor.

An Apple Writer slave diskette can be booted without affecting the current file in memory. See the DOS Manual, pages 13 and 141, for more information about slave diskettes.

CONVERT APPLE WRITER FILES TO DOS TEXT FILES

The following program will convert an Apple Writer file into a DOS text file. It doesn't do any formatting or insert any carriage returns into the text file. This means that long paragraphs won't load with INPUT statements in Basic.

```
10 D$ = CHR$(4)
20 HOME
30 ONERR GOTO 400
40 PRINT D$"MON 0"
100 INPUT "APPLE WRITER FILE NAME? TEXT. ";A$
110 PRINT D$"BLOAD TEXT."A$
120 PRINT : PRINT A$" LOADED"
130 PRINT : INPUT "DESTINATION? ";T$
140 AD = 6401
150 PRINT D$"OPEN "T$
160 PRINT D$"WRITE "T$
170 A = PEEK (AD)
180 AD = AD + 1
190 IF A > = 224 THEN A = A - 64
200 IF A > = 192 THEN A = A + 32
210 IF A < 32 THEN A = A + 192
220 IF A < 64 THEN A = A + 128
230 IF A = 96 THEN 400
240 PRINT CHR$(A);
250 GOTO 170
400 PRINT : PRINT D$"CLOSE"
410 PRINT "DONE"
```

(Continued)

The resultant text file needs to be broken up to work on it from Basic. The following program will read a DOS sequential text file and send its contents to the interface in slot 1. It is careful not to strip off any leading spaces.

```
1000 ONERR GOTO 2000
1010 LET D$ = CHR$ (4)
1020 INPUT "FILE NAME? ";F$
1030 PRINT D$;"OPEN ";F$
1040 PRINT D$;"READ ";F$
1050 REM      MAIN READ-PRINT LOOP
1060 GET A$
1070 PR# 1
1080 PRINT A$;
1090 GOTO 1060
2000 REM      FILE DONE, FINISH UP
2010 CALL 1002
2020 PRINT
2030 PRINT D$;"CLOSE"
2040 END
```


CONVERT DOS TEXT FILES TO APPLE WRITER FILES

The following program will convert a DOS text file to Apple Writer format and saves it to the disk. The destination file name will be the name you gave to the first question with "TEXT." added to the front.

```
100 LET D$ = CHR$ (4)
110 ONERR GOTO 380
120 HOME
130 FOR I = 768 TO 808
140 READ J: POKE I,J: NEXT : RETURN
150 DATA 169,1,133,30,169,25
160 DATA 133,31,32,12,253,160
170 DATA 0,9,128,201,141,240,12
180 DATA 201,160,144,241,201
190 DATA 224,144,2,233,97,105
200 DATA 64,145,30,230,30,208
210 DATA 227,230,31,208,223
300 VTAB 10: INPUT "TEXT FILE TO TRANSFER: ";N$
310 PRINT : INPUT "FROM WHICH DRIVE: ";IN$
320 IN = VAL (IN$)
330 IF IN > 2 THEN 190
340 IF IN > 0 THEN PRINT D$;"OPEN ";N$;"D";IN
350 IF IN = 0 THEN PRINT D$;"OPEN ";N$
360 PRINT D$;"READ ";N$
370 CALL 768
380 PRINT D$;"CLOSE ";N$
390 POKE 216,0
400 POKE 6400,131
410 LET E = PEEK (30) + PEEK (31) * 256
420 POKE E,96
430 PRINT
440 PRINT D$;"BSAVE TEXT.";N$;"A$1900,L";E - 6400 + 1
450 PRINT : PRINT "DONE"
```

Issued 21 Sep 81

Apple Writer

Page 2500.009.01

APPLE WRITER AND CARRIAGE RETURNS

Apple Writer's printer output module always prints at least one character on each line. It sends out a space character before the carriage return on a "blank" line. This can interfere with some printers. Here is how to modify the PRINTER module to stop sending the extra spaces.

```
UNLOCK PRINTER
BLOAD PRINTER
CALL -155
1000:EA EA EA EA EA
0DEB:EA EA EA EA EA
0FCD:EA EA EA EA EA
0FEB:EA EA EA EA EA
1243:EA EA EA EA EA
15B9:EA EA EA EA EA
3D0G
BSAVE PRINTER, A$803, L$101C
LOCK PRINTER
```

APPLE WRITER AND CARRIAGE RETURNS

Apple Writer's printer output module always prints at least one character on each line. It sends out a space character before the carriage return on a "blank" line. This can interfere with some printers. Here is how to modify the PRINTER module to stop sending the extra spaces.

```
UNLOCK PRINTER
BLOAD PRINTER
CALL -155
1000:EA EA EA EA EA
0DEB:EA EA EA EA EA
0FCD:EA EA EA EA EA
0FEB:EA EA EA EA EA
1243:EA EA EA EA EA
15B9:EA EA EA EA EA
3D0G
BSAVE PRINTER, A$803, L$101C
LOCK PRINTER
```

CASSETTE FILE STORAGE

The following program was written for those people who want to send Apple Writer files through the mail on cassette instead of diskette. It requires that both parties have Apple Writer 1.0 or 1.1. The original file must be on diskette. The received file will automatically be saved to diskette.

```

100 REM
    CASSETTE APPLE-WRITER
    BY GROVER F. NUNNERY
    APPLE COMPUTER, INC
    CHARLOTTE, NC

102 TEXT : NORMAL : SPEED= 255
104 A$ = "*****"
    *****"

106 B$ = "***"
108 BEL$ = CHR$(7): REM CTRL-G
110 D$ = CHR$(13) + CHR$(4): REM
    <CR> + CTRL-D
112 PRINT D$;"NOMON C,I,0"
200 REM

    TITLE

202 HOME : PRINT A$: PRINT B$;: HTAB
    9: PRINT "APPLE WRITER 1.0 /
    1.1";: HTAB 38: PRINT B$
204 PRINT B$;: HTAB 38: PRINT B$
206 PRINT B$;: HTAB 13: PRINT "C
    ASSETTE FILER";: HTAB 38: PRINT
    B$
208 PRINT B$;: HTAB 38: PRINT B$
210 PRINT B$;" BY GROVER NUNNERY
    , APPLE COMPUTER ";B$
212 PRINT A$: POKE 34,8: PRINT
214 POKE 216,0: REM RESET ONERR
216 ONERR GOTO 214
218 HOME : PRINT "YOU MAY CHOOSE
    FROM THE FOLLOWING -": PRINT
220 PRINT " <L> LOAD FROM CASSE
    TTE"
222 PRINT " <S> SAVE TO CASSETT
    E"
224 PRINT " <?> CATALOG DISKETT
    E"
226 PRINT " <R> RUN APPLE WRITE
    R"
228 PRINT "<ESC> QUIT": PRINT

```

(Continued)

```

230 HTAB 1: PRINT "SELECT: "; GET
    W$
232 IF W$ = "L" THEN PRINT "LOA
    D": GOTO 600
234 IF W$ = "S" THEN PRINT "SAV
    E": GOTO 300
236 IF W$ = "?" THEN PRINT "CAT
    ALOG": PRINT D$;"CATALOG": PRINT
    : HTAB 7: PRINT "PRESS ANY K
    EY TO CONTINUE "; GET W$: GOTO
    214
238 IF W$ = "R" THEN PRINT "RUN
    APPLE WRITER": PRINT D$;"VE
    RIFY TEDITOR": TEXT : VTAB 2
    0: PRINT D$;"BRUN TEDITOR"
240 IF W$ = CHR$(27) THEN 800
242 GOTO 230
300 REM
    GET FILE TO RECORD

302 GOSUB 700: REM GET FILENAME
304 PRINT D$;"BLOAD ";N$
306 FLASH : HTAB 12: PRINT " MEA
    SURING FILE ": NORMAL
308 AD = 6400: REM BEGINNING
310 IF PEEK (AD) < > 96 THEN A
    D = AD + 1: GOTO 310
312 L = AD - 6399: REM LENGTH
400 REM
    DECIMAL TO HEX CONVERTER

402 HX$ = "0123456789ABCDEF"
404 P = 3: REM POWER OF 16
406 FOR X = 1 TO 4
408 H1%(X) = AD / 16 ^ P
410 IF H1%(X) > 0 THEN AD = AD -
    (H1%(X) * 16 ^ P)
412 P = P - 1: NEXT X
414 FOR X = 1 TO 4:H1$(X) = MID$
    (HX$,(H1%(X) + 1),1): NEXT X
416 H2$ = H1$(1) + H1$(2) + H1$(3
    ) + H1$(4)
500 REM
    WRITE FILE TO CASSETTE

502 PRINT D$;"OPEN WRITER"
504 PRINT D$;"WRITE WRITER"
506 PRINT "1900.";H2$;"W"
508 PRINT "3D0G"
510 PRINT "GOTO 530"
    
```

(Continued)

```
512 PRINT D$;"CLOSE WRITER"
514 HOME : PRINT "MAKE A NOTE OF
      THIS INFORMATION."
516 PRINT "IT WILL BE NEEDED TO
      RELOAD FILE."
518 PRINT : PRINT "--> NAME: ";N
      $: PRINT "--> MEASUREMENT: 1
      900.";H2$: PRINT "--> LENGTH
      : ";L: PRINT
520 PRINT "START CASSETTE IN ";:
      INVERSE : PRINT "RECORD";::
      NORMAL : PRINT " MODE";BEL$
      : PRINT
522 HTAB 1: PRINT "PRESS <RETURN
      > WHEN READY";: GET CR$
524 IF CR$ < > CHR$ (13) THEN
      522
526 PRINT D$;"EXEC WRITER"
528 CALL - 151
530 INPUT "";DUMMY$: HOME
532 PRINT D$;"DELETE WRITER"
534 GOTO 214
600 REM
      READ FILE FROM CASSETTE

602 GOSUB 700: REM GET FILENAME
604 PRINT : INPUT "MEASUREMENT:
      1900.";H2$
606 PRINT : INPUT "LENGTH: ";L
608 PRINT D$;"OPEN READER"
610 PRINT D$;"WRITE READER"
612 PRINT "1900.";H2$;"R"
614 PRINT "3D0G"
616 PRINT "GOTO 630"
618 PRINT D$;"CLOSE READER"
620 HOME : PRINT "START CASSETTE
      IN ";: INVERSE : PRINT "PLA
      Y";: NORMAL : PRINT " MODE";
      BEL$: PRINT
622 HTAB 1: PRINT "PRESS <RETURN
      > WHEN READY";: GET CR$
624 IF CR$ < > CHR$ (13) THEN
      622
626 PRINT D$;"EXEC READER"
628 CALL - 151
630 INPUT "";DUMMY$: HOME
632 PRINT D$;"DELETE READER"
634 PRINT D$;"BSAVE ";N$;" , A640
      0, L";L
```

(Continued)

```
636 GOTO 214
700 REM
    GET FILENAME

702 N0$ = "TEXT."
704 PRINT
706 HTAB 1: CALL - 958: PRINT "
    FILENAME: ";N0$;: GET N1$
708 IF N1$ = "?" THEN HTAB 11: PRINT
    "CATALOG": PRINT D$;"CATALOG
    ": PRINT : HTAB 7: PRINT "PR
    ESS ANY KEY TO CONTINUE ";: GET
    W$: POP : GOTO 214
710 IF N1$ = CHR$ (8) THEN N0$ =
    "": GOTO 706
712 IF N1$ = CHR$ (27) THEN 800
714 PRINT N1$;: INPUT " ";N$:N$ =
    N0$ + N1$ + N$
716 RETURN
800 REM
    END

802 PRINT D$;"NOMON C,I,O"
804 TEXT : HOME : END
```

APPLE COMPUTER TECHNICAL NOTES

Issued 3 Jun 82

Apple Writer ///

Page 2550.000.01

INDEX TO APPLE WRITER ///

ALPHABETIC LISTING

Doc	Title
001	Top Line

NUMERIC LISTING

Doc	Issue Date	Previous Date	Pages
001	JUN 3 82	-	1

Issued 3 Jun 82

Apple Writer ///

Page 2550.001.01

TOP LINE

The TopLine command changes the default top line, but page 1 may still have a top line. The reason for this is that the default top line gets printed before Apple Writer starts reading text and has a chance to see any .TL commands.

You must insure that the TL parameter in the print options menu is a blank line and that there isn't a .TL command in your file to suppress printing of a top line on page 1.

You can also use the default top line to print on page 1 and insert a .TL<CR> in the text to suppress printing of the top line on subsequent pages.

The TL command is described on page 66 of the Apple Writer /// manual.

INDEX TO APPLESOFT

ALPHABETIC LISTING

Doc	Title
016	Applesoft Internal Program Storage Format
026	Applesoft Numerics Package
002	Applesoft's Hi-Res Collision Counter
029	Cassette Program Loading Errors
013	Directly Printing the Text Screen to a Printer
017	Disabling RESET
014	Documentation Aids
009	DOS and Hi-Res
005	Dotted Vertical Lines in Hi-Res
990	Errata - Applesoft Reference Manual
991	Errata - Applesoft Tutorial
024	FRE(0) Problem
022	Garbage Collection
010	HCOLOR= and HPLOT TO
004	Hi-Res and Text
007	Hi-Res Screen Swapping
003	HPLOT TO after a DRAW
035	Internals - Device I/O Routines
038	Internals - Error Processing Routines
037	Internals - Hi-Res Graphics Routines
031	Internals - Introduction
036	Internals - Miscellaneous Routines
034	Internals - String Utilities
032	Internals - TXTPTR Routines
028	Logarithms
021	Numeric Comparison Problems
025	Out of Memory Errors
023	Overflow Error in First Line of Program
015	PEEKs, POKEs, and CALLs
001	Program and Hi-Res Memory Conflicts
006	RAM Applesoft and Hi-Res
019	Syntax Error when the Program Is Run
011	TRACE and DOS Commands
012	User Defined Functions and Chain
008	Using Applesoft Shape Tables from Disk
030	VAL (A\$) Problems
020	Variable Confilct
027	VTAB Notes

(Continued)

NUMERIC LISTING

Doc	Issue Date	Previous Date	Pages
000	MAY 12 82	OCT 22 81	2
001	APR 26 82	SEP 25 81	1
002	SEP 25 81	-	1
003	SEP 25 81	-	1
004	SEP 25 81	-	1
005	SEP 25 81	-	1
006	SEP 25 81	-	1
007	SEP 25 81	-	1
008	OCT 1 81	-	2
009	APR 26 82	OCT 1 81	1
010	SEP 24 81	-	1
011	SEP 22 81	-	1
012	SEP 25 81	-	1
013	OCT 1 81	-	1
014	SEP 25 81	-	1
015	SEP 22 81	-	1
016	SEP 25 81	-	1
017	SEP 25 81	-	1
019	SEP 28 81	-	1
020	SEP 28 81	-	1
021	OCT 19 81	-	1
022	SEP 25 81	-	1
023	OCT 1 81	-	1
024	OCT 1 81	-	1
025	OCT 1 81	-	1
026	OCT 1 81	-	1
027	SEP 25 81	-	1
028	OCT 1 81	-	1
029	SEP 25 81	-	1
030	SEP 28 81	-	1
031	OCT 21 81	-	1
032	OCT 21 81	-	2
034	DEC 14 81	-	3
035	OCT 19 81	-	2
036	NOV 19 81	-	3
037	OCT 30 81	-	2
038	NOV 24 81	-	2
990	APR 26 82	SEP 28 81	4
991	APR 28 82	SEP 28 81	3

Issued 26 Apr 82

Applesoft

Page 2600.001.01

PROGRAM AND HI-RES MEMORY CONFLICTS

The easiest way to use Hi-Res graphics with a long Applesoft program is to relocate the program. This way you avoid all the problems of program text and variables being stored in the Hi-Res buffers. The following programs will initialize the Apple's memory and RUN your application program. The program will load above the Hi-Res buffer. DOS and CHAIN will continue to load programs there until the system is re-booted or a FP command is executed.

For HGR

```
10 POKE 103,1
20 POKE 104,64
30 POKE 16384,0
40 PRINT CHR$(4);"RUN PROGRAM"
```

For HGR2 or both pages

```
10 POKE 103,1
20 POKE 104,96
30 POKE 24576,0
40 PRINT CHR$(4);"RUN PROGRAM"
```

Issued 25 Sep 81

Applesoft

Page 2600.002.01

APPLESOFT'S HI-RES COLLISION COUNTER

The collision counter in Applesoft's Hi-Res package is a byte that is incremented when a shape is drawn and it crosses a non-black dot on the screen. Applesoft does not initialize or use this byte.

The collision counter could be used to detect when a lunar lander hit a mountain by setting the counter to zero there and checking for a non-zero value after drawing the lander. It is up to the user to initialize and check this byte.

H PLOT TO AFTER A DRAW

The Applesoft DRAW command doesn't leave the internal pointer for the last plotted point where the H PLOT TO command can use it. H PLOT TO X,Y will draw a line from a random point to X,Y. The following program contains a machine language program that will decode the position and set up the H PLOT pointers. Then H PLOT TO X,Y will draw from the last plotted point of the shape.

```
10 FOR J=768 TO 780
20 READ A
30 POKE J,A
40 NEXT J
50 DATA 32,203,245,166,224,164,225
60 DATA 165,226,76,17,244
100 SHLOAD
110 HGR
120 HCOLOR=3
130 SCALE=1
140 DRAW 1 AT 100,100
150 CALL 768: REM THIS IS IT!
160 H PLOT TO 10,10
170 END
```

Issued 25 Sep 81

Applesoft

Page 2600.004.01

HI-RES AND TEXT

The Apple cannot display normal text on the graphics screen. However, characters can be drawn on the HIRES screen, even over existing pictures. The HI-RES CHARACTER GENERATOR in the Applesoft Tool Kit is a convenient way to create your own character set and output it to the Hi-Res screen. Other vendors that offer Hi-Res character generators are listed in the Vendor List, section 9900.

Issued 25 Sep 81

Applesoft

Page 2600.005.01

DOTTED VERTICAL LINES IN HI-RES

The Apple HIRES screen cannot display all colors at all locations. This causes colored vertical or near vertical lines to appear as dashed lines or not appear at all. This effect is explained more thoroughly in the Apple II Reference Manual on page 19.

Issued 25 Sep 81

Applesoft

Page 2600.006.01

RAM APPLESOFT AND HI-RES

RAM Applesoft (on disk or cassette) is loaded into decimal memory locations 2048 to 12288. The high resolution graphics screen begins at decimal address 8192 and ends at decimal address 16384. Therefore performing an HGR statement in a RAM Applesoft program will clear the High Resolution graphics page one and also part of Applesoft. The result will be to hang the system or drop the Apple into the monitor. In either case you will have to reload the RAM Applesoft and reboot the system if you have a disk drive. The only way to use Hi-Res graphics is to use HGR2 or to get an Applesoft firmware card or a Language Card.

Issued 25 Sep 81

Applesoft

Page 2600.007.01

HI-RES SCREEN SWAPPING

It is possible to plot data on one Hi-Res screen while displaying the other so that the user doesn't see the screen being changed. The POKES listed on pages 132 and 133 of the Applesoft Reference Manual control which screen is displayed and location 230 (decimal) controls which screen will be plotted on by the HIRES commands. POKE 230,32 will cause Applesoft to draw on screen 1 (HGR) and POKE 230,64 will draw to screen 2 (HGR2).

To draw on Page 1 and display Page 2 Hi-Res

```
POKE 230, 32
POKE -16299, 0
```

To draw on Page 2 and display Page 1 Hi-Res

```
POKE 230, 64
POKE -16300, 0
```

USING APPLESOFT SHAPE TABLES FROM DISK

The following programs were written to make it easier to use Applesoft shape tables. The first assumes that the shape table has been entered into memory the same as the example in the Applesoft reference manual. (It doesn't require that you use the same address, just the proper format.)

SHAPE LOADER

The Applesoft SHLOAD statement loads shape tables from cassette tape and sets up the proper pointer so that Applesoft can use it. The following program loads the shape table from the disk as a binary file and sets up the proper pointer for Applesoft. You can use any file name you choose. This routine should be the first thing in the program since it destroys the string variables in the process of loading the shape table.

```

100 INPUT "WHICH SHAPE TABLE TO LOAD ? ";F$
110 PRINT CHR$(4);"BLOAD ";F$;"A$D000"
120 DS = PEEK (977) + PEEK (978) * 256 + 3233
130 HM = PEEK (115) + PEEK (116) * 256 - PEEK (DS) -
      PEEK (DS + 1) * 256
140 PRINT CHR$(4);"BLOAD ";F$;"A";HM
150 HIMEM: HM
160 POKE 232,HM - INT (HM / 256) * 256: POKE 233,HM / 256

```

(Continued)

SHAPE SAVER

This program saves your shape table on the disk as a binary file for the SHAPE LOADER routine to use. You should load this program before getting into the monitor and typing in your shape table. The best place to start your shape table is at \$2000. That leaves lots of room for the shape saver and the table. Use the instructions in chapter 9 of the Applesoft Reference manual. When you've got it all typed in and checked, type 3D0G to get back into Applesoft and RUN the program.

```

1000 REM SHAPE SAVER
1005 H$ = "0123456789ABCDEF"
1010 TEXT : HOME : VTAB 5
1020 INPUT "WHAT IS THE STARTING ADDRESS (HEX)?";A$
1030 IF LEN (A$) = 0 THEN END
1040 GOSUB 5000
1050 IF F < 0 THEN 1000
1060 S = X
1070 PRINT
1080 INPUT "WHAT IS THE ENDING ADDRESS (HEX)?";A$
1090 IF LEN (A$) = 0 THEN 1000
1100 GOSUB 5000
1110 IF F < 0 THEN 1000
1120 LET E = X
1130 IF E < = S THEN PRINT CHR$ (7);"THE END IS LESS THAN THE
START!": GOSUB 5100: GOTO 1000
1140 PRINT
1150 PRINT "WHAT NAME DO YOU WANT TO USE ?"
1160 INPUT " ";A$
1170 PRINT CHR$ (4);"BSAVE ";A$;" ,A";S;" ,L";E - S + 1
1180 END
5000 PRINT : PRINT "CONVERTING TO DECIMAL"
5010 LET F = 0:X = F
5020 FOR J = 1 TO LEN (A$)
5030 FOR I = 1 TO 16
5040 IF MID$ (A$,J,1) = MID$ (H$,I,1) THEN X = X * 16 + I - 1:
F = F + 1
5050 NEXT I,J
5060 IF F < > LEN (A$) THEN F = - 1:"THAT'S NOT HEX!!!";
CHR$ (7): GOSUB 5100
5070 IF X > 2 ^ 16 THEN F = - 1: PRINT "THAT'S TOO BIG!!!";
CHR$ (7): GOSUB 5100
5080 RETURN
5100 FOR J = 1 TO 1000: NEXT J: RETURN

```

DOS AND HI-RES

DOS uses two memory locations (\$26,\$27) that are also used by Applesoft's Hi-Res routines during the H PLOT TO X,Y statement. DOS commands between H PLOT TO X,Y statements won't plot properly. The pointer can be saved and restored to enable mixing DOS and Hi-Res. The added two lines (25 and 35) in the example below demonstrate how to maintain the pointer values.

This won't work

```
10 D$ = CHR$(4)
20 H PLOT 1,2
30 PRINT D$;"CATALOG"
40 H PLOT TO 33,44
```

This will work

```
10 D$ = CHR$(4)
20 H PLOT 1,2
=> 25 A = PEEK(38): B = PEEK(39)
30 PRINT D$;"CATALOG"
=> 35 POKE 38,A: POKE 39,B
40 H PLOT TO 33,44
```

Issued 24 Sep 81

Applesoft

Page 2600.010.01

HCOLOR= AND HPLOT TO

HCOLOR immediately followed by a HPLLOT TO X,Y will draw a line using the previous color. This is because HPLLOT TO assumes that the internal color mask has already been set up by HPLLOT. A HPLLOT is required to plot with the new color.

Issued 22 Sep 81

Applesoft

Page 2600.011.01

TRACE AND DOS COMMANDS

TRACE won't work with DOS commands unless you define D\$ = CHR\$(13) +
CHR\$(4) because DOS expects the control-D to be the first character on a
line of output and TRACE does not send out a RETURN.

USER DEFINED FUNCTIONS AND CHAIN

User defined functions in Applesoft may cause problems if CHAIN is used. When a DEF FN statement is encountered in Applesoft there is an entry made in the simple variable table that points to the rest of the function in the text of the program. Strange and perhaps fatal things can happen if you use a function defined in the previous program without having the same image of the function at the same memory locations.

The easy way around this problem is not to use defined functions or re-define all of your defined functions in each of the chained modules.

First program:

```
5 REM PROGRAM 1
10 DEF FN A(X)=X*X
20 PRINT FN A(2)
30 PRINT CHR$(4);"BLOAD CHAIN,A520"
40 CALL 520"PROGRAM 2"
```

Second program:

```
5 REM PROGRAM 2
10 DEF FN A(X)=X*X
20 PRINT FN A(2)
30 END
```

Issued 1 Oct 81

Applesoft

Page 2600.013.01

DIRECTLY PRINTING THE TEXT SCREEN TO A PRINTER

The exact contents of the text screen can be sent to a printer with the following program. It uses the VTAB command to find the starting address of each line and adds a character counter to index across the screen. This can easily be included as a subroutine in an application program.

```
10 FOR V = 1 TO 24
20 VTAB V
30 P = PEEK (40) + PEEK (41) * 256
40 FOR H = 0 TO 39
50 PRINT CHR$( PEEK (P+H));
60 NEXT H
70 PRINT
80 NEXT V
90 END
```

Decreasing the limits of V and H in lines 10 and 40 will limit the program to send only part of the screen.

DOCUMENTATION AIDS

It really helps modifying and debugging programs if it is easy to find the various subroutines. Try the following tricks to make it easier to see what's going on in your software.

Use linefeeds (CTRL-J) as the first and last character in the text of your REMs. It will make the text of the remark stand out in the listings. Some printers will even put the remark at the left margin along with the line numbers.

You can use colons to indent FOR-NEXT loops. It makes them stand out and lets you know if you've closed them all before going on to the next routine.

```
10 REM
    THIS STANDS OUT

20 FOR J = 1 TO 10
30 : FOR K = 1 TO 10
40 :: PRINT J * K
50 : NEXT K
60 NEXT J
```

Issued 22 Sep 81

Applesoft

Page 2600.015.01

PEEKs, POKEs, AND CALLs

The PEEK, POKE, and CALL statements in Applesoft all refer to machine language programs or data being used from a BASIC program. Often their use cannot be explained because a machine language program is loaded with or generated from the specific BASIC program. There is a good description of the useful routines in the monitor in THE APPLE MONITOR PEELED (Apple product # A2L0013). There is also a list of general purpose PEEKs, POKEs, and CALLs in the Applesoft II Reference Manual starting on page 128 and in the Apple II Reference Manual.

APPLESOFT INTERNAL PROGRAM STORAGE FORMAT

Applesoft stores its programs one line at a time normally starting at \$801 and ascending in memory by line number. Each line is stored in this format:

The first item is a two byte pointer to the absolute address of the beginning of the next line. The least significant byte is first.

The second field is the two byte integer representation of the line number for that line. The least significant byte is first.

The next byte is a token that stands for the first Applesoft keyword. If the lines starts with "A = 1" then the ASCII for the variable name is first. The rest of the command will follow with keywords and symbols reduced to one byte tokens and ASCII text unmodified.

The next byte will be either a \$00 if this is the end of the line or \$3A if this is a multi-statement line. If this is a multi-statement line then the next byte will be the token or variable name as in the previous item.

There is a list of Applesoft's tokens on page 121 of the Applesoft Reference manual.

							\$8xx							
00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E
00	0D	08	0A	00	97	3A	BA	22	48	49	22	00	00	00
E	LINK	LINE #		H	:	P	"	H	I	"	E	LINK		
O				O		R					O			
L				M		I					L			
				E		N								
						T								

If the link field for a line is 00 00 then Applesoft considers it the last line in the program while running or listing. The SAVE command uses a different pointer for the end of the program. \$AF,B0 normally points to the high byte of the line number of the line beyond the last line.

Issued 25 Sep 81

Applesoft

Page 2600.017.01

DISABLING RESET

The Auto Start Monitor ROM allows the application program to retain control when the RESET key is pressed. The easiest way to take advantage of this is to have reset re-run the Applesoft program.

```
10 POKE 1010,102 : POKE 1011,213 : CALL-1169
30 REM
30 REM          RESET WILL RE-RUN THE PROGRAM NOW
40 REM
100 POKE 1010,191: POKE 1011,157: CALL -1169
110 REM
120 REM          THIS IS BACK TO NORMAL
```

Issued 28 Sep 81

Applesoft

Page 2600.019.01

SYNTAX ERROR WHEN THE PROGRAM IS RUN

Applesoft requires that the first byte in the program storage area is a zero. Some machine language programs don't leave a zero there and Applesoft may react strangely. Here's how to initialize that byte, even if the program memory pointer has been changed.

```
POKE PEEK (103) + PEEK (104) * 256,0
```

Issued 28 Sep 81

Applesoft

Page 2600.020.01

VARIABLE CONFLICTS

Only the first two characters of an Applesoft variable name are significant. That's why the variables BACK and BALL are considered to be the same variable BA. This can cause enormous problems if you don't keep track of what variable names you are using.

Issued 19 Oct 81

Applesoft

Page 2600.021.01

NUMERIC COMPARISON PROBLEMS

When two numbers print as equal but an IF statement doesn't think that they are equal, then the least significant bits in the internal binary storage format of the numbers are probably different. Applesoft's PRINT statement will truncate a number that is extremely close to being an integer. For example, 3^2 and $3*3$ will both print as 9 but they won't compare as equal. Printing $3^2 - 3*3$ will result in 3.7252903E09 and the expression $(3^2 = 3*3)$ will be false.

There is a formula to round Applesoft real numbers to a specified number of decimal places on page 18 of the Applesoft Reference manual and a program to limit the number of digits to the right of the decimal point on page 22 of the same manual.

GARBAGE COLLECTION

If your Applesoft program sometimes mysteriously stops for up to five minutes in the middle of a program and then just as mysteriously resumes as if nothing had happened, then read on. When Applesoft re-assigns a string value it assigns the new value just below the last one. Soon all of free memory is full of old strings that aren't being used. When there's no room for one more string then Applesoft sorts through all its strings, one at a time, and throws out any unused ones. This is called Garbage Collection and it can last up to five minutes depending on how much memory is available and how many string variables there are. The more string variables there are the longer it will take. Applesoft gives no warning of garbage collection, it just does it. And there is no way to prevent it.

Applesoft also does garbage collection when it encounters the FRE (Ø) function, and this gives us a tool for at least warning the user that the computer will be busy for a while. Applesoft checks the value of PEEK (112) - PEEK (11Ø) and when it is less than one, Applesoft does its garbage collecting. So we can check that value to decide when to print a "DON'T WORRY" message and then use A = FRE(Ø) to force garbage collection. Here is an example line that you could use in your program.

```
1000 IF PEEK (112) - PEEK (11Ø) < 2 THEN PRINT "STANDBY":  
      A = FRE(Ø)
```

Issued 1 Oct 81

Applesoft

Page 2600.023.01

OVERFLOW ERROR IN FIRST LINE OF PROGRAM

If the first number in the first line of the first program run after booting DOS is a negative number then Applesoft might return an OVERFLOW ERROR. This usually occurs when the first statement is "CALL -936" which can be replaced with "HOME".

Issued 1 Oct 81

Applesoft

Page 2600.024.01

FRE(Ø) PROBLEM

PRINT FRE(Ø) can cause the Apple to hang if it is done in command mode immediately after loading a program from disk or using Applesoft RENUMBER because the variable pointers aren't all reset properly. Typing CLEAR before the PRINT FRE(Ø) will return the expected results.

OUT OF MEMORY ERRORS

Sooner or later everyone sees "ERROR OUT OF MEMORY". However there are two ways to run out of memory in Applesoft. The most common is to have a program that is too big or uses too many variables. The only solution to that problem is to trim down the program, keep the data on a disk, or chain the program in from the disk in segments.

The less common cause is stack overflow. This is easy to spot because after getting the OUT OF MEMORY error, PRINT FRE(0) tells you that there is still free memory. The problem is that Applesoft uses the 6502 stack to save its recursive subroutine calls and the stack is a limited resource. Here are some causes

1. Too many FOR-NEXT loops
2. Too many GOSUBs
3. Excessively complex mathematical or string functions
4. GOSUBs with no RETURN
5. Improper recovery in ONERR GOTO routines
6. CALLs or interrupts that don't restore the stack properly

These effects are cumulative and you might be affected by more than just one. One through four are inherent in your program structure. If your program is cleanly structured then you probably won't suffer from these.

If you are using ONERR GOTO, I suggest you carefully read pages 81 and 82 and look at page 136 of the Applesoft Reference manual. There are two correct ways to leave the ONERR routine. You can use RESUME so that Applesoft can take care of the stack and re-execute the statement that caused the error. But if you don't use RESUME, you must use the stack recovery routine on pages 82 and 136 (the example on page 136 is easier to use) before you do a GOTO to any other part of the program. The routine on page 136 can be replaced with CALL -3288 and no POKEs if you can insure that you will be using ROM Applesoft.

When Applesoft executes a CALL, it does a 6502 JSR to the specified address. It's up to you to leave the stack the way you found it. Likewise any routines to handle interrupts from a peripheral card must remove any bytes pushed on the stack and maintain the 6502's internal registers.

A very complex program might run into one of the first three problems as a part of normal operation. There is a routine in Applesoft that clears the stack. CALL 54915 will clear the stack without clearing the variables. It wipes out all pending FOR-NEXT loops, GOSUBs, and formulas. This is one way to implement a program controlled restart. Beware, this is no substitute for a well structured program.

Issued 1 Oct 81

Applesoft

Page 2600.026.01

APPLESOFT NUMERICS PACKAGE

Sometimes Applesoft's math package doesn't give the exact answer that you expect. That is caused by the fact that Applesoft does all of its calculations in a 32 bit binary floating point format and there are no exact equivalents to most numbers. Also, Applesoft uses natural logarithms to calculate many of its transcendental functions, which also adds small errors to the results.

Issued 25 Sep 81

Applesoft

Page 2600.027.01

VTAB NOTES

The VTAB statement moves the cursor to an absolute vertical position. It completely ignores the current text window. A VTAB to a line above the text window will start writing as if the window were the whole screen until it scrolls down to the window. After it scrolls into the window, it will stay there and the information above the window won't be scrolled. A VTAB below the window will cause writing only on the line that the cursor is on and normal scrolling within the window with each carriage return.

Issued 1 Oct 81

Applesoft

Page 2600.028.01

LOGARITHMS

In Applesoft, LOG (X) returns the natural logarithm of X. The logarithm base ten of a number can be calculated by

$$\text{Logarithm base } 10 = \text{LOG (X) / LOG (10)}$$

Issued 25 Sep 81

Applesoft

Page 2600.029.01

CASSETTE PROGRAM LOADING ERRORS

Applesoft interacts with the Auto Start so that if an ERR message is generated while loading a program from cassette, the Apple will hang. It will ignore all commands including RESET. There are two ways to regain control. First turn the Apple off and back on. Second, rewind the cassette and play it into the Apple some more until it returns to command mode.

Issued 28 Sep 81

Applesoft

Page 2600.030.01

VAL (A\$) PROBLEMS

Applesoft in a non-disk Apple doesn't initialize its memory properly. The result is that the following program doesn't give the desired result.

```
100 GET A$  
110 PRINT A$, VAL (A$)
```

If you input "1" the answer will be "1, 1.111111E16". A HIMEM:
statement will reset the pointer.

```
16K HIMEM: 16384  
32K HIMEM: 32768  
48K HIMEM: 49152
```

INTERNALS - INTRODUCTION

The following six notes are a guide for the 6502 machine language programmer who wants to take advantage of the various subroutines in the Applesoft ROM. The information was first published in The Apple Orchard for March 1980. The addresses given assume that the user has an Apple II Plus, an Applesoft firmware card, or Applesoft loaded into a Language Card. This data is meant for the experienced programmer, NOT THE BEGINNER. Consult your Applesoft Reference manual for more information.

Take special note of CHRGET. It is the heart of Applesoft. When Applesoft wants the next character of an instruction it points TXTPTR at the program or the input buffer and JSRs to CHRGET. When Applesoft READs DATA, TXTPTR is temporarily set to the last used DATA statement.

ABBREVIATIONS

msb most significant bit or byte
lsb least significant bit or byte
eol end of line token (\$00)

A the 6502 accumulator
X the 6502 X register
Y the 6502 Y register
Z the zero flag of the 6502 status register
C the carry flag of the 6502 status register

A,X is a 16 bit number where A has the msb and X the lsb.

(Y,A) is the number or string whose address is in Y and A with the msb in Y and the lsb in A.

FAC the floating point accumulator
ARG the ARGument register

Issued 21 Oct 81

Applesoft

Page 2600.032.01

INTERNALS - TXTPTR ROUTINES

CHRGET 00B1

Increment TXTPTR and use CHRGOT to get the character TXTPTR now points at.

CHRGOT 00B7

Load A from TXTPTR and set certain 6502 status flags. X and Y are not changed. On exit:

A = the character currently pointed at by TXTPTR
 Z is set if A is ":" or eol (\$3A or \$00)
 C is clear if A is an ASCII number("0" to "9").

LINGET DA0C

Read a line number (integer 0 to 63999) from TXTPTR into LINNUM. LINGET assumes that the 6502 registers and A have been set up by the JSR to CHRGET that fetched the first digit. LINGET normally exits through CHRGOT which fetches the character after the last digit. If the number is greater than 63999 then LINGET exits via SYNTAX ERROR. LINNUM is zero if there is no number at TXTPTR.

GTBYTC E6F5

JSR to CHRGET to gobble a character, evaluates the formula at TXTPTR, and returns a single byte integer in X and FACLO. On entry TXTPTR points to the first character of the formula. GTBYTC normally exits through CHRGOT. If FAC is greater than 255 or less than 0 then it exits through ILLEGAL QUANTITY ERROR.

GETBYT E6F8

Evaluates the formula at TXTPTR and returns a single byte integer in X and FACLO. On entry TXTPTR points to the first character of the formula. GETBYT normally exits through CHRGOT. If FAC is greater than 255 or less than 0 then it exits through ILLEGAL QUANTITY ERROR.

PLOTFNS F1EC

Get two LORES plotting coordinates (0-47,0-47) separated by a comma from TXTPTR. On entry TXTPTR points to the first character of the formula for the first number. PLOTFNS puts the first number in FIRST and the second number in H2 and V2.

(Continued)

APPLE TECH NOTES

Copyright (C) 1981 by Apple Computer, Inc.

"AppleIACTNJuly82_2600-032-01" 145 KB 1962-10-11 dpi: 300h x 300v pix: 1943h x 3002v

HFNS F6B9

Get HIRES plotting coordinates (0-279,0-191) from TXTPTR. On entry TXTPTR points to the first character of the formula for the first number. HFNS leaves the 6502 registers set up for HPOSN. On exit:

A = vertical coordinate
X = lsb of horizontal coordinate
Y = msb of vertical coordinate

INTERNALS - STRING UTILITIES

In Applesoft strings have two parts: the descriptor and the ASCII string. A string descriptor contains the length of the string and the address of its first character. See page 137 of the Applesoft Reference Manual for how these are used in normal variable storage. Through most of the routines the descriptor is left in memory and a two byte pointer to the descriptor is kept in the FAC. The pointer is the address of the descriptor. The actual string could be anywhere in memory. In a program 10 A\$ = "HI" will leave a descriptor pointing into the program text.

STRINI E3D5

Get space for creation of a string whose descriptor is pointed at by FACMO,FACLO and create a descriptor for it in DSCTMP. On entry A = length of the string and FACMO,FACLO point to the string descriptor. STRINI uses GETSPA which will force garbage collection if there isn't enough room. GETSPA will exit with an "OUT OF MEMORY ERROR" if garbage collection can't make enough room. GETSPA moves FRESPC and FRETOP down enough to store the string. STRINI returns with the descriptor to the new string in DSCTMP and the original pointer in DSCPNT.

STRSPA E3DD

Get space for the creation of a string that is A bytes long and create a descriptor for it in DSCTMP. STRSPA uses GETSPA which will force garbage collection if there isn't enough room. GETSPA will exit with an "OUT OF MEMORY ERROR" if garbage collection can't make enough room. GETSPA moves FRESPC and FRETOP down enough to store the string. Returns with A unchanged and the new descriptor in DCSTMP.

GETSPT DA7B

GETSPT moves the descriptor pointed at by FACMO,FACLO into the memory pointed to by FORPNT. GETSPT also moves the text of the string up to FRETOP if it isn't in the program text. GETSPT will free up the string if it was a string temporary. GETSPT uses GETSPA and may cause garbage collection or an "OUT OF MEMORY ERROR".

MOVINS E5D4

Move a string whose descriptor is pointed to by STRNG1 to memory pointed to by FRESPA.

(Continued)

MOVSTR E5E2

Move the string pointed to by Y,X with a length of A to memory pointed to by FRESPA.

STRTXT DE81

Build a temporary descriptor for a string literal whose first character is pointed to by TXTPTR plus C and ends with a quote. The resulting temporary descriptor is pointed to by FACMO,FACLO. Leading quotes should be skipped before calling STRLIT. On exit the character after the string literal is pointed to by STRNG2.

STRLIT E3E7

Build a temporary descriptor for a string literal whose first character is pointed to by Y,A and ends with a quote. The resulting temporary descriptor is pointed to by FACMO,FACLO. Leading quotes should be skipped before calling STRLIT. On exit the character after the string literal is pointed to by STRNG2.

STRLT2 E3ED

Take a string literal whose first character is pointed to by Y,A and build a temporary descriptor for it pointed to by FACMO,FACLO. Characters other than zero that terminate the string should be saved in CHARAC and ENDCHR. Leading quotes should be skipped before calling STRLT2. On exit the character after the string literal is pointed to by STRNG2.

PUTNEW E42A

Move the descriptor in DSCTMP to a temporary descriptor, put a pointer to the descriptor in FACMO,FACLO, and flag the result as a string.

GETSPA E452

Get A bytes of memory below HIMEM for a string. GETSPA will force garbage collection if there isn't enough room. GETSPA will exit with an "OUT OF MEMORY ERROR" if garbage collection can't make enough room. Moves FRESPC and FRETOP down enough to store the string. On entry A = number of characters. Returns with A unchanged and pointer to the space in FRESPC, FRETOP, and Y,X.

(Continued)

FRESTR E5FD

Make sure that the last FAC result was a string and free up the temporary string result. A check is made to see if the descriptor is a temporary one allocated by PUTNEW. On exit the address of the string is in INDEX and Y,X. The length of the string is in A. Uses FRETMP.

FREFAC E600

Free up the temporary string pointed to by FACMO,FACLO. A check is made to see if the descriptor is a temporary one allocated by PUTNEW. On exit the address of the string is in INDEX and Y,X. The length of the string is in A. Uses FRETMP.

FRETMP E604

Free up a temporary string. On entry the pointer to the descriptor is in Y,A. A check is made to see if the descriptor is a temporary one allocated by PUTNEW. If so, the temporary is freed up by updating TEMPPT. If a temp is freed up a further check is made to see if the string is the lowest in memory. If so, that area of memory is freed up by updating FRETOP. On exit the address of the string is in INDEX and Y,X. The length of the string is in A.

FRETMS E635

Free the temporary descriptor without freeing up the string. On entry Y,A point to the descriptor to be freed. On exit Z is set if anything was freed.

Issued 19 Oct 81

Applesoft

Page 2600.035.01

INTERNALS - DEVICE I/O ROUTINES

INLIN D52C

Input a line of text without sending out a prompt from the current input device into the input buffer, BUF. INLIN uses GDBUF to strip off the most significant bit on all input characters.

INLIN+2 D52E

Input a line of text prompting with the character in X from the current input device into the input buffer, BUF. INLIN uses GDBUF to strip off the most significant bit on all input characters.

GDBUFS D539

Puts a zero at the end of the input buffer, BUF, and masks off the msb on all bytes. On entry: X = the end of the input line. On exit:

A = 0
X = FF
Y = 1

INCHR D553

Get one character from the current input device in A and mask off the msb. INCHR uses the main Apple input routines and supports normal handshaking.

STROUT DB3A

Print string pointed to by Y,A. The string must end with a nul or a quote.

STRPRT DB3D

Print a string whose descriptor is pointed to by FACMO,FACLO.

OUTDO DB5C

Print the character in A. INVERSE, FLASH, and NORMAL in effect.

CRDO DAFB

Print a carriage return.

(Continued)

APPLE TECH NOTES

Copyright (C) 1981 by Apple Computer, Inc.

"AppleACTNJuly82_2600-035-01" 101 KB 1962-10-11 dpi: 300h x 300v pix: 1991h x 2991v

OUTSPC DB57

Print a space.

OUTQST DB5A

Print a question mark.

INPRT ED19

Print " IN " and the current line number from CURLIN. Uses LINPRT.

LINPRT ED24

Prints the 2 byte unsigned number in X,A.

PRNTFAC ED2E

Prints the current value of FAC. FAC is destroyed. Uses FOUT and STROUT.

INTERNALS - MISCELLANEOUS ROUTINES

PTRGET DFE3

Read a variable name from CHRGET and find it in memory. On entry TXTPTR points to the first character of the variable name. On exit the address to the value of the variable is in VARPNT and Y,A. If PTRGET can't find a simple variable it creates one. If it can't find an array it creates one dimensioned from 0 to 10 and sets all elements equal to zero.

GETARYPT F7D9

Read a variable name from CHRGET and find it in memory. On entry TXTPTR points to the first character of the variable name. This routine leaves LOWTR pointing to the name of the variable array. If the array can't be found the result is an OUT OF DATA ERROR.

FNDLIN D61A

Searches the program for the line whose number is in LINNUM. On exit:

 C set LOWTR points to the link field of the desired line.
 C clear then not found. LOWTR points to the next higher line.

DATA D995

Move TXTPTR to the end of the statement. Looks for the ":" or eol (0).

DATAN D9A3

Calculate the offset in Y from TXTPTR to the next ":" or eol (0).

REMN D9A6

Calculate the offset in Y from TXTPTR to the next eol (0).

ADDON D998

Add Y to TXTPTR.

SCRTCH D64B

The "NEW" command. Clears the program, variables, and stack.

(Continued)

CLEARC D66C

The "CLEAR" command. Clears the variables and stack.

STKINI D683

Clears the stack.

RESTOR D849

Sets the DATA pointer, DATPTR, to the beginning of the program.

STXTPT D697

Set TXTPTR to the beginning of the program.

BLTU D393

Block transfer makes room by moving everything forward. On entry:

 Y,A and HIGHDS = destination of high address + 1
 LOWTR = lowest address to be moved
 HIGHTR = highest address to be moved + 1

On exit:

 LOWTR is unchanged
 HIGHTR = LOWTR - \$100
 HIGHDS = lowest address transfered - \$100

REASON D3E3

Makes sure there's enough room in memory for another variable. Checks to be sure that the address Y,A is less than FRETOP. May cause garbage collection. Causes "OUT OF MEMORY ERROR" if there's no room.

GARBAG E484

Move all currently used strings up in memory as far as possible. This maximizes the free memory area for more strings or numeric variables.

(Continued)

Issued 19 Nov 81

Applesoft

Page 2600.036.03

CONT D898

Moves OLDTXT and OLDLIN into TXTPTR and CURLIN.

NEWSTT D7D2

Execute a new statement. On entry TXTPTR points to the ":" preceding the statement or the zero at the end of the previous line. Use NEWSTT to restart the program with CONT. THIS ROUTINE DOES NOT RETURN.

RUN D566

Run the program in memory. THIS ROUTINE DOES NOT RETURN.

GOTO D93E

Uses LINGET and FNDLIN to update TXTPTR. GOTO assumes that the 6502 registers and A have been set up by the CHRGET that fetched the first digit.

LET DA46

Uses CHRGET to get address of the variable, "=", evaluate the formula, and store it. On entry TXTPTR points to the first character of the variable name.

INTERNALS - HI-RES GRAPHICS ROUTINES

NOTE: Regardless of which screen is being displayed, HPAG (location \$E6) determines which screen is drawn on. (\$20 for HGR, \$40 for HGR2)

HGR2 F3D8

Initialize and clear page 2 Hi-Res.

HGR F3E2

Initialize and clear page 1 Hi-Res.

HCLR F3F2

Clear the current Hi-Res screen to black.

BKGND F3F6

Clear the current Hi-Res screen to last plotted color.

HPOSN F411

Positions the Hi-Res cursor without plotting. HPAG determines which page the cursor is pointed at. On entry:

Horizontal = Y,X
Vertical = A

HLOT F457

Call HPOSN then try to plot a dot at the Hi-Res cursor's position. The dot may not be plotted if plotting non-white at a complementary color X coordinate.

HLIN F53A

Draws a line from the last plotted point or line destination to the coordinate in the 6502 registers. On entry:

Horizontal = X,A
Vertical = Y

(Continued)

HFIND F5CB

Convert the Hi-Res cursor's position to X-Y coordinates. HFIND allows the user to tell where on the screen the a shape table left the cursor. On exit:

 \$E0 = horizontal lsb
 \$E1 = horizontal msb
 \$E2 = vertical

DRAW F601

Draw the shape pointed to by Y,X using the current HCOLOR starting at the current Hi-Res cursor position. On entry A = rotation factor.

XDRAW F65D

Draw the shape pointed to by Y,X by starting at the current Hi-Res cursor position inverting the existing color of the dots the shape draws over. On entry A = rotation factor.

SETHCOL F6EC

Set the Hi-Res color to X. X must be less than 8.

SHLOAD F775

Loads a shape table into memory from tape above MEMSIZ (HIMEM) and sets the pointer at \$E8.

Issued 24 Nov 81

Applesoft

Page 2600.038.01

INTERNALS - ERROR PROCESSOR ROUTINES

ERROR D412

Checks ERRFLG and jumps to HNDLERR if ONERR is active. Otherwise it prints <cr> "?" <error message pointed to by X> " ERROR". If this is during program execution then it also prints " IN " and the CURLIN.

HANDLERR F2E9

Saves CURLIN in ERRLIN, TXTPTR in ERRPOS, X in ERRNUM, and REMSTK in ERRSTK. X contains the error code. See the Applesoft Reference Manual page 136 for the value of X for a given error.

RESUME F317

Restores CURLIN from ERRLIN and TXTPTR from ERRPOS and transfers ERRSTK into the 6502 stack pointer.

ISCNTC D858

Checks the Apple keyboard for a control-C (\$83). Executes the BREAK routine if there is a control-C.

CHKNUM DD6A

Make sure FAC is numeric. See CHKVAL.

CHKSTR DD6C

Make sure FAC is a string. See CHKVAL.

CHKVAL DD6D

Checks the result of the most recent FAC operation to see if it is a string or numeric variable. A TYPE MISMATCH ERROR results if FAC and C don't agree. On entry:

C set checks for strings
C clear checks for numerics.

(Continued)

ERRDIR E306

Causes ILLEGAL DIRECT ERROR if the program isn't running. X is modified.

ISLETC E07D

Checks A for an ASCII letter ("A" to "Z"). On exit C set if A is a letter.

PARCHK DEB2

Checks for "(", evaluates a formula, and checks for ")". Uses CHKOPN and FRMEVL then falls into CHKCLS.

CHKCLS DEB8

Checks at TXTPTR for ")". Uses SYNCHR.

CHKOPN DEBB

Checks at TXTPTR for "(" . Uses SYNCHR.

CHKCOM DEBE

Checks at TXTPTR for ", ". Uses SYNCHR.

SYNCHR DECO

Checks at TXTPTR for the character in A. TXTPTR is not modified. Normally exits through CHRGET. Exits with SYNTAX ERROR if they don't match.

Issued 26 Apr 82

Applesoft

Page 2600.990.01

ERRATA - APPLESOFT REFERENCE MANUAL
 A2L0006
 030-0013-03

Page 5

Paragraph 4, the last line should read:

However, only the first 9 digits are usually significant, and the tenth digit is rounded off.

Page 15

Line 90 should read

90 DIM A(8): REM DIMENSION ARRAY WITH MAX. 9 ELEMENTS

Page 15

Line 180 should read

180 IF A(I) <= A(I+1) THEN GOTO 240

Page 21

The line about mid-page that reads

C\$ = RIGHT\$(B\$,3) + "-" + LEFT\$(B\$,4) + "-" + MID\$(B\$,6,7)

should read

C\$ = RIGHT\$(B\$,4) + "-" + LEFT\$(B\$,4) + "-" + MID\$(B\$,6,7)

Page 23

Line 180 should read

180 I = I+1: IF I < 15 THEN GOTO 130

Page 51

If TAB(X) is the last item in a PRINT statement Applesoft will act as if there is a semi-colon after it.

Page 52

If SPC(X) is the last item in a PRINT statement Applesoft will act as if there is a semi-colon after it.

(Continued)

APPLE TECH NOTES

Copyright (C) 1982 by Apple Computer, Inc.

"Apple|ACTN|July82_2600-990-01" 90 KB 1962-10-11 dpi: 300h x 300v pix: 1961h x 2996v

Page 53

FRE returns the amount of memory ... Applesoft sometimes stores duplicate strings separately.

Page 53

PRINT FRE (0) sometimes returns a negative number because of the way integers are handled in Applesoft. Adding 65536 to the negative number will give the positive representation.

Page 63

Rule 2 should read "The total number of elements RECALLED must be at least equal to" instead of "...equal th".

Page 66

Similarly, a response... If an ASCII NUL character, CONTROL-SHIFT-P, is included in a line of input, the input will be truncated at the character before the NUL. A NUL as the first character in response to INPUT A\$ will result in a null string. A NUL as the first character in response to INPUT A will result in ?REENTER.

Page 66-67

A line of input longer than 255 characters will be cancelled by the monitor and the user is left in the input statement. A line greater than 239 but less than 255 will be truncated to 239 characters.

Page 81

ONERR GOTO, "When an error occurs..." should read "After executing an ONERR GOTO command, when an error occurs...".

Page 96

Paragraph 1, line 6, the sentence that starts "Press the RESET key..." should read "Type CALL -151" because Apple IIs with the Auto-Start ROM will not go into the monitor when the RESET key is pressed.

Page 102

The function for LOG of X base 10 is $\text{LOG}(X) / \text{LOG}(10)$.

Page 118

"1) Use multiple statements" states that the maximum line number is 65529 when the actual maximum line number is 63999.

(Continued)

Page 131

Second paragraph, line 7, replace "POKE 22,W" with "POKE 33,W".

Page 135

Replace the top part with

260 POKE -16296,0

Clear game control "annunicator" output #0 (Game I/O connector, pin 15) to TTL low (0.3 volts). This is the "off" condition: maximum current 8 milliamperes.

270 POKE -16295,0

Set game control "annunicator" output #0 (Game I/O connector, pin 15) to TTL high (3.5 volts). This is the "on" condition: maximum current 0.4 milliamperes.

280 POKE -16294,0

Clear game control "annunicator" output #1 (Game I/O connector, pin 14) to TTL low (0.3 volts). This is the "off" condition: maximum current 8 milliamperes.

290 POKE -16293,0

Set game control "annunicator" output #1 (Game I/O connector, pin 14) to TTL high (3.5 volts). This is the "on" condition: maximum current 0.4 milliamperes.

300 POKE -16292,0

Clear game control "annunicator" output #2 (Game I/O connector, pin 13) to TTL low (0.3 volts). This is the "off" condition: maximum current 8 milliamperes.

310 POKE -16291,0

Set game control "annunicator" output #2 (Game I/O connector, pin 13) to TTL high (3.5 volts). This is the "on" condition: maximum current 0.4 milliamperes.

320 POKE -16290,0

Clear game control "annunicator" output #3 (Game I/O connector, pin 12) to TTL low (0.3 volts). This is the "off" condition: maximum current 8 milliamperes.

330 POKE -16289,0

Set game control "annunicator" output #3 (Game I/O connector, pin 12) to TTL high (3.5 volts). This is the "on" condition: maximum current 0.4 milliamperes.

(Continued)

Page 137

The table for string pointers is wrong.

STRING POINTERS

+	+-----+	
+	+-----+	
+	+-----+	

STRING POINTERS

+	+-----+	
+	+-----+	

Issued 28 Apr 82

Applesoft

Page 2600.991.01

ERRATA - APPLESOFT TUTORIAL
030-0044-00
A210018

Page 6

References to the RESET key may need to mention the control key for newer Apples.

Page 17

"STOPPING THE COMPUTER"; Pressing RETURN after a control-C may be required if the Apple is waiting for input from the keyboard.

Page 23-24

There are only five different elementary arithmetic operations. +,-,*,/,^

Page 37

GAMEPOINTS = 45 won't work because it contains a key word, GAMEPO INT, and it will return a SYNTAX ERROR. GAMEPTS = 45 should be OK.

Page 38

The line before the last listing should say "Try the statements below in order:" instead of "... in the next page..."

Page 59

Line 230 at the bottom of the page should read

```
230 IF N < = 10 THEN GOTO 210
```

Page 63

Line 270 should read

```
270 X = X / 7
```

Page 65

Line 3220 should read

```
3020 COLOR = N
```

(Continued)

APPLE TECH NOTES

Copyright (C) 1982 by Apple Computer, Inc.

"AppleI ACTN July82_2600-991-01" 86 KB 1962-10-11 dpi: 300h x 300v pix: 1973h x 2985v

Page 69

The last line on the page reads:

```
120 PRINT "THE STRIKES AND BALLS ARE ";STRIKES;" ";BALLS
```

It is difficult to tell this because the space after STRIKES;" comes at the end of the line of listing.

Page 71

Lines 650 and 660 are reversed. They should read

```
650 NEXT Y
660 NETX X
```

Page 79

The two example line 780s are incorrect. The word "BACKGROUND" will be parsed as "BACK GR OUND" because GR is a key word and because there is already a variable BALL which would share the same Applesoft variable name, BA. Try using "FIELD" instead.

Page 81

Paragraph 3, "One possible solution is given on the next page," should refer to the program below on the same page.

Page 85

Line 220 has an extra ")". It should read

```
220 COLOR= INT (16 * RND(1))
```

Page 95

The program will give incorrect values for the Y variable unless some time delay is installed. Add this line: 1005 FOR J = 1 TO 10: NEXT J

Page 96

Line 360 should read

```
360 FOR S = 0 TO 1: REM 2 LINES, FROM Y AND Y + 1
```

Page 105

Replace line 120 with

```
120 WHOLE$ = HALF$ + " " + OTHERHALF$
```

(Continued)

Issued 28 Apr 82

Applesoft

Page 2600.991.03

Page 108

According to page 99 and iii, the title on this page should read "INTRODUCING ARRAYS: DIM".

Page 109

The last sentence in the last paragraph should read "The program below accomplishes this."

Page 109

Line 340 should read

```
340 TEMP = GLASS(WINE):GLASS(WINE) = GLASS(MILK):GLASS(MILK) = TEMP
```

Page 110

Starting in the middle of the 9th line, the text should read; "Then line 320 makes sure that the value of WINE is not equal to the value of MILK at any given time. The contents of variables GLASS(WINE) and GLASS(MILK) are switched in line 340. Finally the array is printed with lines 370 through 390."

Page 130

Clear the entire screen should be press <ESC> then <SHIFT> and <P> instead of <CONTROL> and <P>.

Page 148-149

The green keys should have slashed zeros to differentiate them from capital O.

 Issued 12 May 82

Applesoft Firmware Card

Page 2700.000.01

INDEX TO APPLESOFT FIRMWARE CARD

ALPHABETIC LISTING

Doc	Title
002	Firmware Card and System Errors
001	Firmware Card Options

NUMERIC LISTING

Doc	Issue Date	Previous Date	Pages
000	MAY 12 82	OCT 21 81	1
001	OCT 1 81	-	1
002	SEP 21 81	-	1

FIRMWARE CARD OPTIONS

The Applesoft firmware card was designed to supply floating point Basic to Apple II owners. There are three options on the card that the user may select.

The first and most obvious option is the switch on the rear of the card. It is used to force the selection of which bank of ROMs, the firmware card or the motherboard, will be used immediately after a system reset. If the switch is up then the system will default to Applesoft, otherwise it will default to Integer Basic.

This can also be controlled by software by accessing address \$C080 to select Applesoft and \$C081 to select Integer Basic. DOS confuses things by forcing the version of BASIC it was last used every time it gets control, regardless of the position of the switch and DOS gets control immediately after a reset in a system with an Auto-Start ROM.

The second option concerns the monitor ROM. Between ICs B3 and B4 there is a solder circle divided into two pads with the label "F8" silk screened under it. If these pads are separated then the monitor ROM on the motherboard will always be used. But if the pads are connected then the old monitor ROM on the card will be used with Applesoft.

This can be interesting if you have an old monitor with your Integer Basic. If the switch is set to go to Integer Basic then the Apple will drop into the old monitor when you RESET it and while you are in Integer Basic the Auto Start monitor functions won't be available.

The third option also concerns using your own language and monitor. Apple Computer uses 2316 type ROMs for all our firmware. However 2716 type (+5v only version) EPROMs can be used to replace the ROMs on the firmware card. There are two pairs of solder spots that can be connected to reconfigure the card for 2716s. They are in the upper right corner with a box drawn around them and "2716" written over it. The card can only use one type at a time, 2716s and 2316s cannot be mixed.

Issued 21 Sep 81

Applesoft Firmware Card

Page 2700.002.01

FIRMWARE CARD AND SYSTEM ERRORS

Occasionally, peripheral cards with solder plated edge connectors collect some oxidation on the contact fingers which can cause an intermittent connection. This can result in various unusual and unexpected system errors when it happens to the Applesoft Firmware Card.

To clean off contacts, turn off the power and remove the cards. Using a soft pencil eraser ("Pink Pearl" or such), gently clean off the contacts. Replace the boards, seat firmly, then reboot the system. If this does not correct the problem, contact your dealer for assistance.

Issued 12 May 82

Artist Designer

Page 2800.000.01

INDEX TO ARTIST DESIGNER

ALPHABETIC LISTING

Doc	Title
001	Artist Designer and the Apple ///

NUMERIC LISTING

Doc	Issue Date	Previous Date	Pages
000	MAY 12 82	OCT 28 81	1
001	OCT 28 81	-	1

Issued 28 Oct 81

Artist Designer

Page 2800.001.01

ARTIST DESIGNER AND THE APPLE ///

The Artist Designer package won't work on the Apple /// in emulation mode because it is written in Pascal and there is no Language Card in the Apple ///.

 Issued 12 May 82

Auto-Start ROM

Page 2900.000.01

INDEX TO AUTO-START ROM

ALPHABETIC LISTING

Doc	Title
001	Syntax Errors and Control-S

NUMERIC LISTING

Doc	Issue Date	Previous Date	Pages
000	MAY 12 82	OCT 28 81	1
001	OCT 28 81	-	1

SYNTAX ERRORS AND CONTROL-S

Sometimes if you use the control-S feature to examine a Basic program listing, you will get a SYNTAX ERROR with the next typed line. This is probably because you accidentally typed an extra control-S after the program was done listing. This extra control-S is put into the input buffer like any other characters and, since no Basic command starts with a control-S, you get a SYNTAX ERROR. Typing the left-arrow key until the Apple generates a new prompt line will insure that there is no control-S in the buffer.

 Issued 12 May 82

Business Basic

Page 3100.000.01

INDEX TO BUSINESS BASIC

ALPHABETIC LISTING

Doc	Title
001	Business Basic Numerics
002	Business Basic Turnkey Operation
003	Finding What Volume is in What Drive
990	Errata - Apple Business Basic Reference Manual

NUMERIC LISTING

Doc	Issue Date	Previous Date	Pages
000	MAY 12 82	SEP 30 81	1
001	SEP 30 81	-	1
002	SEP 25 81	-	1
003	JAN 18 82	-	1
990	APR 26 82	SEP 30 81	1

Issued 30 Sep 81

Business Basic

Page 3100.001.01

BUSINESS BASIC NUMERICS

Apple /// Business Basic uses the same floating point math package as Applesoft internally. However, Business Basic will only display 6 decimal digits maximum when it prints. This limitation was built in because the last three digits of Applesoft's math package develop errors that can seriously affect some applications.

Issued 25 Sep 81

Business Basic

Page 3100.002.01

BUSINESS BASIC TURNKEY OPERATION

The Apple /// will automatically run a program called HELLO on the boot diskette after booting.

Issued 18 Jan 82

Business Basic

Page 3100.003.01

FINDING WHAT VOLUME IS IN WHAT DRIVE

Sometimes it is necessary to find what is the volume name of the diskette in ".D2". Here is a small program that does that.

```
10 TEMP$ = PREFIX$
20 PREFIX$ = ".D1"
30 D1$ = PREFIX$
40 PREFIX$ = ".D2"
50 D2$ = PREFIX$
60 PREFIX$ = TEMP$
70 PRINT ".D1 contains ";D1$;" and .D2 contains ";D2$
```

 Issued 26 Apr 82

Business Basic

Page 3100.990.01

ERRATA - APPLE BUSINESS BASIC REFERENCE MANUAL
 A3L0002
 030-0122-00

Page 128

The information at the top of the page is missing its periods. It should read

```

)PRINT USING "+###.###"+ 3.14159
+ 3.142
    
```

This spec, +###.##,

Page 151

The line at the top of the page should read

```

DEF FN MODB(A)=INT ((A - INT (A / B) * B +.5)
    
```

Issued 12 May 82

Centronics Printer Interface

Page 3200.000.01

INDEX TO CENTRONICS PRINTER INTERFACE

ALPHABETIC LISTING

Doc	Title
002	Controlling the Most Significant Bit

NUMERIC LISTING

Doc	Issue Date	Previous Date	Pages
000	MAY 12 82	-	1
002	DEC 14 81	-	3

CONTROLLING THE MOST SIGNIFICANT BIT

INTRODUCTION

This driver allows the user control bit 8 from an Apple Centronics Printer Interface with a POKE. Bit 8 is used by some printers to select expanded or normal print mode or to enable alternate or graphics character sets. DOS 3.2 or DOS 3.3 is required to use this routine. It will work in the Apple /// in Emulation mode.

SOFTWARE ENTRY

First you must decide which slot the interface will go in and enter the driver. The routine is customized for this slot number and won't work properly if used with a different configuration. Enter the driver using the values from the table for words in brackets, < >.

SLOT	1	2	3	4	5	6	7
CODE	C1	C2	C3	C4	C5	C6	C7

Enter the monitor with CALL -155 and type

```

3B0:A9 <SLOT>
:20 95 FE
:A9 80
:20 ED FD
:A9 C5
:85 36
:A9 03
:85 37
:4C EA 03
:29 7F
:0D CD 03
:4C 02 <CODE>
:80
    
```

(Continued)

To check your typing, type

3B0L

and compare your listing to the one below for an interface plugged into slot 1.

03B0-	A9 01	LDA	#\$01
03B2-	20 95 FE	JSR	\$FE95
03B5-	A9 80	LDA	#\$80
03B7-	20 ED FD	JSR	\$FDED
03BA-	A9 C5	LDA	#\$C5
03BC-	85 36	STA	\$36
03BE-	A9 03	LDA	#\$03
03C0-	85 37	STA	\$37
03C2-	4C EA 03	JMP	\$03EA
03C5-	29 7F	AND	#\$7F
03C7-	0D CD 03	ORA	\$03CD
03CA-	4C 02 C1	JMP	\$C102
03CD-	80	???	

Now return to BASIC with 3D0G

SAVING THE PROGRAM TO DISK:

The driver should be in memory before the printer is used. Save the driver by typing

BSAVE CEN 730, A\$3B0, L\$1E

(Continued)

USING THE PRINTER:

The first time you want to use the printer you must load the driver and initialize the interface. From command mode type

```
BLOAD CEN 730  
CALL 944
```

This may be done from a program by entering

```
100 PRINT D$;"BLOAD CEN 730" : CALL 944
```

assuming that D\$ is a control-D.

If you want to switch back to the video monitor for output type

```
PR#0
```

or in a program enter

```
200 PRINT D$;"PR#0"
```

Then to reconnect the printer, all that is required is

```
CALL 954
```

or from a program

```
300 CALL 954
```

SETTING THE PRINT MODES:

To set normal print mode POKE 973,0

To set the high bit to "1" POKE 973,128

INDEX TO COMMUNICATIONS INTERFACE

ALPHABETIC LISTING

Doc	Title
005	Applesoft and Datamover
990	Errata - Communications Interface Manual
006	Initializing with POKEs
002	Printer Interfacing
001	Receiving Lower Case Characters in Terminal Mode
004	Screen Echo During Output
003	Speeding Up the Communications Interface

NUMERIC LISTING

Doc	Issue Date	Previous Date	Pages
000	JUN 1 82	MAY 12 82	1
001	OCT 1 81	-	1
002	NOV 19 81	-	1
003	MAY 21 82	-	1
004	DEC 14 81	-	1
005	NOV 17 81	-	1
006	MAY 21 82	-	1
990	APR 26 82	OCT 21 81	1

Issued 1 Oct 81

Communications Interface

Page 3300.001.01

RECEIVING LOWER CASE CHARACTERS IN TERMINAL MODE

The lower case information on page 17 of the manual is only effective in terminal mode. To get the lower case characters to be displayed in inverse you need to:

```
POKE 1784+slot,224
IN# slot
ctrl-A ctrl-F
```

Apples with lower case display capability can receive true ASCII by:

```
POKE 1784+slot,0
IN# slot
ctrl-A ctrl-F
```

PRINTER INTERFACING

The Communications Interface was not designed to drive a printer and a driver routine is required to use it with most printers. The Printer Driver on page 31 of the Communications Interface manual is required when the printer requires a linefeed after a Carriage Return. It was written when the only available DOS was version 3.1. That's why it doesn't work with DOS 3.2 or DOS 3.3. It's easy to fix it, just load the routine into memory, type POKE 845,110, and save the routine to tape or disk using the instructions in the manual.

You can also modify the Printer Driver routine to print at 110 baud for slower printers. All you have to do is type POKE 802,82: POKE 818,82 and save the routine as before.

TAB does not work properly with this interface. The Integer Basic version is limited to 40 columns and an Applesoft TAB(20) will sometimes output 20 spaces instead of going to column 20. Here is a fix that will work with either Basic:

```
10 PRINT A$;: POKE 36,33: PRINT "THERE"  
HI                               THERE
```

SPEEDING UP THE COMMUNICATIONS INTERFACE

There are instructions in the back of the Communications Interface Manual for modifying it to operate at speeds in excess of 300 baud. If you perform one of these modifications the card will still have two speeds. The default speed will be the higher of the two available speeds. The lower speed will be 1/4 of the higher speed and can be used by following the instructions in the manual for operation at 110 baud.

4800 BAUD OPERATION

The modification for 4800/1200 baud in the manual is missing a wire. Add a wire from pin 15 of A1 to pin 15 of A2. Chip A1 still isn't required.

9600 BAUD OPERATION

The Communications Card can be modified to operate at 9600 baud. First perform the 4800/1200 baud modification. Then cut the trace from A1 pin 2 to pin 38 of the edge connector and wire it to pin 37 instead. The default speed for the card is now 9600 baud.

This modification won't work on newer versions of the Communications Card with gold plated edge connectors because there isn't any connector for pin 37.

SCREEN ECHO DURING OUTPUT

The Communications Interface will normally echo whatever is sent out through the interface to the video monitor. The video echo can be stopped by fooling the interface into thinking that it's in full duplex terminal mode. The following program will demonstrate.

```
10 SLOT = 2
20 PRINT CHR$(4);"PR# SLOT"
30 PRINT
40 POKE 1912 + SLOT,145
50 PRINT "THIS WON'T ECHO"
60 POKE 1912 + SLOT,11
70 PRINT "THIS WILL ECHO"
90 PRINT CHR$(4);"PR# 0"
```

APPLESOFT AND DATAMOVER

DATAMOVER will not work with Applesoft. It will require a complete re-write of the program. Applesoft programs can be sent to another Apple, by adding the following lines to your program and running it.

```
0 SPEED = 100
1 SLOT = 2
2 PRINT CHR$(4);"PR#";SLOT
3 PRINT "FP"
4 LIST 6-
5 PRINT CHR$(4);"PR#";SLOT
6 REM YOUR PROGRAM
```

Receiving End: do IN#2 only (do not go into terminal mode)

INITIALIZING WITH POKES

Neither the PR# nor IN# commands in Applesoft and Integer Basic initialize the Communications Interface. This can cause problems for the user who needs to modify the parameters of the interface for his application. He must send a character through the Communications Interface before poking in the new parameters. The following POKES will initialize the memory locations used by the Communications Interface. Please refer to the manual for more information on what each POKE will do. Replace the letter "s" in the list with the slot number in which the interface is installed.

```

10 POKE 1784+s,32      lower case, page 17
20 POKE 1912+s,0      video echo, page 17
30 POKE 2040+s,17     STAT, page 27
40 POKE -16242+s*16,3  reset ACIA, page 27
50 POKE -16242+s*16,17 status, page 27
    
```

The next list of POKES will replace the PR# and IN# commands. These POKES must be used to benefit from the previous POKES. The CALL 1002 should be used if you will be doing DOS commands while the interface is enabled. However, if speed is of the essence, don't use the CALL 1002 until after the data transfer is complete since DOS slows down I/O. These POKES must all be on one command line separated by colons to work in command mode. They can have separate line numbers in a program.

```

60 POKE 54,5          PR#s
70 POKE 55,192+s
80 POKE 56,7          IN#s
90 POKE 57,192+s
100 CALL 1002
    
```

The normal way to reset the I/O to the Apple video and keyboard is:

```

900 D$ = CHR$(4): REM CTRL-D
910 PRINT D$;"PR#0"
920 PRINT D$;"IN#0"
    
```

However, this will only work after a PRINT and will be ignored after a GET or PRINT terminated with a comma or semicolon. To avoid having to do an extra PRINT you can use:

```

900 CALL -375 : REM THIS IS IN#0
910 CALL -365 : REM THIS IS PR#0
920 CALL 1002 : REM THIS RECONNECTS DOS
    
```

SPECIAL NOTE: Don't allow echoing to the Apple's video while sending information to the interface. Your program or variables will suffer if you don't disable the video output for lines more than 40 characters long.

Issued 26 Apr 82

Communications Interface

Page 3300.990.01

ERRATA - COMMUNICATIONS INTERFACE MANUAL |
030-0008-01

Page 17

The lower case info only applies to TERMINAL modes and the method to use it isn't obvious. Also, the POKE 1784+n,160 causes the lower case to be displayed as FLASHING. POKE 1784+n,224 will cause lower case to be INVERSE.

Page 24

DATAMOVER will not work with Applesoft. It will require a complete re-write of the program.

Page 28

The Com Card Print Routine was written when the only available DOS was version 3.1. POKE 845,110 will fix it to work with DOS 3.2 or DOS 3.3.

Page 36

The modification for 4800 baud in the manual is missing a wire. Add a wire from pin 15 of A1 to pin 15 of A2. Chip A1 is still not required.

INDEX TO DOS

ALPHABETIC LISTING

Doc	Title
026	Basic Entry Vectors for DOS
008	Binary File Address and Length Parameters
028	Booting Without a Disk in the Drive
015	BRUN HELLO?
004	Cassette Applesoft and DOS
030	Chain
024	Current Status
005	DOS 3.1 and the Auto-Start ROM
020	DOS 3.3 and the Language Card
007	DOS, the IN# Command, and the PR# Command
016	DOS and Interrupts (IRQ)
018	DOS and Machine Language Routines
012	DOS and the I/O Vectors
009	DOS Commands Being Ignored
010	DOS Diskette Use
023	DOS Text File Format
019	DOS vs Applesoft Hi-Res
003	Dumping DOS Text Files
990	Errata - The DOS Manual
014	Hidding Things Above the DOS Buffers
013	INTBAS and FPBAS on the Master Diskette
011	Muffin
022	Nonstop Catalog
017	Not Direct Command Error
033	Phone List - Printer Problems
032	Renumber and the FRE (0) Command
031	Renumber and the MAXFILES Command
027	Renumber Fix
001	Saving Room when BSAVEing Hi-Res
002	Speeding Up I/O
029	The Program Applesoft on the DOS 3.3 Master Diskette
006	The BASICS Diskette
021	Transferring DOS Text Files to a Printer
025	Which Version of DOS is This?

NUMERIC LISTING

Doc	Issue Date	Previous Date	Pages
000	MAY 12 82	SEP 30 81	2
001	APR 28 82	SEP 29 81	1
002	SEP 29 81	-	1
003	SEP 29 81	-	1

(Continued)

APPLE COMPUTER TECHNICAL NOTES

Page 3700.000.02

DOS

Issued 12 May 82

004	SEP 29 81	-	1
005	SEP 29 81	-	1
006	SEP 29 81	-	1
007	SEP 29 81	-	1
008	SEP 29 81	-	1
009	SEP 29 81	-	1
010	SEP 29 81	-	1
011	SEP 29 81	-	1
012	SEP 29 81	-	1
013	SEP 29 81	-	1
014	SEP 29 81	-	1
015	SEP 21 81	-	1
016	SEP 28 81	-	1
017	SEP 29 81	-	1
018	SEP 29 81	-	1
019	APR 26 82	SEP 29 81	1
020	SEP 29 81	-	1
021	APR 28 82	SEP 29 81	1
022	SEP 29 81	-	1
023	SEP 29 81	-	1
024	SEP 29 81	-	1
025	SEP 29 81	-	1
026	SEP 29 81	-	1
027	SEP 21 81	-	1
028	SEP 29 81	-	1
029	SEP 28 81	-	1
030	SEP 29 81	-	1
031	SEP 25 81	-	1
032	SEP 25 81	-	1
033	NOV 19 81	-	1
990	APR 26 82	SEP 28 81	2

Issued 28 Apr 82

DOS

Page 3700.001.01

SAVING ROOM WHEN BSAVING HI-RES |

One sector can be saved on the disk for each Hi-Res screen that you save if |
you save it with a length of \$1FF8 instead of \$2000. The reason is that
the starting address and length of the binary file are stored as the first
4 bytes of the file. Saving a file with a length of \$2000 actually stores
\$2004 bytes and requires another sector. The syntax will now be:

```
10 D$ = CHR$(4)
100 PRINT D$;"BSAVE HIRES-1, A$2000, L$1FF8"
200 PRINT D$;"BSAVE HIRES-2, A$4000, L$1FF8"
```

SPEEDING UP I/O

All Apple input and output goes through DOS if DOS is booted. This can slow down transactions to the point of losing data when communicating with fast devices. Some applications will work if the slight time delay that DOS introduces is removed from the transaction. In the following example, line 10 disconnects DOS, lines 20 and 30 are the I/O transaction, and line 40 reconnects DOS. There is more information on this subject starting on page 100 of The DOS Manual (A2L0036).

```
10 IN# 1: PR# 1
20 PRINT "CONTROL STRING"
30 INPUT "";A$
40 IN# 0: PR# 0 : CALL 1002
```

DUMPING DOS TEXT FILES

There is no direct way to transfer DOS text files to a printer. This program will move text files to any slot desired by the user. It doesn't strip leading spaces or limit you to 239 character strings.

```
100 KB = -16384: KBS = -16368
110 D$ = CHR$(4)
120 HOME
130 INPUT "TEXT FILE NAME? ";FILES$
140 PRINT
150 INPUT "DESTINATION SLOT? ";SL
160 ON ERR GOTO 500
170 PRINT D$;"OPEN ";FILES$
180 PRINT D$;"READ ";FILES$
190 GET A$
200 PR# SL
210 PRINT A$;
220 IF PEEK (KB) <> 155 THEN 190
230 POKE KBS,0
500 PR# 0
510 CALL 1002
520 PRINT
530 PRINT D$;"CLOSE ";FILES$
540 END
```

The program can be stopped before the end of the file has been reached by pressing the escape key.

Issued 29 Sep 81

DOS

Page 3700.004.01

CASSETTE APPLESOFT AND DOS

DOS has provisions for using a RAM based version of Applesoft. This Applesoft was distributed on early DOS diskettes and on cassettes. If you have an Applesoft cassette, there are three things that need to be done to use it with DOS.

```
Boot DOS
LOAD the Applesoft tape
SAVE APPLESOFT
```

The program APPLESOFT will have to be saved on the diskette in the drive whenever any Applesoft program is loaded. Use the FP command instead of RUN APPLESOFT. See pages 28 and 29 of the DOS manual for more information on the FP command.

Issued 29 Sep 81

DOS

Page 3700.005.01

DOS 3.1 AND THE AUTO-START ROM

DOS 3.1 predates the Auto-Start ROM. It doesn't know about the reset vector and start-up byte. So, the system will re-boot the disk if RESET is pressed after booting with DOS 3.1. The reset vector can be modified by the HELLO program to re-enter the current Basic after the RESET key is pressed. Add this line to the HELLO program:

```
10 POKE 1010,208: POKE 1011,3: POKE 1012,166
```

THE BASICS DISKETTE

The BASICS diskette that ships with DOS 3.3 and the Language System is a Pascal diskette. It has a Pascal directory that can be seen from the Pascal Filer but not from DOS.

The DOS 3.3 COPY and COPYA programs will successfully copy the BASICS diskette. Pascal users can copy the BASICS diskette with the Filer like any other Pascal diskette.

The BASICS diskette MUST boot from slot 6, drive 1 because it is a Pascal diskette.

Issued 29 Sep 81

DOS

Page 3700.007.01

DOS, THE IN# COMMAND, AND THE PR# COMMAND

DOS uses the Apple I/O vectors to intercept commands from BASIC. The PR# and IN# statements modify the I/O vectors directly which makes it impossible for DOS to see subsequent commands from BASIC. The usual symptom is that the DOS doesn't work after you turn the printer on or off. Please read pages 100 to 105 of the DOS manual for a complete discussion of these effects and how to avoid them.

Issued 29 Sep 81

DOS

Page 3700.008.01

BINARY FILE ADDRESS AND LENGTH PARAMETERS

The starting address and length of a binary file are loaded into two registers in memory when the file is loaded. Page 144 of The DOS Manual has the addresses to look at for this information. To find the starting address and length of the FID program on the master diskette:

```
BLOAD FID
PRINT "ADDRESS = "; PEEK (43634) + PEEK (43635) * 256
PRINT "LENGTH = "; PEEK (43616) + PEEK (43617) * 256
```

which would return the starting address of 2051 and a length of 4686.

Issued 29 Sep 81

DOS

Page 3700.009.01

DOS COMMANDS BEING IGNORED

The control-D of a DOS command must be the first character on a output line. This means that it must be immediately preceded by a carriage return. If it isn't, the command will be printed out to the screen or the printer and ignored by DOS.

The most common cause for this is having a GET statement or a PRINT statement that ends with a semi-colon or comma as the last I/O operation before the command that gets ignored. Another PRINT with no semi-colon or comma will generate the carriage return required for the next DOS command to be recognized.

Issued 29 Sep 81

DOS

Page 3700.010.01

DOS DISKETTE USAGE

Apple Disk IIs use the bottom surface of the diskette to save the data. The top surface has a "loading head" to keep the diskette in contact with the head.

We don't recommend using both sides of the diskette for data. The loading head collects a lot of dust and dirt during normal operation. Turning the diskette over will expose the data surface to the abrasive affects of this dust and may cause loss of data.

Issued 29 Sep 81

DOS

Page 3700.011.01

MUFFIN

MUFFIN can't convert some 13 sector diskettes because of the software protection scheme used on them. The same diskettes usually won't copy either. These diskettes will have to be used by booting with the BASICS diskette or BRUNning BOOT13 on the DOS master diskette.

DOS AND THE I/O VECTORS

The Apple monitor sends all input and output through a pair of two byte vectors. DOS takes the contents of the vectors and saves them in its own memory and puts pointers to itself in the monitor's I/O vectors. Since both BASICS use the monitor routines for their I/O, DOS can intercept and interpret commands from Applesoft and Integer Basic.

If for some reason the I/O vectors are changed, DOS is no longer able to intercept commands. There is a routine in DOS that will check the monitor I/O vectors to see if they point to DOS. This routine will store the current values in the I/O vectors and put a pointer to DOS in the vector if DOS isn't already there. This routine starts at \$3EA and can be accessed from Basic with CALL 1002. It is most useful when the I/O vectors need to be modified to a custom I/O routine like the Hi-Res Character Generator routine in Contributed Software, volume 3. Here's how to use the Character Generator and DOS:

```
10 POKE 54,0
20 POKE 55,96
30 CALL 1002
40 PRINT CHR$(4);"CATALOG"
50 PRINT CHR$(4);"PR#0"
```

Issued 29 Sep 81

DOS

Page 3700.013.01

INTBAS AND FPBAS ON THE MASTER DISKETTE

The INTBAS and FPBAS files on the master diskette are there to load the language card if there is one in the system when the disk is booted. These files are written to operate in the memory space of the Language Card and will not work anywhere else.

People with an Apple II who want Applesoft need either the Applesoft Firmware Card or the Language Card.

People with an Apple II Plus who want Integer Basic need either the Integer Basic Firmware Card or the Language Card.

Issued 29 Sep 81

DOS

Page 3700.014.01

HIDDING THINGS ABOVE THE DOS BUFFERS

It is possible to position machine language routines between DOS and its first file buffer. Routines hidden this way can't be overwritten by Basic programs. \$9D00 and \$9D01 (48K system) point into the first, highest, buffer. Add the number of pages (256 byte blocks) you need to the contents of \$9D01 and put a \$0 at the address now pointed at and that address + \$25. JSR \$A7D4 will rebuild the DOS file buffers and move HIMEM down for the current type of Basic. The first available address for your routine will be the pointer value + \$2D.

Issued 21 Sep 81

DOS

Page 3700.015.01

BRUN HELLO?

When DOS is booted it will "RUN" the program that was in memory when the diskette was initialized. You can change DOS so that it will "BRUN" the HELLO file instead. In a 48K Apple change the byte at \$9E42 from \$06 to \$34. The following sequence of commands will create a disk that will BRUN HELLO when it is next booted. First put a blank diskette in the drive.

```
CALL -151
9E42:34
3D0G
INIT HELLO
DELETE HELLO
BLOAD BINARY PROGRAM (from another diskette)
BSAVE HELLO (on the new diskette)
```


DOS AND INTERRUPTS (IRQ)

Interrupting the Apple's 6502 while using DOS can be treacherous. DOS uses location \$45 when converting user supplied decimal numeric parameters to hexadecimal for its internal use. The Apple monitor's interrupt handler also uses \$45 to save the 6502 accumulator. So, if an interrupt occurs while DOS is parsing a command, the numeric parameters that DOS uses when it executes the command will be changed. DOS also uses \$45 when displaying the sector length of the files in a catalog so an interrupt can also cause them to be wrong.

DOS uses timing loops during its read and write operation. Interrupting the Read Write Track Sector routines can kill a disk. DOS protects itself from IRQ, but has no control over NMI. Don't use NMI while using DOS.

NOT DIRECT COMMAND ERROR

DOS will sometimes return a NOT DIRECT COMMAND ERROR when sending it commands from machine language routines. When DOS receives certain commands it checks to insure that the current BASIC interpreter is running. It does this so that the user can respond to an INPUT statement with "RUNNING" without trying to run the file "NING". There is a list of what commands are limited in this sense at the bottom of page 122 of the DOS Manual. The following, relocatable machine language routine will insure that DOS is satisfied.

```
LDA #$80
STA $33      This location must not equal $DD
STA $76      This location must not equal $FF
STA $D9      This location must be greater than $7F
RTS
```

DOS AND MACHINE LANGUAGE ROUTINES

There are two ways to use DOS from machine language routines, the Read Write Track Sector routines and the normal DOS commands. Pages 94 to 98 of the DOS manual describes how to use RWTS to access any part of the diskette. Be warned that RWTS will allow you to write anywhere on the disk, even over existing files, the DOS image, and the catalog. The other way is to send the normal DOS commands, one character at a time the normal output channels. The characters need to be sent out in the 6502 accumulator with a JSR to \$FDED. Remember to precede the commands with a carriage return and control-D and follow it with another carriage return.

Here is a short routine to do a catalog.

```

300:LDX #$00
302:LDA $0320,X
305:JSR $FDED
308:INX
309:TAY          ;Check the accumulator for a $00
30A:BNE $0302   ;And don't stop till you get one
30C:RTS

320:0D 04 43 41 54 41 4C 4F 47 0D 00
          C A T A L O G

```

Here is a short-cut if Applesoft is available.

```

300:LDA #$20
302:LDY #$03
304:JSR $DB3A   ;Applesoft's string printing routine
307:RTS

320:0D 04 43 41 54 41 4C 4F 47 0D 00
          C A T A L O G

```

DOS VS APPLESOFT HI-RES

DOS uses and does not restore \$26 and \$27 while reading text files. These are the locations used by the Applesoft Hi-Res routines to save the last plotted point. Programs that mingle DOS commands and HPLOT TO X,Y instructions will not give the expected results because DOS will overwrite the information that Applesoft has put in \$26 and \$27. Here is an example of a program that reads a text file and plots the data it reads.

```
10 D$ = CHR$(4)
20 HGR
30 PRINT D$;"OPEN FILE"
40 PRINT D$;"READ FILE"
50 HPLOT 0,0
60 T1 = PEEK (38): T2 = PEEK (39)
70 INPUT X,Y
80 POKE 38,T1: POKE 39,T2
90 HPLOT TO X,Y
100 IF X + Y > 0 THEN 60           :REM STOP WHEN X+Y=0
110 PRINT D$;"CLOSE FILE"
```

Issued 29 Sep 81

DOS

Page 3700.020.01

DOS 3.3 AND THE LANGUAGE CARD

DOS 3.3 looks at location \$E000 on both the mother board ROMS and on the firmware card to see if the desired version of Basic is available. If there is a Language card and it were to power up matching one of the two signature values, DOS would jump into the random information on the card assuming that it was Basic. To prevent this DOS stores a \$00 at \$E000 on the Language card during its boot process. The machine language instruction that does it is stored on the disk at Track 0, Sector 9, starting at Byte \$D3.

TRANSFERRING DOS TEXT FILES TO A PRINTER

The following program will read a DOS sequential text file and send its contents to the interface in slot 1. It is careful not to strip off any leading spaces.

```
1000 ONERR GOTO 2000
1010 LET D$ = CHR$ (4)
1020 INPUT "FILE NAME? ";F$
1030 PRINT D$;"OPEN ";F$
1040 PRINT D$;"READ ";F$
1050 REM      MAIN READ-PRINT LOOP
1060 GET A$
1070 PR# 1
1080 PRINT A$;
1090 GOTO 1060
2000 REM      FILE DONE, FINISH UP
2010 PR# 0
2020 CALL 1002
2030 PRINT
2040 PRINT D$;"CLOSE"
2050 END
```

Issued 29 Sep 81

DOS

Page 3700.022.01

NONSTOP CATALOG

The CATALOG command stops every 18 lines to allow the user to read the information on the screen before continuing. This pause can be deleted by changing three bytes in memory. In a 48K Apple:

POKE 44601,234

POKE 44602,234

POKE 44603,234

DOS TEXT FILE FORMAT

DOS sequential text files store information as strings of ASCII characters, just as if the disk were a printer. Therefore numeric variables are stored as a string of characters rather than in their internal format. The number of characters required by a given variable, VAR, can be found with:

```
PRINT LEN (STR$ (VAR))
```

The carriage return or other separator that is also sent out after the data will also require a byte, allocate space for it!!!

CURRENT STATUS

The following locations reflect the current status of DOS. Changing these locations may not change what DOS will do next. The addresses assume a 48K Apple.

Address		Description
Dec	Hex	
-18441	B7F7	Last used slot
-18440	B7F8	Last used drive
-21933	AA53,AA54	Character out vector (CSW)
-21931	AA55,AA56	Character in vector (KSW)
-21929	AA57	Current Maxfiles
-21928	AA58	Default Maxfiles
-21922	AA5E	MON vs NOMON status (bit 6 = Command) (bit 5 = Input) (bit 4 = Output)
-21914	AA66,AA67	Volume (read only)
-21912	AA68,AA69	Drive (OK to change)
-21910	AA6A,AA6B	Slot (OK to change)
-21908	AA6C,AA6D	Record length
-21906	AA6E,AA6F	Record number
-21904	AA70,AA71	Byte in record
-21902	AA72,AA73	Address
-21899	AA75-AA93	Filename (\$AA75 = \$0 for CATALOG etc.)
-21834	AAB6	Which BASIC (0=Int) (64=AS ROM) (128=AS RAM)

WHICH VERSION OF DOS IS THIS?

You need to look at two locations to determine which version of DOS the Apple booted with. You need to check both locations to be completely sure. These addresses are for a 48K Apple. Subtract \$4000 (16384) for a 32K system or \$8000 (32768) for a 16K system.

ADDRESSES

DOS	\$9D02 (-25342)	\$BD84 (-17020)
3.1	\$4D (77)	\$D0 (208)
3.2	\$81 (129)	\$D0 (208)
3.2.1	\$81 (129)	\$AE (174)
3.3	\$81 (129)	\$0B (11)

BASIC ENTRY VECTORS FOR DOS

The following table shows the re-entry points DOS uses for the current Basic. You can change these to re-enter your machine language routine after a command or DOS error. These addresses are for a 48K Apple.

Addr	INT	ASROM	ASRAM	Description
\$9D56	36 E8	-	-	Chain entry
\$9D58	E5 24	FC 24	06 25	Run
\$9D5A	E3 E3	65 D8	67 10	Error entry
\$9D5C	00 E0	00 E0	84 1D	Initialize Basic memory
\$9D5E	03 E0	3C D4	3C 0C	Continue after DOS commands
\$9D60	-	F2 D4	F2 0C	Line link resequence

RENUMBER FIX

The Renumber program on the master diskette has an error. Sometimes it mistakes the number after a "*" as a line number and rennumbers it. So if you had a line

```
10 LET A=B*10
```

it might become

```
20 LET A=B*20
```

Here is a permanent fix for the problem:

For RAM Applesoft

(DOS 3.2)

```
LOAD RENUMBER
POKE 14342,172
POKE 14343,171
UNLOCK RENUMBER
SAVE RENUMBER
LOCK RENUMBER
```

For ROM Applesoft

```
LOAD RENUMBER
POKE 4815,172
POKE 4816,171
UNLOCK RENUMBER
SAVE RENUMBER
LOCK RENUMBER
```

(DOS 3.3)

```
LOAD RENUMBER
POKE 14316,172
POKE 14317,171
UNLOCK RENUMBER
SAVE RENUMBER
LOCK RENUMBER
```

```
LOAD RENUMBER
POKE 4789,172
POKE 4790,171
UNLOCK RENUMBER
SAVE RENUMBER
LOCK RENUMBER
```

Issued 29 Sep 81

DOS

Page 3700.028.01

BOOTING WITHOUT A DISK IN THE DRIVE

The disk drive will run continuously if there isn't a readable diskette in place. This includes trying to boot a 13 sector diskette on a 16 sector Apple and vice versa. It's OK to insert a disk while the drive is running. The only other way to stop the drive is to turn off the power or press RESET (or control-RESET).

Issued 28 Sep 81

DOS

Page 3700.029.01

THE PROGRAM APPLESOFT ON THE DOS 3.3 MASTER DISKETTE

The program APPLESOFT on the DOS 3.3 master diskette is not the Applesoft language. It's a program to load the Applesoft language into your language card if you have one. If you type FP, DOS will run the program called Applesoft on the master diskette which will check for a language card, load the language card with Applesoft from the file FPBAS if there is one, and leave you in Integer Basic command mode. You can replace the APPLESOFT on your diskette with the APPLESOFT MUFFINED from a DOS 3.2 diskette and use the FP command as per the manual.

CHAIN

The DOS master diskette contains a utility program called CHAIN. There are instructions on its use on page 106 of The DOS Manual. There are a few things to look out for though.

CHAIN does an effective PR#0 and IN#0 when it is used. This will disable an 80 column card or a Hi-Res character generator. The first program can save four memory locations that, when the second program restores them, will leave the input and output vectors unchanged.

```

63000 REM      PREPARE TO CHAIN
63010 J = PEEK (999) + PEEK (1000) * 256 - 110
63020 FOR K = 0 TO 3
63030 A(K) = PEEK (J+K)
63040 NEXT K
63050 REM      CHAIN NOW

```

```

10 REM      PROGRAM #2
20 FOR K = 0 TO 3
30 POKE J+K,A(K)
40 NEXT K
50 REM      REST OF THE PROGRAM

```

CHAIN overlays the existing Applesoft program with the new program. Watch out for programs that use DEF FN and ONERR GOTO statements. You will have to re-execute these statements in the new program if they are to work properly. Both statements contain pointers into the program text and the text probably isn't the same in the new program.

Issued 25 Sep 81

DOS

Page 3700.031.01

RENUMBER AND THE MAXFILES COMMAND

Applesoft Renumber won't work with a MAXFILES other than 3. Renumber is a machine language program that is loaded as an Applesoft program. When the Applesoft program Renumber is run it relocates the machine language up to HIMEM in your system. DOS defaults to MAXFILES 3 so HIMEM is on a page boundary, \$9600 in a 48K Apple. But changing MAXFILES moves HIMEM off the page boundary. Renumber's relocater will still do the move, but it doesn't relocate the address properly. The usual symptom is to drop into the monitor with an address in the range \$0200 - \$0300.

Issued 25 Sep 81

DOS

Page 3700.032.01

RENUMBER AND THE FRE (Ø) COMMAND

Renumber doesn't always reset the variable pointers when it returns to Applesoft. This will cause the Apple to hang or drop into the monitor when the PRINT FRE (Ø) command is used. To check the amount of free memory left after a Renumber operation simply do a CLEAR before the PRINT FRE (Ø).

Issued 19 Nov 81

DOS

Page 3700.033.01

PHONE LIST - PRINTER PROBLEMS

The PHONE LIST program provided on the DOS 3.2 Plus Master Diskette will not print correctly on the Silentype printer. This can be corrected by adding the following line to the program:

```
1111 IF PR THEN PRINT
```

Issued 26 Apr 82

DOS

Page 3700.990.01

ERRATA - THE DOS MANUAL
A2L0036
030-0115-00

Page 41

The last item in the table should refer to MASTER CREATE instead of MASTER UPDATE.

Page 62

Please note that in both line 100 and 110 there are no characters between the quotes.

Page 103

Note 3 refers to Table I which is actually TABLE 1 in the manual.

Page 105

Note 7 refers to Table II which is actually TABLE 2 in the manual.

Page 106

Applesoft CHAIN does a IN#0 and PR#0. It will also reset LOMEM: to the top of the new program.

Page 130

Relative byte 22 of a directory entry is always a \$00.

Page 141

The label on the left that says "Lowest RAM memory address:" should be at the bottom of the diagram and the arrow where it is now should read "1024 (\$0400)".

Page 156

The FP command assumes that you have a Language card, an Apple II Plus, an Applesoft firmware card, or the old RAM Applesoft on the diskette in the last used drive. We no longer offer RAM Applesoft. The FPBAS file on the Master Diskette will only work with a Language Card.

Page 157

The first and third paragraphs are the same.

(Continued)

APPLE TECH NOTES

Copyright (C) 1982 by Apple Computer, Inc.

"AppleIACTNJuly82_3700-990-01" 102 KB 1962-10-11 dpi: 300h x 300v pix: 1964h x 2997v

Page 166

RECOVERING FROM ACCIDENTAL RESETS, "If DOS has been booted and dhen..." should say "If DOS has been booted and then...".

Page 171

CHAINING IN APPLESOFT Should read "To RUN" instead of "To REN".

Page 178

The first paragraph mentions LOADAPA and the "&S" command which aren't part of the DOS package. LOADAPA is a part of the DOS Tool Kit. There is a program on page 151 of the DOS Manual that will allow you to see control characters.

INDEX TO DOS TOOL KIT

ALPHABETIC LISTING

Doc	Title
011	ANIMATRIX - Paddles Required
010	APA - XREF and DATA Statements
005	EDASM - Accumulator Addressing Mode
009	EDASM - Disabling "PRESS SPACE TO CONTINUE"
004	EDASM - Indexed Indirect Addressing Error
001	EDASM - PR# Command and Page Length
006	EDASM - Printer Compatibility
003	EDASM - Printing Text Files
008	EDASM - RLOAD and String Variables
007	EDASM - RLOAD File Name Restrictions
002	EDASM - Using Relocatable Files
991	Errata - Apple 6502 Assembler/Editor Manual
990	Errata - The DOS Tool Kit Manual
012	HRCG - Switching It On and Off

NUMERIC LISTING

Doc	Issue Date	Previous Date	Pages
000	MAY 12 82	SEP 30 81	1
001	SEP 30 81	-	1
002	SEP 30 81	-	1
003	SEP 30 81	-	1
004	SEP 30 81	-	1
005	SEP 30 81	-	1
006	SEP 30 81	-	1
007	SEP 30 81	-	1
008	SEP 30 81	-	1
009	SEP 30 81	-	1
010	SEP 30 81	-	1
011	SEP 30 81	-	1
012	SEP 28 81	-	1
990	APR 26 82	SEP 30 81	1
991	APR 28 82	SEP 30 81	2

Issued 30 Sep 81

DOS Tool Kit

Page 3800.001.01

EDASM - PAGE LENGTH IN THE EDASM PR# COMMAND

The PR# command is supposed to allow setting the physical and printable page length for your printer, but it doesn't work properly. It loses linefeeds and the listing will advance up the page. If you let EDASM default to formfeeds, it will print properly on most printers.

EDASM - USING RELOCATABLE FILES

Relocatable files work if the REL is the first thing in the file and you include an ORG after it. The address in the ORG doesn't matter.

```

        REL
        ORG      $300      * THIS NUMBER IS REQUIRED BUT IGNORED
TEXT    EQU      $400
START  LDA      #$00
        STA      TEXT
        LDA      #$01
        STA      TEXT + 1
        RTS
    
```

This example creates a relocatable file that puts an inverse "@" and "A" in the upper left corner of the screen.

Issued 30 Sep 81

DOS Tool Kit

Page 3800.003.01

EDASM - PRINTING TEXT FILES

EDASM is a great way to edit DOS text files, but the LIST and PRINT commands output to the screen. However, they can be diverted to the printer if the printer is activated with a direct DOS command.

```
.PR#1  
LIST  
.PR#0
```

This will transmit the listing of the file to the printer just as it would normally go to the screen. The PR#0 will be printed on the paper. Pressing the RESET key will turn off the printer and return to the editor if the Apple has an Auto-Start ROM.

EDASM - INDEXED INDIRECT ADDRESSING ERROR

A single character argument to an indexed indirect addressing mode statement will assemble as absolute indexed instead. EDASM will assemble "LDA (0,X)" as "9D 00 00" instead of "A1 00". Adding a leading zero or a "\$" to an address or using a multi-character label fixes it.

L	EQU	\$3		
LL	EQU	\$4		
	LDA	(1,X)	;9D 01 00	WRONG
	LDA	(02,X)	;A1 02	RIGHT
	LDA	(L,X)	;9D 03 00	WRONG
	LDA	(LL,X)	;A1 04	RIGHT
	LDA	(\$5,X)	;A1 05	RIGHT

Issued 30 Sep 81

DOS Tool Kit

Page 3800.005.01

EDASM - ACCUMULATOR ADDRESSING MODE

EDASM requires that accumulator addressing mode instructions have an "A" as an argument. For example, "LSR A" is the proper syntax for "4A".

Issued 30 Sep 81

DOS Tool Kit

Page 3800.006.01

EDASM - PRINTER COMPATIBILITY

EDASM assumes that the printer will do a carriage return as a part of any formfeeds it generates during an assembly listing. Some printers leave the print head at the last used character position and start printing the next heading at mid-page.

Issued 30 Sep 81

DOS Tool Kit

Page 3800.007.01

EDASM - RLOAD FILE NAME RESTRICTIONS

RLOAD ignores spaces in the file name you give it. The statement

30 ADRS=USR (0), "MY MODULE,S6,D1"

will look for the relocatable file "MYMODULE" on slot 6, drive 1.

Issued 30 Sep 81

DOS Tool Kit

Page 3800.008.01

EDASM - RLOAD AND STRING VARIABLES

The relocating loader is supposed to put the binary routine just below the current HIMEM in memory and set HIMEM to just below the binary routine. It does lower HIMEM but it doesn't adjust the start of string storage pointer.

Strings that are defined before using RLOAD may be over written and string variables used in the same program after using RLOAD may overwrite the binary information.

One way to guarantee that strings won't cause problems is to have a short program do the RLOAD and then RUN or CHAIN your application program.

Issued 30 Sep 81

DOS Tool Kit

Page 3800.009.01

EDASM - DISABLING "PRESS SPACE TO CONTINUE"

One byte needs to be changed to kill the "PRESS SPACE TO CONTINUE" message.

```
BLOAD ASSM
POKE 9202,96
UNLOCK ASSM
BSAVE ASSM, A$ 1200, L$ 22FB
LOCK ASSM
```

Issued 30 Sep 81

DOS Tool Kit

Page 3800.010.01

APA - XREF AND DATA STATEMENTS

The Applesoft variable command, &X, will hang if the program ends with a DATA statement. The XREF program is looking for a ":" to terminate the DATA statement. Adding a ":" to the end of each DATA statement will fix the &X problem.

Issued 30 Sep 81

DOS Tool Kit

Page 3800.011.01

ANIMATRIX - PADDLES REQUIRED

Animatrix has no provision for controlling the grid cursor from the keyboard.

HRCG - SWITCHING IT ON AND OFF

The Hi-Res Character Generator can be turned on and off under software control. The following two subroutines will switch back and forth. The variable "ADRS" is assumed to be assigned the same as in the subroutine starting at line 2000 of MAXWELL on the Tool Kit diskette.

```
3000 REM          TURN OFF HRCG
3010 PRINT CHR$(4);"PR#0"
3020 PRINT CHR$(4);"IN#0"
3030 HOME
3040 TEXT
3050 RETURN

4000 REM          TURN ON HRCG
4010 CALL ADRS + 3
4020 RETURN
```

Beware, the HRCG overwrites the text screen when outputting to the Hi-Res screen. It will destroy any information there. You will have to rewrite the text screen after using HRCG.

Issued 26 Apr 82

DOS Tool Kit

Page 3800.990.01

ERRATA - APPLESOFT TOOLKIT MANUAL
A2L0038
030-0113-00

Page 11

If you want your program ... change line 180 to:

180 CALL ADRS + 3

Issued 28 Apr 82

DOS Tool Kit

Page 3800.991.01

ERRATA - APPLE 6502 ASSEMBLER/EDITOR MANUAL

030-0112-00

A2L0039

Page 7

"In the following descriptions and examples, the horizontal line" should read "...the vertical line"

Page 17

The end or the first paragraph should read "end of the range." instead of "end of the line."

Page 19

Paragraph 4 starts out "Back up your diskettes" when it means "Back up your files". It also should state that you use a different file name each time you back up your file.

Page 20

The description for TLOAD mentions the appendix "Editor Tape Formats" which doesn't exist. See the TSAVE command.

Page 20

Paragraph 4, the tape format for TSAVE is different than Basic programs. It is similar to Integer Basic but not Applesoft. The files can not be loaded directly anyway because of the tokenizing of key words that both Basics do.

Page 20

The TSAVE format should be:

(10-second leader)
(16-bit file length)
(10-second leader)
(file data)

Page 25

The default tab settings are 14,19,29

Page 28

Paragraph 6, "Commands Available During Assembly" actually appear on page 30 as "Assembly Mode Commands"

(Continued)

APPLE TECH NOTES

Copyright (C) 1982 by Apple Computer, Inc.

"AppleACTNJuly82_3800-991-01" 105 KB 1962-10-11 dpi: 300h x 300v pix: 1970h x 2979v

Page 30

The section, "Assembly Mode Commands", doesn't mention that the escape key will stop the listing like the space bar and switch the display to the right half of the virtual 80-column display. The space bar is used to reset the display to the left half.

Page 31

The second paragraph should start "The Listing OFF Command..." instead of "The Listing NO Command..."

Page 32

The discussion of LABEL mixes the terms "Label" and "Symbol".

Page 32

The last line refers to the appendix on Object File Formats. This information is on Page 48 and 49 under "Assembler Directive Summary" and "Operation Code Summary".

Page 33

Paragraph 2 line 6 refers to the syntax summary on page 49 when the information is on page 48.

Page 60

Sector	Byte (Hex)	Contents of byte
1 to N	6 to c1+5	Binary code image, of length in bytes 4 and 5 above
	c1+6	Begin Relocation Dictionary, which consists of N 4 bytes entries N is variable (0 to ?)

 Issued 12 May 82

Elementary, My Dear Apple

Page 4100.000.01

INDEX TO ELEMENTARY, MY DEAR APPLE

ALPHABETIC LISTING

Doc	Title
002	Elementary, My Dear Apple and the Apple ///
990	Errata - Elementary, My Dear Apple Manual
001	Lemonade's Music

NUMERIC LISTING

Doc	Issue Date	Previous Date	Pages
000	MAY 12 82	SEP 30 81	1
001	SEP 30 81	-	1
002	SEP 25 81	-	1
990	APR 26 82	SEP 30 81	1

Issued 30 Sep 81

Elementary, My Dear Apple

Page 4100.001.01

LEMONADE'S MUSIC

There is no provision to turn off the music in the Lemonade program.

Issued 25 Sep 81

Elementary, My Dear Apple

Page 4100.002.01

ELEMENTARY, MY DEAR APPLE AND THE APPLE ///

Elementary, My Dear Apple will not work on the Apple /// in Apple II
emulation mode.

Issued 26 Apr 82

Elementary, My Dear Apple

Page 4100.990.01

ERRATA - ELEMENTARY, MY DEAR APPLE MANUAL
030-0114-00

Page 2

It is stated that when a blinking cursor and a prompt is shown, one can press the RESET key and you will always return to the main menu. There is a possibility that RESET will not return to the menu. In that case the system must be re-booted.

Page 5

In the paragraph beginning " If you press a wrong key...." it says to go back and correct errors using the forward arrow key. It should say the back or left arrow key.

Page 6

After one is in the immediate mode, it is impossible to get back to the main menu without typing RUN INDEX after the prompt and the flashing cursor.

Page 7

There is no mention that in the LEMONADE game you can use the arrow keys to correct mistakes.

Page 18

Paragraph 3, if you delete the entire word list with DEL 9010,9999 you should add line 4325 IF I = 9010 THEN 4332 to avoid problems with the problem editor.

Issued 12 May 82

Formulex

Page 4200.000.01

INDEX TO FORMULEX

ALPHABETIC LISTING

Doc	Title
001	Formulex and the Apple ///

NUMERIC LISTING

Doc	Issue Date	Previous Date	Pages
000	MAY 12 82	OCT 28 81	1
001	OCT 28 81	-	1

Issued 28 Oct 81

Formulex

Page 4200.001.01

FORMULEX AND THE APPLE ///

The Formulex package won't work on the Apple /// in emulation mode because it is written in Pascal and there is no Language Card in the Apple ///.

INDEX TO FORTTRAN

ALPHABETIC LISTING

Doc	Title
004	Available Memory
010	Compiler Swapping in FORTRAN
007	Creating FORTRAN Data Files
005	Data File Limits
990	Errata - FORTRAN Language Reference Manual
006	FORT2 Copy-Protection
002	FORTTRAN System Requirements
008	Making the Most of Your Disk Space
003	Unformatted I/O in Apple FORTRAN
003	Unformatted I/O in Apple FORTRAN
009	Using Library Units in FORTRAN with Pascal 1.1
001	Using WCHAR from FORTRAN

NUMERIC LISTING

Doc	Issue Date	Previous Date	Pages
000	JUN 1 82	MAY 12 82	1
001	SEP 28 81	-	1
002	SEP 30 81	-	1
003	MAY 14 82	SEP 30 81	1
003	SEP 30 81	-	1
004	SEP 30 81	-	1
005	OCT 29 81	-	1
006	SEP 30 81	-	1
007	OCT 29 81	-	1
008	OCT 29 81	-	1
009	MAY 24 82	-	1
010	OCT 29 81	-	1
990	APR 26 82	SEP 30 81	1

USING WCHAR FROM FORTRAN

The Turtlegraphics WCHAR procedure, which writes a single character on the hi-res screen, will often give a "Value Range Error" (S#20, P#17, I#11) when called from FORTRAN. WCHAR does not check the high byte of the character word passed before calling the DRAWHBLOCK routine, so if the high byte contains a value, WCHAR will choke by trying to access an index beyond the end of SYSTEM.CHARSET.

This error will occur when passing a single character from a character array, but not from a single character "string". For example

```
CHARACTER CH(10)
READ (*,100) CH(1)
CALL WCHAR (CH(1))
```

will result in a "Value Range Error". However,

```
CHARACTER*1 CH
READ (*,100) CH
CALL WCHAR (CH)
```

will work correctly. To prevent the error in the first example, replace the call statement with

```
CALL WCHAR (CHAR (ICHAR ( CH(1))))
```

which will convert the character to a integer, and then back to a character before calling WCHAR.

Issued 30 Sep 81

FORTRAN

Page 4300.002.01

FORTRAN SYSTEM REQUIREMENTS

The Apple Fortran system requires an Apple II or Apple II Plus with 48K of memory, the language system, and at least one disk drive. Use of two drives is recommended for ease of operation and for serious program development.

Issued 30 Sep 81

FORTRAN

Page 4300.003.01

UNFORMATTED I/O IN APPLE FORTRAN

Unformatted I/O is not supported in Apple FORTRAN, as it isn't part of the 1977 ANSI Subset. Statements of the form PRINT*, READ*, or WRITE* are not allowed. All I/O is of the form READ or WRITE (<unit number>, <format identifier>) <iolist>.

Issued 30 Sep 81

FORTRAN

Page 4300.004.01

AVAILABLE MEMORY

Apple FORTRAN runs under the Pascal operating system, and has approximately 37K available for user programs and data in version 1.0, and over 39K in 1.1, with Swapping toggled on. Please refer to the Pascal 1.1 Update (addendum) for information on the Swapping option.

Issued 29 Oct 81

FORTTRAN

Page 4300.005.01

DATA FILE LIMITS

Apple FORTRAN will allow a total of 16 files to be opened within a program. This is based on the number of file identifier blocks available within the Pascal operating system. However, available buffer space (memory) will more likely be the limiting factor in your FORTRAN program.

Issued 30 Sep 81

FORTTRAN

Page 4300.006.01

FORT2 COPY-PROTECTION

The Apple FORTRAN compiler is protected and cannot be copied. A Bad Block scan of the FORT2: diskette will show blocks 30, 31, and 32 as bad. DO NOT ATTEMPT TO "FIX" THESE BLOCKS AS THIS WILL RUIN THE FORT2: DISKETTE!! See your local service center if your FORT2: diskette doesn't work.

CREATING FORTRAN DATA FILES

An error has been uncovered in the FORTRAN "CLOSE" statement which causes the operating system to mishandle the disk file space. If a data file is created within a program, closed, then reopened, the system will report that there is no room left on the volume, even if little data was actually written to the file. This is because the OPEN statement, with STATUS="NEW", attempts to reserve all the unused blocks in the largest available space, and the CLOSE does not release the unused blocks when the file is closed.

One way to avoid this problem is to "Make" a file on the given disk, of the size you will be needing. For example, your program might be creating a data file (by the name of "MYFILE") which will eventually occupy 100 blocks on your data disk (let's call the disk "DATA"). From the Filer, type "M" (for Make), followed by "DATA:MYFILE[100]". This will create the directory entry, reserving 100 blocks for the data file. The FORTRAN program can then OPEN the data file with STATUS="OLD", and the space will be managed correctly. When all the blocks are filled, the expected error messages will occur.

MAKING THE MOST OF YOUR DISK SPACE

When using FORTRAN in a multiple-drive system, it is not necessary to duplicate system files on both FORT1: and FORT2:. Since the SYSTEM.COMPILER cannot be copied, FORT2: will by necessity have at least one file, but the majority of the space can be free for program development. Follow the procedure for formatting diskettes and transferring files which is described in the Pascal or FORTRAN reference manuals.

Here are two possible FORTRAN configurations which can be used. Your setup will depend on your personal tastes. If you normally do not use, or intend to use, the system work file, you may wish to use this configuration:

FORT1: (boot disk)	FORT2:
SYSTEM.APPLE	SYSTEM.COMPILER
SYSTEM.PASCAL	
SYSTEM.MISCINFO	
SYSTEM.CHARSET	
SYSTEM.EDITOR	
SYSTEM.FILER	
SYSTEM.LINKER	
SYSTEM.LIBRARY	
FORTLIB.CODE	

With this configuration, FORT1: will have only 26 unused blocks, but FORT2: will have 182 blocks available for text and code files! If you exit the Editor by Writing a named file to FORT2: instead of Updating SYSTEM.WRK, you'll have plenty of room to compile and link your FORTRAN programs.

If you plan to use the system work file, use the following configuration, which will leave the majority of free space on the boot disk. This setup may also be used for writing files, of course. The remaining space on FORT2 can be used for storing files not being used.

FORT1: (boot disk)	FORT2:
SYSTEM.APPLE	SYSTEM.COMPILER
SYSTEM.PASCAL	SYSTEM.LINKER
SYSTEM.MISCINFO	SYSTEM.EDITOR
SYSTEM.LIBRARY	SYSTEM.FILER
	SYSTEM.CHARSET
	FORTLIB.CODE

For single-drive systems, please consult your FORTRAN reference manual.

USING LIBRARY UNITS IN FORTRAN WITH PASCAL 1.1

FORTRAN, running under Pascal version 1.1, does not access the SYSTEM.LIBRARY in order to find the location of each library segment. Therefore, in FORTRAN programs which use library units (\$USES <filename>), a stack overflow will occur almost immediately after execution begins.

A program which will repair the segment dictionary of a FORTRAN code file is included on the FORT2 diskettes. If your FORTRAN package does not include this program, you may get a copy from your dealer or regional support center, or from the International Apple Core, on their April 1981 Disk of the Month (called ATTACH). This Pascal program should be used after compiling and linking, but prior to the first execution of each FORTRAN program.

Another method of forcing the operating system to load the locations of each segment is to compile the following Pascal program:

```
PROGRAM READTABLE;  
  
USES TURTLEGRAPHICS; (* or any other intrinsic unit *)  
  
BEGIN  
END.
```

Place the code file on your FORTRAN boot disk, and name that code file SYSTEM.STARTUP. The program will be executed automatically during each boot and will cause the table to be read in.

Issued 29 Oct 81

FORTRAN

Page 4300.010.01

COMPILER SWAPPING IN FORTRAN

The FORTRAN compiler does not have a swapping option. Programs which fail to compile by exceeding the size of the symbol table will have to be reduced in size, or broken into several parts and compiled separately. Apple FORTRAN will allow compilation of program sections through the \$EXT and \$USES compiler directives. Please refer to the FORTRAN Language Reference Manual for more details.

 Issued 26 Apr 82

FORTRAN

Page 4300.990.01

ERRATA - FORTRAN LANGUAGE REFERENCE MANUAL
 A2L0032
 030-0118-00

Page inside cover

The number A2D0032 is for the diskette, not the manual. The correct number is A2L0032.

Page ix

The page number for the Appendices should be 135 instead of 134.

Page xi

The top paragraph should be deleted because the information is presented on page x.

Page 156

In a multi-drive system it isn't necessary to duplicate the system files on FORT2:. It's better to leave FORT2: empty except for the SYSTEM.COMPIILER (which can't be copied anyway) and W)rite your files to FORT2:<filename> when you leave the Editor and when you compile.

Slot 6 Drive 1

FORT1:
 SYSTEM.APPLE
 SYSTEM.PASCAL
 SYSTEM.MISCINFO
 SYSTEM.CHARSET
 SYSTEM.FILER
 SYSTEM.EDITOR
 SYSTEM.LIBRARY
 SYSTEM.LINKER
 FORTLIB.CODE

Slot 6 Drive 2

FORT2:
 SYSTEM.COMPIILER

 (your files)

Issued 12 May 82

Goodspell

Page 4600.000.01

INDEX TO GOODSPELL

ALPHABETIC LISTING

Doc	Title
001	Goodspell's User Dictionary

NUMERIC LISTING

Doc	Issue Date	Previous Date	Pages
000	MAY 12 82	SEP 28 81	1
001	SEP 21 81	-	1

Issued 21 Sep 81

Goodspell

Page 4600.001.01

GOODSPELL'S USER DICTIONARY

The user dictionary built while Goodspell is checking a file is kept in the Apple's memory and can't be saved to disk.

 Issued 12 May 82

Graphics Tablet

Page 4700.000.01

INDEX TO GRAPHICS TABLET

ALPHABETIC LISTING

Doc	Title
001	Area Calculation Errors
002	Circle and Disk Mode
990	Errata - Graphics Tablet Manual

NUMERIC LISTING

Doc	Issue Date	Previous Date	Pages
000	MAY 12 82	SEP 28 81	1
001	SEP 21 81	-	1
002	SEP 18 81	-	1
990	APR 26 82	SEP 28 81	1

Issued 21 Sep 81

Graphics Tablet

Page 4700.001.01

AREA CALCULATION ERRORS

The area calculation for the graphics tablet is not totally accurate. It was not designed to be used in critical situations, but rather as an example of the way such a tool could be used. The code is extremely simple, and the inaccuracies in Basic could account for the error encountered.

Issued 18 Sep 81

Graphics Tablet

Page 4700.002.01

CIRCLE AND DISK MODE

The CIRCLE and DISC routines in the graphics tablet manual do work, although the circle calculation is not accurate. It generates a circle with a radius larger than defined by the points plotted by the user.

Issued 26 Apr 82

Graphics Tablet

Page 4700.990.01

ERRATA - GRAPHICS TABLET MANUAL
030-0076-00

Page 26

The first word on line 5 should read "initialized" instead of
"uninitialized".

Page 55

The following lines are wrong:

2475 GOTO 2340 should be 2475 GOTO 2350

2490 GOTO 2340 should be 2490 GOTO 2348

Issued 1 Jun 82

Hand Holding Basic

Page 4800.000.01

INDEX TO HAND HOLDING BASIC

ALPHABETIC LISTING

Doc	Title
002	Running the Demo
001	String Arrays

NUMERIC LISTING

Doc	Issue Date	Previous Date	Pages
000	JUN 1 82	MAY 12 82	1
001	NOV 23 81	-	1
002	MAY 21 82	-	1

Issued 23 Nov 81

Hand Holding Basic

Page 4800.001.01

STRING ARRAYS

Hand Holding Basic doesn't support string arrays. The program acts as if
if were a syntax error and prompts for an acceptable character.

Issued 21 May 82

Hand Holding Basic

Page 4800.002.01

RUNNING THE DEMO

Instructions were missing on how to run the Demo Program in the early version of the manual. To run the Demo Program, get into Level 4 (see the instructions are on page 46) and type in

ROLLING DEMO

INDEX TO HIGH SPEED SERIAL INTERFACE

ALPHABETIC LISTING

Doc	Title
003	Apple Video and Printing
012	Batch move routines and Applesoft
011	Current loop operation
991	Errata - Addendum to the Serial Interface Card Manual
990	Errata - Serial Interface Card Manual
009	Initializing with POKES
005	The P7-04 PROM
004	The P8A PROM
008	Problem with Applesoft's GET statement
010	Remote Operation of an Apple in Basic
006	Serial Card Space-on-Reset
007	Serial handshake modification
002	Tab Problems
013	Using an Apple with a Serial Card as a terminal
001	Using the Serial Card with the Qume Sprint 5

NUMERIC LISTING

Doc	Issue Date	Previous Date	Pages
000	JUN 1 82	MAY 12 82	1
001	OCT 19 81	-	1
002	OCT 19 81	-	1
003	OCT 19 81	-	1
004	OCT 19 81	-	1
005	OCT 19 81	-	1
006	OCT 19 81	-	1
007	OCT 21 81	-	3
008	OCT 21 81	-	1
009	MAY 21 82	-	2
010	OCT 22 81	-	1
011	OCT 20 81	-	1
012	OCT 20 81	-	1
013	OCT 20 81	-	1
990	OCT 21 81	-	1
991	OCT 21 81	-	1

USING THE SERIAL CARD WITH THE QUME SPRINT 5

Set up the Qume:

Form length :11 inches
 Character spacing :10 or 12
 Twintellect :Standard
 Auto LF :On
 Parity :Mark
 Duplex :Test (For normal operation :Full)
 Baud rate :1200

The two switches inside the cover should be set to "MOD" and "HI".

Set up the serial card:

Be sure that the P8A PROM is installed at location B2
 The switch on the Serial card should be set to;

1 2 3 4 5 6 7
 Off Off On Off On Off On

Enter and run this program

```
10 PR# 1
20 PRINT CHR$ (27); CHR$ (26); CHR$ (14)
30 PR#0
```

The Sprint 5 printer will do the Self Test and print "Self Test OK", then it will print the entire character set until the Duplex switch is set to Full.

TAB PROBLEMS

TAB does not work properly with the High Speed Serial Interface. In Integer Basic it is limited to 40 columns and in Applesoft, TAB(20) will sometimes output 20 spaces instead of going to column 20. Here is an example that gets around this problem.

```
10 PRINT "HI";: POKE 36,55: PRINT "THERE"
```

```
HI
```

```
THERE
```

APPLE VIDEO AND PRINTING

If you are sending information to a printer with the Serial Card's line width set to greater than 40 and echoing the information to the Apple screen, you may wipe out your Applesoft program. Apple's screen is memory mapped and an offset greater than 40 will cause the information to be "printed" in the same memory as the start of your Applesoft program. You can use the printer or the video but not both at the same time.

Issued 19 Oct 81

High Speed Serial Interface

Page 4900.004.01

THE P8A PROM

The P8A prom sends an ASCII ETX (ctrl-C) to the printer at the end of each line and waits for the printer to send back an ASCII ACK (ctrl-F) before allowing the Apple to continue executing its program. Pins 2 & 3 on the interface connector must be connected straight across and the printer must be able to send the ACK or the Apple will stop after the first line that it sends. Printers that support the P8A are:

Anderson Jacobson 832 (send <esc>!W to set the AJ's mode, pg 3-30)
Qume Sprint 5
NEC Spinwriter

The P8A prom uses location \$3C as a temporary memory register. Many of the monitor commands can't be used with the printer due to this conflict.

Issued 19 Oct 81

High Speed Serial Interface

Page 4900.005.01

THE P7-04 PROM

Early versions of the High Speed Serial Card would not work with certain other cards in the next higher numbered peripheral slot. The P7-04 PROM has solved this problem.

Issued 19 Oct 81

High Speed Serial Interface

Page 4900.006.01

SERIAL CARD SPACE-ON-RESET

When driving a printer like the Qume the alarm sounds when the Apple is first turned on. This is because the 74LS109 was wired to send out a "space" instead of a "mark" level when the Apple is RESET.

Swapping pins 1 and 5 on the 74LS109 will solve the problem. It can be done on the board or at the socket.

Issued 21 Oct 81

High Speed Serial Interface

Page 4900.007.01

SERIAL HANDSHAKE MODIFICATION

INTRODUCTION:

The High Speed Serial Interface card cannot run faster than 300 baud on most printers due to a lack of a printer busy line. This modification uses the existing data input line to sense if the printer is busy and inhibit output if necessary. This modification will work with cassette, DOS 3.2, or DOS 3.3.

WARNING:

Damage to the High Speed Serial Interface card may not be covered by your warranty. If you aren't sure of the signal levels and pinout of your printer, find out or get someone who knows to help you.

WIRING CHANGES:

First you must determine which wire your printer uses to indicate a printer busy or buffer full condition. Your printer's manual should contain this information or contact the manufacturer.

Examples:

IDS 125/225	pin 4
HEATH H-14	pin 4
TI-810	pin 11
SPINTERM	pin 19
COMPRINT	pin 20

The preferred place to do the wiring change is in the cable, but it can also be done at the Serial Card or the printer. Disconnect the wire between pin 2 of the printer and pin 2 on the Serial Card. Then connect the wire with the printer busy signal to the wire for pin 2 on the Serial Card.

SOFTWARE PATCH:

Next you must decide which slot the interface will go in and type in the software patch. The patch is customized for this slot number and won't work if used with a different configuration. The patch forces the computer to look to see if the printer is busy and wait if it is. Enter the patch using the values from the table for words in brackets, < >.

(Continued)

APPLE TECH NOTES

Copyright (C) 1981 by Apple Computer, Inc.

"AppleIACTNJuly82_4900-007-01" 134 KB 1962-10-11 dpi: 300h x 300v pix: 1976h x 2993v

APPLE COMPUTER TECHNICAL NOTES

 Page 49000.007.02

High Speed Serial Interface

Issued 21 Oct 81

SLOT	1	2	3	4	5	6	7
DATA	90	A0	B0	C0	D0	E0	F0
CODE	C1	C2	C3	C4	C5	C6	C7

Enter the monitor with CALL -155 and type

```

3B0:A9 <SLOT>
:20 95 FE
:A9 00
:20 ED FD
:A9 C5
:85 36
:A9 03
:85 37
:4C EA 03
:2C <DATA> C0
:30 FB
:4C 07 <CODE>
:00 00 00
  
```

To check your typing, type

```
3B0L
```

and compare your listing to the one below for slot 1.

```

03B0-  A9 01      LDA  #$01
03B2-  20 95 FE   JSR  $FE95
03B5-  A9 00      LDA  #$00
03B7-  20 ED FD   JSR  $FDED
03BA-  A9 C5      LDA  #$C5
03BC-  85 36      STA  $36
03BE-  A9 03      LDA  #$03
03C0-  85 37      STA  $37
03C2-  4C EA 03   JMP  $03EA
03C5-  2C 90 C0   BIT  $C090
03C8-  30 FB      BMI  $03C5
03CA-  4C 07 C1   JMP  $C107
03CB-  00         BRK
03CC-  00         BRK
03CD-  00         BRK
  
```

Now return to basic with 3D0G.

SAVING THE PATCH TO DISK:

(Continued)

APPLE TECH NOTES

Copyright (C) 1981 by Apple Computer, Inc.

"AppleACTNJuly82_4900-007-02" 82 KB 1962-10-11 dpi: 300h x 300v pix: 1963h x 2993v

The patch must be in memory before the printer can be used at the higher speeds. Save the patch by typing

```
BSAVE PATCH, A$3B0, L$20.
```

USING THE PRINTER:

The first time you want to use the printer you must load the patch and initialize the interface. From immediate mode type

```
BLOAD PATCH CALL 944.
```

This may be done in a program, by entering

```
100 PRINT D$;"BLOAD PATCH": CALL 944
```

assuming that D\$ is a control-D.

If in the course of the program you need to turn off the printer, type

```
PR#0
```

or in a program enter

```
200 PRINT D$;"PR#0"
```

Then to reconnect the printer, all that is required is

```
CALL 954
```

or from a program

```
300 CALL 954.
```

NOTES:

If the printer does not print after the CALL 944, it is probably sending the opposite polarity busy signal. The patch can be changed to recognize the opposite polarity signal with

```
POKE 968,16.
```

If this doesn't work, have the printer checked.

The modification allows the speed, column width, and other variables to be changed with the POKES in the manual.

PROBLEM WITH APPLESOFT'S GET STATEMENT

The Serial Card interferes with Applesoft's GET statement while echoing the input back out through the Serial Card. This causes the following program to fail.

```
10 IN#1:PR#1
20 GETA$:PRINTA$;:GOTO20
```

The same program will work if the output is echoed to the Apple's screen.

```
10 IN#1:PR#0
20 GETA$:PRINTA$;:GOTO20
```

This second program will print any lower case input as inverse on the screen.

INITIALIZING WITH POKES

Neither the PR# nor IN# commands in Applesoft and Integer Basic initialize the Serial Card. This can cause problems for the user who needs to modify the parameters of the interface for his application. He must send a character through the Serial Card before poking in the new parameters. The following POKES will initialize the memory locations used by the Serial Card. Please refer to the Serial Card manual for more information on what each POKE will do.

All occurrences of "s" should be replaced by the slot number that the Serial Card is plugged into.

10 POKE 1144+s,64	BRATE, page 21
20 POKE 1272+s,2	STBITS, page 21
30 POKE 1400+s,7	STATUS, page 22
40 POKE 1528+s,0	Character counter
50 POKE 1784+s,80	PWDTH, page 23
60 POKE 1912+s,9	NBITS, page 23
70 POKE 2040+s,129	FLAGS, page 24

The following list of POKES will replace the PR# and IN# commands which is required to benefit from the previous POKES. The CALL 1002 should be used if you will be doing DOS commands while the interface is enabled. However, if speed is of the essence, don't use the CALL 1002 until after the data transfer has been made since DOS does slow down I/O. These POKES must all be on one command line separated by colons to work in command mode. They can have separate line numbers in a program.

80 POKE 54,5	PR#s
90 POKE 55,192+s	
100 POKE 56,7	IN#s
110 POKE 57,192+s	
120 CALL 1002	

The normal way to reset the I/O to the Apple video and keyboard is:

```

900 D$ = CHR$(4): REM CTRL-D
910 PRINT D$;"PR#0"
920 PRINT D$;"IN#0"
    
```

However, this will only work after a PRINT and will be ignored after a GET or PRINT terminated with a comma or semicolon. To avoid having to do an extra PRINT you can use:

```

900 CALL -375 : REM THIS IS IN#0
910 CALL -365 : REM THIS IS PR#0
920 CALL 1002 : REM THIS RECONNECTS DOS
    
```

(Continued)

SPECIAL NOTE: Don't allow echoing to the Apple's video while printing.
Your program or variables will suffer if you don't disable the video output
while printing lines more than 40 characters long.

REMOTE OPERATION OF AN APPLE IN BASIC

Using a terminal and a High Speed Serial card to remotely operate an Apple can be interesting. To enable lower case input from a program you must do one of two things:

Input at least one character in upper case mode before poking the value from page 24 of the manual. For example

```
10 PRINT CHR$(4);"PR#3"
20 PRINT CHR$(4);"IN#3"
30 INPUT "PLEASE HIT RETURN";A$
40 POKE 2043,33
50 REM                                THE REST OF THE PROGRAM
```

The other method is to initialize the card yourself with pokes.

```
100 LET SLOT = 3
110 PR# SLOT : IN# SLOT
120 POKE 54,7
130 POKE 56,5
140 CALL 1002
150 POKE 1144+SLOT,8 : REM Baud Rate (p. 21 and 31)
160 POKE 1272+SLOT,2 : REM Stop Bits (p. 21)
170 POKE 1400+SLOT,7 : REM Parity (p. 22)
180 POKE 1784+SLOT,41 : REM Line Width (p. 23)
190 POKE 1912+SLOT,9 : REM Data Bits (p. 23)
200 POKE 2040+SLOT,33 : REM Operation Modes (p. 24)
210 REM                                THE REST OF THE PROGRAM
```

Issued 20 Oct 81

High Speed Serial Interface

Page 4900.011.01

CURRENT LOOP OPERATION

The High Speed Serial Interface is the only interface we market with a 20ma current loop. It has an active send loop and a passive receive loop.

Page 8 of the manual tells how to connect to an 33ASR teletype. This may not work with other 20ma devices. It assumes that the device has a passive send and receive loop.

If the other device has its own active send loop then connect it as follows

- Connect pin 23 to Printer +
- Connect pin 7 to Printer -
- Connect pin 12 to Keyboard +
- Connect pin 13 to Keyboard -

Issued 20 Oct 81

High Speed Serial Interface

Page 4900.012.01

BATCH MOVE ROUTINES AND APPLESOFT

I have successfully used the batch move routines with Applesoft, both input and output. The only possible error I spotted is that after the PR#1 there must be a PRINT to initialize the interface. Using an IN#1 will require an input.

Issued 20 Oct 81

High Speed Serial Interface

Page 4900.013.01

USING AN APPLE WITH A SERIAL CARD AS A TERMINAL

The Serial Card is not fast enough to be used as a terminal with a modem. It will lose characters from the input channel. The software intensive design of this card makes it unsuitable for high speed, bidirectional communications.

 Issued 21 Oct 81

High Speed Serial Interface

Page 4900.990.01

ERRATA - SERIAL INTERFACE CARD MANUAL
 A2L0008
 030-0012-00

Page 16

The information for switch 4 is reversed. It should read:

ON = delay enabled
 OFF = delay disabled

Page 32

Two labels on the schematic diagram are wrong:

XMIT		XMIT
-----0 B	should be	-----0 C
WIRE		WIRE
TO		TO
PIN 3		PIN 3
RECV		RECV
-----0 C	should be	-----0 B
WIRE		WIRE
TO		TO
PIN 2		PIN 2

Issued 21 Oct 81

High Speed Serial Interface

Page 4900.991.01

ERRATA - ADDENDUM TO THE SERIAL INTERFACE CARD MANUAL
031-0012-00

Note:

The P8A PROM interferes with the operation monitor routines that use \$3C.
The P8A uses this location as a temp and doesn't restore it.

 Issued 12 May 82

Integer Basic

Page 5100.000.01

INDEX TO INTEGER BASIC

ALPHABETIC LISTING

Doc	Title
002	How to PEEK and POKE Location 32768
001	Machine Language Routines and the X register

NUMERIC LISTING

Doc	Issue Date	Previous Date	Pages
000	MAY 12 82	SEP 28 81	1
001	SEP 4 81	-	1
002	SEP 21 81	-	1

Issued 4 Sep 81

Integer Basic

Page 5100.001.01

INTEGER BASIC AND THE X REGISTER IN MACHINE LANGUAGE ROUTINES

Integer Basic uses the X register to keep track of where in the line of code it is. Machine language routines that can be called on from a multi-statement lines should save and restore the X register.

Issued 21 Sep 81

Integer Basic

Page 5100.002.01

HOW TO PEEK AND POKE LOCATION 32768

Integer Basic numbers are limited to the range -32767 to 32767. You can still access location 32768 by using -32767-1.

```
PRINT PEEK (-32767-1)
POKE -32767-1,0
```

 Issued 12 May 82

Integer Basic Firmware Card

Page 5200.000.01

INDEX TO INTEGER BASIC FIRMWARE CARD

ALPHABETIC LISTING

Doc	Title
002	Firmware Card and System Errors
001	Firmware Card Options

NUMERIC LISTING

Doc	Issue Date	Previous Date	Pages
000	MAY 12 82	OCT 21 81	1
001	OCT 1 81	-	1
002	SEP 21 81	-	1

FIRMWARE CARD OPTIONS

The Integer Basic firmware card was designed to supply Integer Basic to Apple II Plus owners. There are three options on the card that the user may select.

The first and most obvious option is the switch on the rear of the card. It is used to force the selection of which bank of ROMs, the firmware card or the motherboard, will be used immediately after a system reset. If the switch is up then the system will default to Integer Basic, otherwise it will default to Applesoft.

This can also be controlled by software by accessing address \$C080 to select Integer Basic and \$C081 to select Applesoft. DOS confuses things by forcing the version of BASIC it was last using every time it gets control, regardless of the position of the switch and DOS gets control immediately after a reset in a system with an Auto Start ROM.

The second option concerns the monitor ROM. Between ICs B3 and B4 there is a solder circle divided into two pads with the label "F8" silk screened under it. If these pads are separated then the monitor ROM on the motherboard will always be used. But if the pads are connected then the old monitor ROM on the card will be used with Integer Basic.

This can be interesting if you have an old monitor with your Integer Basic. If the switch is set to go to Integer Basic then the Apple will drop into the old monitor when you RESET it and while you are in Integer Basic the Auto Start monitor functions won't be available.

The third option also concerns using your own language and monitor. Apple Computer uses 2316 type ROMs for all our firmware. However 2716 type (+5v only version) EPROMs can be used to replace the ROMs on the firmware card. There are two pairs of solder spots that can be connected to reconfigure the card for 2716s. They are in the upper right corner with a box drawn around them and "2716" written over it. The card can only use one type at a time, 2716s and 2316s cannot be mixed.

Issued 21 Sep 81

Integer Basic Firmware Card

Page 5200.002.01

FIRMWARE CARD AND SYSTEM ERRORS

Occasionally, peripheral cards with solder plated edge connectors collect some oxidation on the contact fingers which can cause an intermittent connection. This can result in various unusual and unexpected system errors when it happens to the Integer Basic Firmware Card.

To clean off contacts, turn off the power and remove the cards. Using a soft pencil eraser ("Pink Pearl" or such), gently clean off the contacts. Replace the boards, seat firmly, then reboot the system. If this does not correct the problem, contact your dealer for assistance.

 Issued 12 May 82

Language Card

Page 5500.000.01

INDEX TO LANGUAGE CARD

ALPHABETIC LISTING

Doc	Title
003	Basic program and variable storage
001	Introduction to the BASICS Diskette
002	Using the Old Monitor with the Language Card

NUMERIC LISTING

Doc	Issue Date	Previous Date	Pages
000	MAY 12 82	OCT 22 81	1
001	SEP 28 81	-	1
002	SEP 28 81	-	1
003	OCT 20 81	-	1

Issued 28 Sep 81

Language Card

Page 5500.001.01

INTRODUCTION TO THE BASICS DISKETTE

The Language System includes a diskette marked "BASICS". When you boot with this diskette a program is run that looks to see which BASIC, Applesoft or Integer, is in the machine and then loads the Language card with the other language. This means that whichever BASIC you have, you will have access to the other. In addition, if you have an Apple II Plus, the code for the Programmer's Aid #1 is also loaded into the card with Integer Basic. See your dealer for the Programmer's Aid manual (A2L0011). Regular (non-Plus) Apples may still need to install a Programmer's Aid #1 ROM in socket D0. If there is a ROM with part number 341-0016 then you already have the Programmer's Aid #1.

USING THE OLD MONITOR WITH THE LANGUAGE CARD

The Apple Language Card gives you all the advantages of the Auto Start ROM whether you have an Apple II or Apple II Plus. This is because there is an Auto Start ROM on the Language Card that is used instead of the F8 ROM on the main board. There is no way to disable this ROM if you want to use the old monitor in an Apple II. However, there are still two ways to use the old monitor with the Language Card.

The first and most obvious is to replace the Auto Start ROM on the Language Card with an old monitor ROM. In this case you lose the Auto Start ROM's features when in the Basic that is resident on the main board. The Basic that is loaded into the Language Card will still be working out of an image of the Auto Start ROM that was loaded with the Basic. For example, Applesoft would use the old monitor and Integer Basic would use the Auto start ROM.

The second way will give you the old monitor while in the Basic that's in the Language Card but the Basic on the main board will still access the Auto Start ROM. All you need do is load an image of the old monitor into the Language Card yourself. This is a two step process.

The hard part is to get an image of the old monitor into a DOS binary file. First, boot DOS 3.3 on an Apple II without a Language Card. Then type:

```
INT
BSAVE OLDMON, A$F800, L$800
```

Now all you need is a program or subroutine to load the monitor into the card. The following program fragment must be in Integer Basic if you have an Apple II or Applesoft if you have an Apple II Plus. It can be added to the HELLO or APPLESOFT program on the DOS 3.3 Master diskette.

```
10 D$ = "": REM CONTROL D
20 A = PEEK ( -16255): A = PEEK ( -16255)
30 PRINT D$;"BLOAD OLDMON"
40 A = PEEK ( -16254)
```

The old monitor will stay there until you re-boot or reload the Language card.

Issued 20 Oct 81

Language Card

Page 5500.003.01

BASIC PROGRAM AND VARIABLE STORAGE

The Language Card can't normally be used for Basic program or variable storage. It is mapped into the same memory addresses as the Basic interpreter, so when you turn on the Language Card the Basic interpreter is no longer available and the Apple will hang.

Issued 12 May 82

Microchess 2.0

Page 5600.000.01

INDEX TO MICROCHESS 2.0

ALPHABETIC LISTING

Doc	Title
001	Microchess and DOS 3.3

NUMERIC LISTING

Doc	Issue Date	Previous Date	Pages
000	MAY 12 82	SEP 28 81	1
001	SEP 18 81	-	1

Issued 18 Sep 81

Microchess 2.0

Page 5600.001.01

MICROCHESS AND DOS 3.3

MICROCHESS is a 13 sector, copy protected diskette. This means that you will need to use the BASICS diskette or the BOOT13 program from the DOS 3.3 master diskette to play MICROCHESS.

 Issued 1 Jun 82

Pascal Animation Tools

Page 6000.000.01

INDEX TO PASCAL ANIMATION TOOLS

ALPHABETIC LISTING

Doc	Title
001	Pascal Animation Tools and the Apple ///
003	Saving Character Sets

NUMERIC LISTING

Doc	Issue Date	Previous Date	Pages
000	MAY 12 82	OCT 1 81	1
001	OCT 1 81	-	1
003	MAY 21 82	-	1

Issued 1 Oct 81

Pascal Animation Tools

Page 6000.001.01

PASCAL ANIMATION TOOLS AND THE APPLE ///

The Pascal Animation Tools package won't work on the Apple /// in emulation mode because it is written in Pascal and there is no Language Card in the Apple ///.

Issued 21 May 82

Pascal Animation Tools

Page 6000.003.01

SAVING CHARACTER SETS

The most common problem when saving a character set is that the user is saving to the default volume. The default volume is the PASCAL ANIMATION master diskette which is write protected. Be sure to specify another diskette.

The second most common problem is having trouble loading a character set from disk. Usually this is caused by trying to execute the previously defined character set instead of editing it.

INDEX TO PASCAL

ALPHABETIC LISTING

Doc	Title
038	Bibliography
024	Booleans
030	The CALC Utility
022	Chaining in Pascal
011	Clearing the Video Screen
032	Converting Strings to Numeric Variables
008	Debugger
004	Difference Between Repeat..Until and While..Do
991	Errata - Apple Pascal Language Reference Manual
990	Errata - Apple Pascal Operating System Reference Manual
020	External References
015	External Terminal Setup
007	Extra Linefeeds on Printer
018	Immediate Mode Execution
013	Interface Cards in Peripheral Slot #3
033	Intrinsic Unit for Silentype Procedures
026	The Keypress Function
037	Library Units
021	Making a Copy of the "BASICS" Diskette
017	Memory Space Available
016	Nesting Pascal Procedures and Functions
034	Operand Formats
010	Pascal Exponents
025	Pascal Long Integer Compares - Version 1.0 Only
035	Pascal Run-Time Errors
019	Pascal Turnkey System
014	Passing Strings to and from Functions
031	Proposed ISO Standards for Pascal
036	Real Number Format
012	Rebooting Problems in Pascal
001	Resident and Noload Compiler Options with Intrinsic Units
039	Resident Compiler Option
009	Saving a Hi-Res Picture to a Disk File
027	The Scan Function
003	Sending Output to the Printer
005	Speeding-up Pascal Text File Reading
006	System Configuration
023	Terminal Driver Package for Pascal
028	TRUNC and ROUND Functions
029	TTL0UT
002	Using a Single Disk System

(Continued)

APPLE COMPUTER TECHNICAL NOTES

Page 6100.000.02

Pascal

Issued 2 Jun 82

NUMERIC LISTING

Doc	Issue Date	Previous Date	Pages
001	OCT 22 81	-	2
002	OCT 22 81	-	1
003	OCT 22 81	-	1
004	OCT 22 81	-	1
005	OCT 22 81	-	6
006	APR 28 82	OCT 22 81	1
007	OCT 22 81	-	1
008	OCT 22 81	-	1
009	APR 28 82	OCT 22 81	2
010	OCT 22 81	-	1
011	OCT 22 81	-	1
012	OCT 22 81	-	1
013	OCT 22 81	-	1
014	SEP 23 81	-	1
015	SEP 23 81	-	17
016	SEP 23 81	-	1
017	SEP 23 81	-	1
018	SEP 23 81	-	1
019	SEP 23 81	-	1
020	OCT 29 81	-	1
021	OCT 29 81	-	1
022	OCT 29 81	-	1
023	OCT 29 81	-	1
024	OCT 29 81	-	1
025	MAR 5 82	-	2
026	OCT 29 81	-	3
027	OCT 29 81	-	1
028	OCT 29 81	-	1
029	MAR 5 82	-	1
030	OCT 29 81	-	1
031	OCT 29 81	-	1
032	MAR 5 82	-	5
033	OCT 29 81	-	8
034	MAY 21 82	-	7
035	OCT 29 81	-	1
036	OCT 29 81	-	1
037	MAY 21 82	-	5
038	FEB 19 82	-	1
039	MAY 21 82	-	1
990	MAY 25 82	OCT 22 81	1
991	OCT 22 81	-	1

RESIDENT AND NOLOAD COMPILER OPTIONS WITH INTRINSIC UNITS

Intrinsic units can be treated as segments (overlays) in Apple Pascal version 1.1 by using the noload (*\$N+*) and the resident (*\$R unit name*) compiler options. Units specified by these options are read in from the disk only when a reference is made to code contained within the unit. When the calling procedure is exited, and all active references to the unit have been resolved, the unit is "swapped out", and the memory range can be used for other code segments. For more details on the use of these options, refer to pages 66-67 of the Pascal Language Reference Manual, and the Language Manual addendum.

When using the noload option, the unit code will remain in memory until the calling procedure is exited. This may result in a stack overflow (out of memory) if the unit is called repeatedly from inside the same procedure, without exiting. This is a known problem, and can be avoided by specifying the unit as "Resident" within that procedure or segment. Calling the unit from within a different procedure will also remedy this. The following examples illustrate this situation:

EXAMPLE A: Will generate an out of memory STACK OVERFLOW error by repeatedly loading APPLESTUFF.

```
PROGRAM EXAMPLEA;

USES APPLESTUFF;

VAR I: INTEGER;

BEGIN (* MAIN PROGRAM *)
  (*$N+*)
  WRITE (NUMBER OF WORDS AVAILABLE AT START OF PROGRAM : ^);
  WRITELN (MEMAVAIL);
  FOR I:=1 TO 100 DO BEGIN
    WRITELN (I, ^ ^, MEMAVAIL, ^ WORDS ^);
    NOTE (25,1) (* APPLESTUFF MUSIC ROUTINE *)
  END;
  WRITELN (PROGRAM COMPLETED SUCCESSFULLY ^)
END. (* MAIN PROGRAM *)
```

(Continued)

EXAMPLE B: Uses the RESIDENT OPTION to prevent APPLESTUFF from being reloaded.

```
PROGRAM EXAMPLEB;

USES APPLESTUFF;

VAR I: INTEGER;

BEGIN (* MAIN PROGRAM *)
  (*$N+*)
  (*$R APPLESTUFF *)
  WRITE (^NUMBER OF WORDS AVAILABLE AT START OF PROGRAM : ^);
  WRITELN (MEMAVAIL);
  FOR I:=1 TO 100 DO BEGIN
    WRITELN (I, ^ ^, MEMAVAIL, ^ WORDS^);
    NOTE (25,1) (* APPLESTUFF MUSIC ROUTINE *)
  END;
  WRITELN (^PROGRAM COMPLETED SUCCESSFULLY^)
END. (* MAIN PROGRAM *)
```

EXAMPLE C: Uses individual procedure to call APPLESTUFF routine, which allows unit to be released and then reloaded.

```
PROGRAM EXAMPLEB;

USES APPLESTUFF;

VAR I: INTEGER;

PROCEDURE PLAY;
BEGIN
  WRITELN (I, ^ ^, MEMAVAIL, ^ WORDS^);
  NOTE (25,1) (* APPLESTUFF MUSIC ROUTINE *)
END;

BEGIN (* MAIN PROGRAM *)
  (*$N+*)
  WRITE (^NUMBER OF WORDS AVAILABLE AT START OF PROGRAM : ^);
  WRITELN (MEMAVAIL);
  FOR I:=1 TO 100 DO PLAY;
  WRITELN (^PROGRAM COMPLETED SUCCESSFULLY^)
END. (* MAIN PROGRAM *)
```

USING A SINGLE DISK SYSTEM

On a single drive Pascal system, each major function (the editor, filer, compiler, assembler, etc.) should have a "stand-alone" diskette, which should consist of everything necessary to allow the system to function without swapping diskettes. For example:

A stand-alone editor system could be configured as:

```
SYSTEM.PASCAL
SYSTEM.MISCINFO
SYSTEM.EDITOR
SYSTEM.FILER
<plus text files to be edited>
```

A stand-alone compiler system could be configured as:

```
SYSTEM.PASCAL
SYSTEM.MISCINFO
SYSTEM.LIBRARY
SYSTEM.COMPILER
SYSTEM.LINKER
SYSTEM.EDITOR (optional for debugging)
SYSTEM.SYNTAX (also optional)
<plus the file to be compiled>
```

SYSTEM.APPLE does not have to be placed on the disk since it is loaded only during the initial boot; however, by making it resident, each disk would be fully capable of a cold start (power-up or RESET).

SENDING OUTPUT TO THE PRINTER

Data can be directed to the printer and to other system devices from a Pascal program with the addition of a few statements. For example, let's print a string on the printer:

```
PROGRAM DEMO;  
  
VAR REPORT:TEXT;  
    ANSWER:STRING;  
  
BEGIN  
    WRITE(STRING TO PRINT : ^);  
    READLN(ANSWER);  
    REWRITE(REPORT, ^PRINTER: ^);  
    WRITELN(REPORT, ANSWER)  
END.
```

To change devices, simply change the ^PRINTER: ^ in the rewrite statement to the name of the output device. The program APPLE3:DISKIO is an example of writing to a disk file.

To transfer a text file to the printer, use the Filer "T" function. For example, to print the contents of GRAFDEMO.TEXT (found on APPLE3:), type "T", followed by "APPLE3:GRAFDEMO.TEXT,PRINTER:". (Don't include the quotation marks.) The directory of APPLE3: can be printed out by typing "E", followed by "APPLE3:,PRINTER:".

DIFFERENCE BETWEEN REPEAT..UNTIL AND WHILE..DO

Pascal supports two forms of conditional loop: REPEAT..UNTIL and WHILE..DO. A loop of the form

```
REPEAT
  .
  .
UNTIL <condition>
```

will be executed at least once, even though the condition is satisfied when the loop is initially entered.

The loop consisting of

```
WHILE <condition> DO
  .
  .
```

will be executed only if the condition is satisfied when the loop is entered. See pages 22 and 23 of the Pascal User Manual and Report.

SPEEDING-UP PASCAL TEXT FILE READING

The reading of Pascal text files is normally done with the statement using the READLN statement as in

```
    READLN (FILEID, STRINGVARIABLE);
```

This operation can be made much faster by using the routines contained in the following program. Three procedures do the work. Their operation is explained, line-by-line, below:

```

PROCEDURE FILLBUFFER;
(* Fills the working buffer with data from the .TEXT file.      *)

BEGIN
    EMPTY := BLOCKREAD (INFILE,BUFFER,2) = 0;
(* Reads 2 blocks of the file into BUFFER and leaves the      *)
(* variable EMPTY equal to zero if the end of file marker is  *)
(* not yet reached.                                           *)

    IF NOT EMPTY THEN BEGIN
(* If there is still unprocessed data in the buffer, do this: *)

        NOTNULLS := BUFSIZE +
            SCAN (- BUFSIZE, <> CHR(0), BUFFER [1023]);
(* The length of a Pascal .TEXT file should always be in mul- *)
(* tiples of 2 blocks. Since strings (lines) do not span      *)
(* blocks, each block is likely to contain nulls (ASCII ZERO) *)
(* at the end. This line returns the number of real char-    *)
(* acters in the file, and discards the null ones.           *)

        BUFINDEX := 0;
(* The working index into the buffer is reset to zero after re- *)
(* filling the buffer.                                         *)
    END;
END;

PROCEDURE OPENFILE (FNAME: STRING);
(* Opens the file using the name passed by calling procedure. *)

BEGIN
    IF ((POS('text',FNAME) = 0) AND
        (POS('TEXT',FNAME) = 0)) THEN
        FNAME := CONCAT (FNAME,'TEXT');
(* Adds the .TEXT suffix if it's not already there.          *)

    RESET (INFILE, FNAME);
(* Actually opens the referenced file.                        *)

```

(Continued)

```

    FILLBUFFER; FILLBUFFER;
(* The first call to FILLBUFFER skips over the 2 blocks of *)
(* header information on .TEXT files. The second call actually *)
(* fills the buffer with information which will be used. *)
END;

```

```

PROCEDURE READFILE (VAR LINE: STRING);
(* Reads from the file and returns a string at a time in the *)
(* variable LINE. A word about the Pascal .TEXT file format: *)
(* Lines are stored as ASCII characters terminated with car- *)
(* riage returns. If a line contains any leading spaces, and *)
(* most Pascal source files contain some, these spaces are *)
(* "packed" into two bytes. The first byte is an ASCII DLE *)
(* (decimal 16) signifying that the line is packed. The *)
(* second byte is a count of spaces to be expanded. The *)
(* Editor unpacks these lines automatically, as does a READLN *)
(* from a file. We do that operation ourselves in this *)
(* procedure. The increase in speed is because we are using *)
(* highly specialized routine whereas the READLN intrinsic *)
(* is very general in nature; accepting strings, integers *)
(* and reals from the keyboard as well as from files. Note *)
(* that this format is optimized for Pascal source files and *)
(* it wastes two bytes for each and every line that does not *)
(* contain leading spaces. *)

```

```

VAR INDENT, LINELEN: INTEGER;
(* INDENT is the number of space characters to add. LINELEN *)
(* is the length of the new string to be formed. *)

```

```

BEGIN
    IF BUFINDEX >= NOTNULLS THEN FILLBUFFER;
(* If the buffer needs refilling, go and get another buffer. *)

    IF NOT EMPTY THEN BEGIN
(* If the file is not yet empty then do the following: *)

        LINELEN := SCAN (BUFSIZE, = CHR(13), BUFFER [BUFINDEX]);
(* Set LINELEN to the number of characters from the current *)
(* buffer pointer position (BUFINDEX) to the next carriage *)
(* return in the buffer. *)

        IF BUFFER [BUFINDEX] = CHR (16) THEN BEGIN
(* If the character at the buffer index is an ASCII DLE then *)
(* we have to unpack the leading spaces. *)

            INDENT := ORD (BUFFER [BUFINDEX + 1]) - 32;
(* Set INDENT to the number found at BUFINDEX + 1. (The *)
(* of space characters to insert). *)

```

(Continued)

```

        (*$R-*)
        LINE [0] := CHR (LINELEN + INDENT - 2);
        (*$R+*)
    (* Turn off Range Checking so we can manually change the      *)
    (* string length. Set the length of LINE to the number we had *)
    (* already gotten plus the number of spaces to unpack, throw- *)
    (* ing away two bytes for the DLE and count bytes. Turn Range *)
    (* Checking back on.                                          *)

        IF INDENT > 0 THEN FILLCHAR (LINE [1], INDENT, ' ');
    (* If there are spaces then fill in the appropriate number of *)
    (* them, starting with the first position in the new string.  *)

        IF LINELEN > 2 THEN MOVELEFT (BUFFER [BUFINDEX + 2],
                                     LINE [1 + INDENT], LINELEN - 2);
    (* If the string is more than 2 characters long then move the *)
    (* rest of it from the buffer into the string starting just  *)
    (* after the leading spaces previously inserted.              *)

        END ELSE BEGIN
    (* No DLE character was found. That means straight ASCII.    *)

        (*$R-*)
        LINE [0] := CHR (LINELEN);
        (*$R+*)
    (* Turn Range Checking off, set the length of the string to  *)
    (* LINELEN, and turn Range Checking back on.                  *)

        IF LINELEN > 0 THEN MOVELEFT (BUFFER [BUFINDEX],
                                     LINE [1], LINELEN);
    (* Move the characters from the buffer into LINE as above.    *)

        END;

        BUFINDEX := BUFINDEX + LINELEN + 1;
    (* Sets the pointer to the first character of the next string *)
    (* in the buffer for the next time through.                  *)

        END;
    END;
    
```

Here's a program that demonstrates the difference in speed between the two methods of reading strings:

```

PROGRAM QUICKREAD;          (* Very fast line read routine *)

CONST BUFSIZE              = 1024;
      BUFLN                = 1023;
      FILENAME             = 'QWERTY9.TEXT';
                                (* Probably not on user disk *)
    
```

(Continued)

```

VAR   LINE:           STRING;
      INFILE:         FILE;
      TEXTFILE:       TEXT;
      CH, OPTION:     CHAR;
      EMPTY, HELL_FREEZES_OVER: BOOLEAN;
      ERROR:          INTEGER;
      NOTNULLS:       0..BUFSIZE;
                          (* # of non-null chars *)
      BUFINDEX:       0..BUFSIZE;
                          (* Index within buffer *)
      BUFFER:         PACKED ARRAY [0..BUFLN] of CHAR;

```

```

PROCEDURE FILLBUFFER;
BEGIN
  EMPTY := BLOCKREAD (INFILE,BUFFER,2) = 0;
  IF NOT EMPTY THEN BEGIN
    NOTNULLS := BUFSIZE + SCAN (- BUFSIZE, <> CHR(0),
                                BUFFER [1023]);
    BUFINDEX := 0;
  END;
END;

PROCEDURE OPENFILE (FNAME: STRING);
BEGIN
  IF ((POS('text',FNAME) = 0) AND
      (POS('TEXT',FNAME) = 0)) THEN
    FNAME := CONCAT (FNAME,'TEXT');
  RESET (INFILE, FNAME);
  FILLBUFFER; FILLBUFFER;
END;

PROCEDURE READFILE (VAR LINE: STRING);
VAR INDENT,
    LINELEN:   INTEGER;
BEGIN
  IF BUFINDEX >= NOTNULLS THEN FILLBUFFER;
  IF NOT EMPTY THEN BEGIN
    LINELEN := SCAN (BUFSIZE, = CHR(13), BUFFER [BUFINDEX]);
    IF BUFFER [BUFINDEX] = CHR (16) THEN BEGIN
      INDENT := ORD (BUFFER [BUFINDEX + 1]) - 32;
      (*$R-*)
      LINE [0] := CHR (LINELEN + INDENT - 2);
      (*$R+*)
      IF INDENT > 0 THEN FILLCHAR (LINE [1], INDENT, ' ');
      IF LINELEN > 2 THEN MOVELEFT (BUFFER [BUFINDEX + 2],
                                   LINE [1 + INDENT], LINELEN - 2);
    END ELSE BEGIN
      (*$R-*)
      LINE [0] := CHR (LINELEN);
      (*$R+*)
    END;
  END;

```

(Continued)

```

        IF LINELEN > 0 THEN MOVELEFT (BUFFER [BUFINDEX],
                                        LINE [1], LINELEN);
    END;
    BUFINDEX := BUFINDEX + LINELEN + 1;
END;
END;

PROCEDURE GETOPTION;
BEGIN
    REPEAT
        WRITELN;
        WRITE (^OPTIONS: Q)uickread, R)eadln, E)xit ^);
        CH := ^ ^;
        READ (KEYBOARD, CH);
    UNTIL CH IN [^Q^, ^q^, ^R^, ^r^, ^E^, ^e^];
    IF CH IN [^E^, ^e^] THEN BEGIN
        PAGE (OUTPUT);
        RESET (INFILE, FILENAME);
        CLOSE (INFILE, PURGE);
        EXIT (PROGRAM);
    END;
END;

PROCEDURE PRINTFILE;
BEGIN
    PAGE (OUTPUT); WRITELN;
    CASE CH OF
        ^Q^, ^q^: BEGIN
            OPENFILE (FILENAME);
            REPEAT
                LINE := ^ ^;
                READFILE (LINE);
                WRITELN (LINE);
            UNTIL EMPTY;
            CLOSE (INFILE);
        END;
        ^R^, ^r^: BEGIN
            RESET (TEXTFILE, FILENAME);
            REPEAT
                READLN (TEXTFILE, LINE);
                WRITELN (LINE);
            UNTIL EOF (TEXTFILE);
            CLOSE (TEXTFILE);
        END;
    END; (* CASE *)
END;

```

(Continued)

```

PROCEDURE INIT;
VAR TEMP:      STRING;
BEGIN
  PAGE (OUTPUT); WRITELN;
  WRITE (^One moment, please...^);
  REWRITE (TEXTFILE, FILENAME);
  WRITE (TEXTFILE, ^This is a test file written to ^);
  WRITE (TEXTFILE, ^demonstrate the ^);
  WRITE (^.);
  WRITELN (TEXTFILE, ^"QUICKREAD" program.^);
  WRITE (^.);
  WRITELN (TEXTFILE);
  TEMP := ^I'm melting.^;
  REPEAT
    WRITE (TEXTFILE, ^TEST STRING --> ^);
    WRITELN (TEXTFILE, TEMP);
    DELETE (TEMP, LENGTH (TEMP), 1);
    WRITE (^.);
  UNTIL TEMP = ^;
  WRITELN (TEXTFILE, ^          AHHHHHhhhhhh....^);
  WRITELN (TEXTFILE);
  WRITE (TEXTFILE, ^Notice that using READLN takes much ^);
  WRITE (TEXTFILE, ^more time^);
  WRITELN (TEXTFILE, ^ to read a long line^);
  WRITE (^.);
  WRITELN (TEXTFILE, ^than it takes to read a short one.^);
  WRITE (^.);
  CLOSE (TEXTFILE, LOCK);
  WRITELN;
END;

BEGIN
  INIT; (* Write a temporary file for *)
  HELL_FREEZES_OVER := FALSE; (* A possibly valid assumption *)
  REPEAT
    GETOPTIONS; (* Get users choice *)
    PRINTFILE; (* Print the file *)
  UNTIL HELL_FREEZES_OVER; (* FOREVER!!! *)
END.

```

Issued 28 Apr 82

Pascal

Page 6100.006.01

SYSTEM CONFIGURATION

A 48K Apple II or Apple II Plus with at least one disk drive and a 16K memory expansion board like the Apple Language Card is required to run the Apple Pascal system. Other Apple interface cards may be added for additional system functions such as printer, external terminal, modem, or more disk drives. The Pascal system has specific slot assignments for the various peripheral devices. Slot assignments are listed on page 277 of the Pascal Operating System manual.

The Apple Pascal system comes configured to work with Apple interface cards. To use other types of interface cards, it may be necessary to modify the system or add additional driver routines. Pascal version 1.1 has provision for attaching additional peripheral drivers through the use of the ATTACH utility. This utility program, along with extensive documentation, is being supplied by the International Apple Core to all member user groups. Any and all such modifications are at the user's own risk.

Issued 22 Oct 81

Pascal

Page 6100.007.01

EXTRA LINEFEEDS ON PRINTER

The double spacing effect on some printers when used with Pascal can be remedied by e(X)ecuting APPLE3:LINEFEED. This program can be transferred to the diskette you boot (usually APPLE1: or APPLE0:) as SYSTEM.STARTUP and it will execute automatically when the system is booted. The text of this routine is also on the volume APPLE3: for inclusion in your own programs.

Issued 22 Oct 81

Pascal

Page 6100.008.01

DEBUGGER

The "Debug" feature in Pascal is not being supported by UCSD or Softech, Inc. and has not been implemented in Apple Pascal, version 1.1.

SAVING A HI-RES PICTURE TO A DISK FILE

Pascal graphics does not provide built-in routines to load or save the Hi-Res graphics screen to a disk file. The following program illustrates a simple method for saving and loading any Hi-Res screen created in a Pascal environment.

```

PROGRAM DEMOPIC;
(*
(* PROGRAM LOADS AND SAVES HI-RES SCREEN TO DISK
(* 12/79 BASED ON "SLIDE SHOW" BY BILL ATKINSON
(*
*)
USES TURTLEGRAPHICS, APPLESTUFF;
CONST HIRESPI = 8192;
VAR CHEAT: RECORD CASE BOOLEAN OF
    TRUE: (INTPART: INTEGER);
    FALSE: (PTRPART: ^INTEGER);
END;

PROCEDURE DRAWPICS;
(*
(* DRAW SOME STUFF ON THE HI-RES SCREEN
(* THIS CAN BE REPLACED WITH ANY GRAPHICS ROUTINE
(*
*)
BEGIN
    MOVETO (0,0); PENCOLOR (WHITE);
    MOVETO (279,0); MOVETO (279,191);
    MOVETO (0,191); MOVETO (0,0);
    PENCOLOR (NONE);
    MOVETO (75,95); WSTRING (^ THIS IS A TEST ^);
    MOVETO (28,5); WSTRING (^< PRESS RETURN TO EXIT PROGRAM >^);
END;

PROCEDURE ERROR;
(*
(* IF ERROR, PROGRAM WILL TERMINATE FROM HERE
(*
*)
BEGIN
    TEXTMODE;
    WRITELN (^ERROR ENCOUNTERED - PROGRAM TERMINATED^);
    EXIT (PROGRAM)
END;
    
```

(Continued)

```

PROCEDURE BLOAD (FILENAME: STRING);
(*                                     *)
(*  BLOCKREADS THE HI-RES BUFFER INTO MEMORY  *)
(*                                     *)
VAR I,IO: INTEGER;
    F: FILE;
BEGIN
    CHEAT.INTPART:=HIRESPl;
    RESET(F,FILENAME);      (* OPEN FILE FOR INPUT *)
    (*$I-*)
    IO:=BLOCKREAD(F,CHEAT.PTRPART^,16);
    I:=IORESULT;
    (*$I+*)
    CLOSE(F);
    IF (I <> 0) OR (IO <> 16) THEN ERROR;
END;

PROCEDURE BSAVE (FILENAME:STRING);
(*                                     *)
(*  SAVES HI-RES PICTURE TO DISK VIA BLOCKWRITE  *)
(*                                     *)
VAR I,IO: INTEGER;
    F: FILE;
BEGIN
    CHEAT.INTPART:=HIRESPl;
    REWRITE (F,FILENAME);   (* OPEN NEW DISK FILE FOR OUTPUT *)
    (*$I-*)
    IO:=BLOCKWRITE (F,CHEAT.PTRPART^,16);
    I:=IORESULT;
    (*$I+*)
    CLOSE (F,LOCK)
    IF (I <> 0) OR (IO <> 16) THEN ERROR;
END;

BEGIN  (* MAIN PROGRAM *)
    INITTURTLE;
    DRAWPICS;
    BSAVE (^:DEMO.PIC^);
    FILLSCREEN (BLACK);      (* CLEARS HI-RES SCREEN *)
    BLOAD (^:DEMO.PIC^);
    REPEAT UNTIL KEYPRESS;  (* PAUSE *)
    TEXTMODE
END.

```

PASCAL EXPONENTS

The calculation 10^X is not included in the UCSD Pascal definition. The system intrinsic PWROFTEN ("power of ten") returns 10^X , provided X is an integer in the range 0..37. Please refer to page 45 in the Pascal Language Reference manual.

The function EXP (in the library unit TRANSCEN) is of the form e^X , where X is a real number. The relationship between 10^X and e^X is:

$$10^X = e^{(X \text{ LN } 10)} \quad (\text{LN} = \text{natural log})$$

Here is a simple program which illustrates the use of Pascal exponents:

```
PROGRAM EXPONENT;  
  
USES TRANSCEND;  
  
BEGIN  
  WRITELN ('10 ^ 3 = ',PWROFTEN(3));  
  WRITELN ('e ^ 3 = ',EXP(3));  
  WRITELN ('10 ^ 3 by the conversion = ',EXP(3*LN(10)));  
END.
```

Issued 22 Oct 81

Pascal

Page 6100.011.01

CLEARING THE VIDEO SCREEN

The routine CLEARSCREEN as described in Problem Solving Using Pascal, by Kenneth Bowles, is a system function of the UCSD computer facility. On the standard Apple screen, a similar function can be performed by the system intrinsic PAGE(OUTPUT), which issues a form feed.

The following procedure can be used to clear the Apple's screen.

```
PROCEDURE CLEARSCREEN;  
BEGIN  
    WRITE (CHR(12));  
END;
```

CHR(12) is the character used to clear the standard Apple video screen, as shown in the SETUP program. This procedure may be modified for use with other external terminals by replacing CHR(12) with the appropriate control character (or characters) for "Erase Screen".

REBOOTING PROBLEMS IN PASCAL

If recurrent rebooting is a problem in your Pascal system, especially when typing into the editor, it may be due to hitting more than one key at once. Try slowing down to allow the first key to be released before typing the next one. When typing quickly, many people do not completely release one key before striking the next. If you try to press more than one key at a time, the keyboard encoder chip sees the logical AND of the keys currently being held down. The AND operation results in bits being turned off, so it is quite likely that many combinations of keys will result in an ASCII 0 (NUL), which is a BREAK in Pascal.

The reason this doesn't happen in Basic is that a NUL (ASCII 0) entered from the keyboard has no function in Basic or DOS. If slowing down doesn't help the problem, it may be due to memory errors, and it would be wise to have the hardware checked out.

You may also use the SETUP program (found on APPLE3:) to change the "Key for BREAK" parameter in SYSTEM.MISCINFO to an ASCII 127 (DLE). This character cannot be generated from the Apple keyboard, thereby effectively disabling the BREAK function. The RESET key will work normally.

Issued 22 Oct 81

Pascal

Page 6100.013.01

INTERFACE CARDS IN PERIPHERAL SLOT #3

If a Communications card, Serial card, or DC Hayes Micromodem card is placed in slot #3, the Pascal system will not appear to boot correctly on the standard Apple video. This is due to the fact that Pascal has recognized a card in slot #3, which is reserved for an external terminal, and is attempting to communicate with the terminal instead of the default CONSOLE and SYSTEM (video and keyboard). Other cards may also cause this effect, which can be easily remedied by removing the card from slot #3.

PASSING STRINGS TO AND FROM FUNCTIONS

Question:

Since functions cannot return a non-scalar variable, how does a function return a string?

Answer:

Strings can be passed to and from functions or assembly language routines by using variable parameter. A VARIABLE (VAR) parameter is a pass-by-reference parameter. That is, the address of the parameter is passed. In contrast, when a value parameter is used, the value of the parameter is passed (See pp.87-90, Apple /// Pascal Programmer's Manual, Volume 1). An example of passing a string to and from a function is listed below. Since parameter-passing rules are the same for procedures as for functions, a procedure could have been used similarly. The value returned by the function is used to determine the success of the function. This is a good programming convention to use when writing functions.

```
PROGRAM PASS;

VAR I: INTEGER;
    ST1: STRING;
    TABLE: ARRAY [0..50] OF STRING; (* For this program to run, *)
                                      (* TABLE must be initialized *)
                                      (* prior to calling DOIT. *)

FUNCTION DOIT (VAR ST2: STRING; INDEX: INTEGER):BOOLEAN;
    (* VAR designates ST2 as a VARIABLE parameter. *)
BEGIN
    DOIT := TRUE;
    IF LENGTH(TABLE [INDEX]) = 0 THEN (* DOIT returns false if *)
        DOIT := FALSE                (* the given entry(INDEX) *)
    ELSE ST2 := TABLE [INDEX]       (* in the table is null. *)
    END;                               (* Else, it returns true. *)

BEGIN
    FOR I := 0 TO 50 DO
        BEGIN (* ST1 should be null unless *)
            ST1:= ''; (* DOIT alters it. *)
            IF DOIT(ST1,I) THEN (* If the array entry is not *)
                Writeln(ST1) (* null, print the string. *)
            END (* FOR *)
        END
    END.
```

NOTE #1:

"...an individual element of a packed variable cannot be supplied as the actual parameter" (pg.90, Apple /// Pascal Programmer's Manual, Volume 1).

(Continued)

NOTE #2:

The addresses of strings are ALWAYS passed to the procedure or function, whether the string is a VARIABLE parameter or a value parameter. Declaring the string parameter to be a VARIABLE parameter avoids copying the entire value of the string to the function, with the attendant memory and performance penalty. The programmer should bear in mind that, in general, it is good programming technique to use value parameters as much as possible, since this provides for maximum 'decoupling' of function and calling program.

EXTERNAL TERMINAL SETUP

Apple Pascal supports a variety of external terminals which can be used to display 80 character lines and upper/lower case. Pages 199-213 of the Apple Pascal Operating System reference manual describe the procedure for reconfiguring the Pascal system to communicate with the external terminal. You must bind the appropriate GOTO routine into SYSTEM.PASCAL using the program APPLE3:BINDER, and use APPLE3:SETUP to tailor your SYSTEM.MISCINFO file for the specific terminal parameters.

The terminal may be connected to the Pascal system using the High-Speed Serial Card in place of the Communications Card. If the serial card is used, the eighth data bit (most significant bit) switch in the terminal MUST be set to 0. Refer to the operating manual or contact the manufacturer for the location of this switch. With the serial card, all type-ahead is eliminated.

USING THE SETUP PROGRAM

The program SETUP is located on the diskette APPLE3. When executed, it will read in the current SYSTEM.MISCINFO. You may change a single parameter, or edit the entire file. To edit all the parameters, type "C" (Change), and then "P" (Prompted). If you need instructions, type "H" (Help), or "T" (Teach). These instructions are detailed on pages 199-213 of the Pascal Operating System Reference Manual.

To leave editing mode prior to the end of the file, type "!" (Abort). Then type "Q", "Q" (to get to the exit/update level). To leave without updating, simply type "E". If you wish to test your setup, but not save the file on disk, type "M", then "E". To make a permanent record of your changes, type "D", then "E". A file named NEW.MISCINFO will be placed on your boot disk.

If you wish to edit a single parameter, type "C", then "S" (Single). Type the exact name of the field you wish to edit, exactly as it appears below. These parameters are explained in the following sections, arranged by their general function.

SYSTEM.MISCINFO SCREEN CONTROL FUNCTIONS

Backspace

Sent to screen to move cursor left one space without altering the text.
Normal setting for Apple II: CTRL-H (BS).

Erase Line

Sent to screen to erase entire line where cursor is positioned. Not used in the Apple II and should be set to NUL (ASCII 0).

(Continued)

Erase Screen

Erases screen and places cursor at "home" position, the upper left corner of the screen. Normal setting: CTRL-L (FF).

Erase to End of Line

Sent to screen to clear from cursor position to end of line. Normal setting: CTRL-] (GS).

Erase to End of Screen

Clears screen from cursor position to end of screen. Normal setting: CTRL-K (VT).

Has Random Cursor Addressing

TRUE for video terminals only (TRUE for Apple video). Enables use of "GOTOXY" sequence.

Lead-in to Screen

Prefix character for dual-character screen control functions, if any. Set to NUL (ASCII 0) if no lead-in is being used.

Move Cursor Home

Sent to screen to home cursor to "home" position in upper left corner. Normal setting: CTRL-Y (EM).

Move Cursor Right

Moves cursor right one space. Normal setting: CTRL-\ (FS).

Move Cursor Up

Moves cursor up one line. Normal setting: CTRL-_ (US).

Screen Height

Number of lines displayed on terminal. Set to 0 for hard-copy terminal or where paging is not appropriate. Setting is 24 (decimal) for Apple II.

Screen Width

Number of characters displayed on each line. Usually set to 80 for most terminals, 79 will allow nearly all of the Pascal window to be displayed while generating shortened prompt lines and messages which are better suited for a 40-character line.

SYSTEM.MISCINFO KEYBOARD FUNCTIONS**Has Lower Case**

FALSE for Apple II keyboard. Set to TRUE if external terminal can generate lower case from its keyboard.

Key for Break

Character typed to send BREAK to system. Not used in Apple II and should be set to NUL (ASCII 0).

(Continued)

Key for Flush

Character typed to cancel console output. Processing continues without output until FLUSH character typed again, or until file KEYBOARD is accessed. Normal setting: CTRL-F (ACK).

Key for Stop

Stops output processing. This is useful when information is streaming out to the console faster than you can read. Output resumes when STOP is toggled again. Normal setting: CTRL-S (DC3).

Key to Delete Character

Causes one character to be removed from current line until nothing is left on line. Normal setting: CTRL-H (BS).

Key to Delete Line

Cancels current line of input. Normal setting: CTRL-X (CAN).

Key to End File

Console EOF character. When reading from KEYBOARD or INPUT, this character will set EOF=TRUE. Normal setting: CTRL-C (ETX).

Lead-in from Keyboard

Prefix character for dual-character sequences (if any) generated on keyboard. Set to NUL (ASCII 0) if no lead-in is being used.

Key to Move Cursor Down

Character typed to move cursor down one line. On many terminals, this corresponds to the down-arrow. Apple keyboard setting: CTRL-L (FF).

Key to Move Cursor Left

Character typed to move cursor left one space. On many terminals this corresponds to the left-arrow. Apple keyboard setting: CTRL-H (BS).

Key to Move Cursor Right

Character typed to move cursor right one space. On many terminals, this corresponds to the right-arrow. Apple keyboard setting: CTRL-U (NAK).

Key to Move Cursor Up

Character typed to move cursor up one line. On many terminals, this corresponds to the up-arrow. Apple keyboard setting: CTRL-O (SI).

SYSTEM.MISCINFO EDITOR FUNCTIONS**Editor "Accept" Key**

Character typed to confirm last action, e. g., accept text inserted, deleted, exchanged, etc. Normal setting: CTRL-C (ETX).

(Continued)

Editor "Escape" Key

Character typed to instruct editor to ignore (escape) all actions performed since last "accept". Normal setting: ESCAPE (ESC).

Non-printing Character

Symbol displayed on screen to indicate presence of non-printable character (such as control characters). Normal setting: "?".

SYSTEM.MISCINFO HARDWARE FUNCTIONS**Has 8510A**

Always FALSE for Apple II.

Has Byte-Flipped Machine

Always FALSE for the Apple II. The Apple II reads data low-byte first. Byte-flipped machines read data high-byte first. (This parameter is not included in the SETUP program.)

Has Clock

Always FALSE for Apple II. No provision for an accessory real-time clock has been made in the operating system.

SYSTEM.MISCINFO OPERATING SYSTEM PARAMETERS**Has Slow Terminal**

Should be TRUE for terminal running at 600 baud or less. This will issue shortened prompts and messages. FALSE for Apple II.

Student

Instructs operating system to simplify operation for the novice. An example would be a default entry into the Editor if a compile error is encountered. May also disallow entry into certain portions of the system. Should be set to FALSE for Apple II.

Vertical Move Delay

Number of nulls (ASCII 0) to be sent after each carriage return, Erase to End of Line, or Erase to End of Screen to allow cursor to complete movement before additional characters are sent. Set to NUL (ASCII 0) for Apple II.

(Continued)

SYSTEM.MISCINFO PREFIXED PARAMETERS

The following parameters indicate whether or not the control function is to be prefixed by the lead-in character. Set to FALSE if no lead-in is required (as in the case of control characters), and TRUE if the lead-in is needed (ESC sequences, for example).

```
PREFIXED[Delete Character]
PREFIXED[Editor "Accept" Key]
PREFIXED[Editor 'Escape' Key] (This entry is different!)
PREFIXED[Erase Line]
PREFIXED[Erase Screen]
PREFIXED[Erase to End of Line]
PREFIXED[Erase to End of Screen]
PREFIXED[Key for Break]
PREFIXED[Key for Flush]
PREFIXED[Key to Move Cursor Down]
PREFIXED[Key to Move Cursor Left]
PREFIXED[Key to Move Cursor Right]
PREFIXED[Key to Move Cursor Up]
PREFIXED[Key for Stop]
PREFIXED[Key to Delete Character]
PREFIXED[Key to Delete Line]
PREFIXED[Key to End File]
PREFIXED[Move Cursor Home]
PREFIXED[Move Cursor Right]
PREFIXED[Move Cursor Up]
PREFIXED[Non-printing Character]
```

(Continued)

Apple Standard Video and Keyboard Setup Parameters

GOTO Code

The system is already configured for the Apple keyboard and video. No modifications are necessary. These are the "normal" settings suggested in the previous sections which describe the setup parameters.

Parameter Values

Backspace	CTRL-H (08)	
Editor "Accept" Key	CTRL-C (03)	
Editor "Escape" Key	ESC (27)	
Erase Line	NUL (00)	
Erase Screen	CTRL-L (12)	
Erase to End of Line	CTRL-] (29)	
Erase to End of Screen	CTRL-K (11)	
Has 8510A	FALSE	
Has Clock	FALSE	
Has Lower Case	FALSE	
Has Random Cursor Addr.	TRUE	
Has Slow Terminal	FALSE	
Key for Break	NUL (00)	
Key for Flush	CTRL-F (06)	
Key for Stop	CTRL-S (19)	
Key to Delete Character	CTRL-H (08)	
Key to Delete Line	CTRL-X (24)	
Key to End File	CTRL-C (03)	
Key to Move Cursor Down	CTRL-L (12)	* These may be
Key to Move Cursor Left	CTRL-H (08)	* altered by
Key to Move Cursor Right	CTRL-U (21)	* the user for
Key to Move Cursor Up	CTRL-O (15)	* convenience.
Lead-in from Keyboard	NUL (00)	
Lead-in to Screen	NUL (00)	
Move Cursor Home	CTRL-Y (25)	
Move Cursor Right	CTRL-\ (28)	
Move Cursor Up	CTRL- (31)	
Non-printing Character	"?" (63)	

(Continued)

Apple Standard Video and Keyboard Setup Parameters

PREFIXED[Delete Character]	FALSE
PREFIXED[Editor "Accept" Key]	FALSE
PREFIXED[Editor "Escape" Key]	FALSE
PREFIXED[Erase Line]	FALSE
PREFIXED[Erase Screen]	FALSE
PREFIXED[Erase to End of Line]	FALSE
PREFIXED[Erase to End of Screen]	FALSE
PREFIXED[Key for Break]	FALSE
PREFIXED[Key for Flush]	FALSE
PREFIXED[Key to Move Cursor Down]	FALSE
PREFIXED[Key to Move Cursor Left]	FALSE
PREFIXED[Key to Move Cursor Right]	FALSE
PREFIXED[Key to Move Cursor Up]	FALSE
PREFIXED[Key for Stop]	FALSE
PREFIXED[Key to Delete Character]	FALSE
PREFIXED[Key to Delete Line]	FALSE
PREFIXED[Key to End File]	FALSE
PREFIXED[Move Cursor Home]	FALSE
PREFIXED[Move Cursor Right]	FALSE
PREFIXED[Move Cursor Up]	FALSE
PREFIXED[Non-printing Character]	FALSE
Screen Height	24
Screen Width	79
Student	FALSE
Vertical Move Delay	0

(Continued)

Soroc IQ120 Terminal Setup Parameters

GOTO Code

Use APPLE3:SOROC.GOTO No modifications are necessary.

Parameter Values

Backspace	CTRL-H (08)	
Editor "Accept" Key	CTRL-C (03)	
Editor "Escape" Key	ESC (27)	
Erase Line	NUL (00)	
Erase Screen	"*" (42)	
Erase to End of Line	"T" (84)	
Erase to End of Screen	"Y" (89)	
Has 8510A	FALSE	
Has Clock	FALSE	
Has Lower Case	TRUE	
Has Random Cursor Addr.	TRUE	
Has Slow Terminal	FALSE	
Key for Break	NUL (00)	
Key for Flush	CTRL-F (06)	
Key for Stop	CTRL-S (19)	
Key to Delete Character	CTRL-H (08)	
Key to Delete Line	DEL (127)	
Key to End File	CTRL-C (03)	
Key to Move Cursor Down	CTRL-J (10)	* These may be
Key to Move Cursor Left	CTRL-H (08)	* altered by
Key to Move Cursor Right	CTRL-L (12)	* the user for
Key to Move Cursor Up	CTRL-K (11)	* convenience.
Lead-in from Keyboard	NUL (00)	
Lead-in to Screen	ESC (27)	
Move Cursor Home	CTRL-^ (30)	
Move Cursor Right	CTRL-L (12)	
Move Cursor Up	CTRL-K (11)	
Non-printing Character	"?" (63)	

(Continued)

Soroc IQ120 Terminal Setup Parameters

PREFIXED[Delete Character]	FALSE
PREFIXED[Editor "Accept" Key]	FALSE
PREFIXED[Editor 'Escape' Key]	FALSE
PREFIXED[Erase Line]	FALSE
PREFIXED[Erase Screen]	TRUE
PREFIXED[Erase to End of Line]	TRUE
PREFIXED[Erase to End of Screen]	TRUE
PREFIXED[Key for Break]	FALSE
PREFIXED[Key for Flush]	FALSE
PREFIXED[Key to Move Cursor Down]	FALSE
PREFIXED[Key to Move Cursor Left]	FALSE
PREFIXED[Key to Move Cursor Right]	FALSE
PREFIXED[Key to Move Cursor Up]	FALSE
PREFIXED[Key for Stop]	FALSE
PREFIXED[Key to Delete Character]	FALSE
PREFIXED[Key to Delete Line]	FALSE
PREFIXED[Key to End File]	FALSE
PREFIXED[Move Cursor Home]	FALSE
PREFIXED[Move Cursor Right]	FALSE
PREFIXED[Move Cursor Up]	FALSE
PREFIXED[Non-printing Character]	FALSE
Screen Height	24
Screen Width	80
Student	FALSE
Vertical Move Delay	0

(Continued)

Hazeltine 1500 Terminal Setup Parameters

GOTO Code

Use APPLE3:HAZEL.GOTO No modifications are necessary.

Parameter Values

Backspace	CTRL-H (08)	
Editor "Accept" Key	CTRL-C (03)	
Editor "Escape" Key	ESC (27)	
Erase Line	NUL (00)	
Erase Screen	CTRL-\ (28)	
Erase to End of Line	CTRL-O (15)	
Erase to End of Screen	CTRL-X (24)	
Has 8510A	FALSE	
Has Clock	FALSE	
Has Lower Case	TRUE	
Has Random Cursor Addr.	TRUE	
Has Slow Terminal	FALSE	
Key for Break	NUL (00)	
Key for Flush	CTRL-F (06)	
Key for Stop	CTRL-S (19)	
Key to Delete Character	CTRL-H (08)	
Key to Delete Line	DEL (127)	
Key to End File	CTRL-C (03)	
Key to Move Cursor Down	CTRL-K (11)	* These may be
Key to Move Cursor Left	CTRL-H (08)	* altered by
Key to Move Cursor Right	CTRL-P (16)	* the user for
Key to Move Cursor Up	CTRL-L (12)	* convenience.
Lead-in from Keyboard	NUL (00)	
Lead-in to Screen	"~" (126)	
Move Cursor Home	CTRL-R (18)	
Move Cursor Right	CTRL-P (16)	
Move Cursor Up	CTRL-L (12)	
Non-printing Character	"?" (63)	

(Continued)

Hazeltine 1500 Terminal Setup Parameters

PREFIXED[Delete Character]	FALSE
PREFIXED[Editor "Accept" Key]	FALSE
PREFIXED[Editor "Escape" Key]	FALSE
PREFIXED[Erase Line]	FALSE
PREFIXED[Erase Screen]	TRUE
PREFIXED[Erase to End of Line]	TRUE
PREFIXED[Erase to End of Screen]	TRUE
PREFIXED[Key for Break]	FALSE
PREFIXED[Key for Flush]	FALSE
PREFIXED[Key to Move Cursor Down]	FALSE
PREFIXED[Key to Move Cursor Left]	FALSE
PREFIXED[Key to Move Cursor Right]	FALSE
PREFIXED[Key to Move Cursor Up]	FALSE
PREFIXED[Key for Stop]	FALSE
PREFIXED[Key to Delete Character]	FALSE
PREFIXED[Key to Delete Line]	FALSE
PREFIXED[Key to End File]	FALSE
PREFIXED[Move Cursor Home]	TRUE
PREFIXED[Move Cursor Right]	FALSE
PREFIXED[Move Cursor Up]	TRUE
PREFIXED[Non-printing Character]	FALSE
Screen Height	24
Screen Width	80
Student	FALSE
Vertical Move Delay	0

ADDITIONAL NOTES

The four parity switches must be set to OFF. The CR/Auto LF switch must be set to CR. These DIP switches are located beneath the front panel plate of the terminal.

(Continued)

Hazeltine 1510 Terminal Setup Parameters

GOTO Code

Use APPLE3:HAZEL.GOTO No modifications are necessary.

Parameter Values

Backspace	CTRL-H (08)	
Editor "Accept" Key	CTRL-C (03)	
Editor "Escape" Key	ESC (27)	
Erase Line	NUL (00)	
Erase Screen	CTRL-\ (28)	
Erase to End of Line	CTRL-O (15)	
Erase to End of Screen	CTRL-X (24)	
Has 8510A	FALSE	
Has Clock	FALSE	
Has Lower Case	TRUE	
Has Random Cursor Addr.	TRUE	
Has Slow Terminal	FALSE	
Key for Break	NUL (00)	
Key for Flush	CTRL-F (06)	
Key for Stop	CTRL-S (19)	
Key to Delete Character	CTRL-H (08)	
Key to Delete Line	DEL (127)	
Key to End File	CTRL-C (03)	
Key to Move Cursor Down	CTRL-K (11)	* These may be
Key to Move Cursor Left	CTRL-H (08)	* altered by
Key to Move Cursor Right	CTRL-P (16)	* the user for
Key to Move Cursor Up	CTRL-L (12)	* convenience.
Lead-in from Keyboard	"~" (126)	
Lead-in to Screen	"~" (126)	
Move Cursor Home	CTRL-R (18)	
Move Cursor Right	CTRL-P (16)	
Move Cursor Up	CTRL-L (12)	
Non-printing Character	"?" (63)	

(Continued)

Hazeltine 1510 Terminal Setup Parameters

PREFIXED[Delete Character]	FALSE
PREFIXED[Editor "Accept" Key]	FALSE
PREFIXED[Editor 'Escape' Key]	FALSE
PREFIXED[Erase Line]	FALSE
PREFIXED[Erase Screen]	TRUE
PREFIXED[Erase to End of Line]	TRUE
PREFIXED[Erase to End of Screen]	TRUE
PREFIXED[Key for Break]	FALSE
PREFIXED[Key for Flush]	FALSE
PREFIXED[Key to Move Cursor Down]	TRUE
PREFIXED[Key to Move Cursor Left]	FALSE
PREFIXED[Key to Move Cursor Right]	FALSE
PREFIXED[Key to Move Cursor Up]	TRUE
PREFIXED[Key for Stop]	FALSE
PREFIXED[Key to Delete Character]	FALSE
PREFIXED[Key to Delete Line]	FALSE
PREFIXED[Key to End File]	FALSE
PREFIXED[Move Cursor Home]	TRUE
PREFIXED[Move Cursor Right]	FALSE
PREFIXED[Move Cursor Up]	TRUE
PREFIXED[Non-printing Character]	FALSE
Screen Height	24
Screen Width	80
Student	FALSE
Vertical Move Delay	0

(Continued)

IBM 3101-22 Terminal Setup Parameters

GOTO Code

```
(*SU-*)
PROGRAM GOXY; (* For IBM-3010-22 *)
PROCEDURE FGOTOXY(X,Y:INTEGER);
VAR SEND: PACKED ARRAY [0..3] OF 0..255;
BEGIN
  IF X>79 THEN X:=79 ELSE IF X<0 THEN X:=0;
  IF Y>23 THEN Y:=23 ELSE IF Y<0 THEN Y:=0;
  SEND[0]:=27; (* Lead-in *) SEND[1]:=ORD('^Y');
  SEND[2]:=32+X; SEND[3]:=32+Y;
  UNITWRITE(2,SEND,4)
END;

BEGIN (* Dummy Main *)
END.
```

Parameter Values

Backspace	CTRL-H (08)	
Editor "Accept" Key	CTRL-C (03)	
Editor "Escape" Key	CTRL-Q (17)	
Erase Line	"O" (79)	
Erase Screen	"L" (76)	
Erase to End of Line	"I" (73)	
Erase to End of Screen	"J" (74)	
Has 8510A	FALSE	
Has Clock	FALSE	
Has Lower Case	TRUE	
Has Random Cursor Addr.	TRUE	
Has Slow Terminal	FALSE	
Key for Break	NUL (00)	
Key for Flush	CTRL-F (06)	
Key for Stop	CTRL-S (19)	
Key to Delete Character	CTRL-H (08)	
Key to Delete Line	CTRL-X (24)	
Key to End File	CTRL-C (03)	
Key to Move Cursor Down	"B" (66)	* These may be
Key to Move Cursor Left	"D" (68)	* altered by
Key to Move Cursor Right	"C" (67)	* the user for
Key to Move Cursor Up	"A" (65)	* convenience.
Lead-in from Keyboard	ESC (27)	
Lead-in to Screen	ESC (27)	

(Continued)

IBM 3101-22 Terminal Setup Parameters

Move Cursor Home	"H"	(72)
Move Cursor Right	"C"	(67)
Move Cursor Up	"A"	(65)
Non-printing Character	"?"	(63)
PREFIXED[Delete Character]	FALSE	
PREFIXED[Editor "Accept" Key]	FALSE	
PREFIXED[Editor "Escape" Key]	FALSE	
PREFIXED[Erase Line]	TRUE	
PREFIXED[Erase Screen]	TRUE	
PREFIXED[Erase to End of Line]	TRUE	
PREFIXED[Erase to End of Screen]	TRUE	
PREFIXED[Key for Break]	FALSE	
PREFIXED[Key for Flush]	FALSE	
PREFIXED[Key to Move Cursor Down]	TRUE	
PREFIXED[Key to Move Cursor Left]	TRUE	
PREFIXED[Key to Move Cursor Right]	TRUE	
PREFIXED[Key to Move Cursor Up]	TRUE	
PREFIXED[Key for Stop]	FALSE	
PREFIXED[Key to Delete Character]	FALSE	
PREFIXED[Key to Delete Line]	FALSE	
PREFIXED[Key to End File]	FALSE	
PREFIXED[Move Cursor Home]	TRUE	
PREFIXED[Move Cursor Right]	TRUE	
PREFIXED[Move Cursor Up]	TRUE	
PREFIXED[Non-printing Character]	FALSE	
Screen Height	24	
Screen Width	80	
Student	FALSE	
Vertical Move Delay	0	

(Continued)

VC 404 Terminal Setup Parameters

GOTO Code

```
(*SU-*)
PROGRAM GOXY; (* For VC 404 Video Terminal *)
PROCEDURE FGOTOXY(X,Y:INTEGER);
VAR SEND: PACKED ARRAY [0..2] OF 0..255;
BEGIN
  IF X>79 THEN X:=79 ELSE IF X<0 THEN X:=0;
  IF Y>23 THEN Y:=23 ELSE IF Y<0 THEN Y:=0;
  SEND[0]:=16; (* Lead-in and DC1 *)
  SEND[1]:=32+Y; SEND[2]:=32+X;
  UNITWRITE(2,SEND,3,0,12)
END;

BEGIN (* DUMMY MAIN *)
END.
```

Parameter Values

Backspace	CTRL-H (08)	
Editor "Accept" Key	CTRL-C (03)	
Editor "Escape" Key	ESC (27)	
Erase Line	NUL (00)	
Erase Screen	CTRL-X (24)	
Erase to End of Line	CTRL-V (22)	
Erase to End of Screen	CTRL-W (23)	
Has 8510A	FALSE	
Has Clock	FALSE	
Has Lower Case	TRUE	
Has Random Cursor Addr.	TRUE	
Has Slow Terminal	FALSE	
Key for Break	NUL (00)	
Key for Flush	CTRL-F (06)	
Key for Stop	CTRL-S (19)	
Key to Delete Character	CTRL-H (08)	
Key to Delete Line	CTRL-X (24)	
Key to End File	CTRL-C (03)	
Key to Move Cursor Down	CTRL-J (10)	* These may be
Key to Move Cursor Left	CTRL-H (08)	* altered by
Key to Move Cursor Right	CTRL-U (21)	* the user for
Key to Move Cursor Up	CTRL-Z (26)	* convenience.
Lead-in from Keyboard	NUL (00)	
Lead-in to Screen	NUL (00)	

(Continued)

VC 404 Terminal Setup Parameters

Move Cursor Home	CTRL-Y (25)
Move Cursor Right	CTRL-U (21)
Move Cursor Up	CTRL-Z (26)
Non-printing Character	"?" (63)
PREFIXED[Delete Character]	FALSE
PREFIXED[Editor "Accept" Key]	FALSE
PREFIXED[Editor "Escape" Key]	FALSE
PREFIXED[Erase Line]	FALSE
PREFIXED[Erase Screen]	FALSE
PREFIXED[Erase to End of Line]	FALSE
PREFIXED[Erase to End of Screen]	FALSE
PREFIXED[Key for Break]	FALSE
PREFIXED[Key for Flush]	FALSE
PREFIXED[Key to Move Cursor Down]	FALSE
PREFIXED[Key to Move Cursor Left]	FALSE
PREFIXED[Key to Move Cursor Right]	FALSE
PREFIXED[Key to Move Cursor Up]	FALSE
PREFIXED[Key for Stop]	FALSE
PREFIXED[Key to Delete Character]	FALSE
PREFIXED[Key to Delete Line]	FALSE
PREFIXED[Key to End File]	FALSE
PREFIXED[Move Cursor Home]	FALSE
PREFIXED[Move Cursor Right]	FALSE
PREFIXED[Move Cursor Up]	FALSE
PREFIXED[Non-printing Character]	FALSE
Screen Height	24
Screen Width	80
Student	FALSE
Vertical Move Delay	10

Issued 23 Sep 81

Pascal

Page 6100.016.01

NESTING PASCAL PROCEDURES AND FUNCTIONS

Apple Pascal will allow nesting of procedures inside functions and functions inside procedures. Procedures and functions may be nested up to 15 deep.

Segment procedures and segment functions may be nested up to six levels deep, although the resultant code will not be nested. Please refer to the Apple Pascal Language Reference Manual for more details on segmentation.

MEMORY SPACE AVAILABLE

The maximum space in Pascal version 1.1 is about 37K with the command level swapping OFF, and nearly 40K with swapping ON. Please refer to the Apple Pascal Update addendum for further information about the swapping option.

A "stack overflow" execution error is caused by literally running out of room while the program is being executed. When the Program Stack pointer and the Top-of-Heap pointer cross, the system has used all the memory available, and the attempt to use more causes the error message to appear. This situation can occur in many ways, but the two most common causes will be discussed here.

Probably the most common situation is found when the program is using the TURTLEGRAPHICS library unit. In order to protect the hi-res page from being overwritten by program code or by data, the graphics unit sets the heap pointer to above the hi-res area. This insures that the system will not attempt to use that area, since it has already been allocated. However, this has the effect of reducing the available memory by a significant amount, and a stack overflow situation will occur more frequently because of this.

A second, fairly common situation is found when attempting to open a disk file. If you have a program which is already quite large, any attempt to allocate a large block of memory can trigger the stack overflow. Before the operating system can open the disk file, it must read in the directory of the volume. This is a 2K block of information whose memory space is allocated at the time of the disk access. Therefore, if there isn't enough space for the directory, the system will report the error.

The Pascal intrinsic function MEMAVAIL can be used to report the current number of words available to your program. The value of MEMAVAIL will vary depending upon the size of your program, library units called, and heap utilization.

If a stack overflow execution error occurs, place a WRITELN (MEMAVAIL) statement immediately before the point at which the error occurs. The value returned will indicate whether the error is due to running out of memory, or due to some other problem not yet uncovered. If the value of MEMAVAIL drops below about 1024 words, the next directory access will trigger the error condition. A large program can often be restructured so that memory usage will be more efficient.

Issued 23 Sep 81

Pascal

Page 6100.018.01

IMMEDIATE MODE EXECUTION

Apple Pascal does not have an "immediate mode" where program instructions can be executed individually, since it is a compiled language. The command line is used for accessing specific system functions such as the Compiler, Editor, Filer, Assembler, or Linker.

Issued 23 Sep 81

Pascal

Page 6100.019.01

PASCAL TURNKEY SYSTEM

A Pascal turn-key system can be created by transferring the desired program code file to the BOOT diskette, and changing the code file name to SYSTEM.STARTUP. If no file by that name is present on the boot volume, the system will default to the command line.

A SYSTEM.STARTUP code file can be executed from the command line by typing "X" (Execute) "SYSTEM.STARTUP." (notice the trailing period). The final period prevents the operating system from appending the .CODE suffix to the file name before executing.

EXTERNAL REFERENCES

The use of "external" linking in Apple Pascal is restricted to assembly routines. Pascal (P-code) procedures which are to be compiled separately should be put into a library unit and accessed through the "USES" command.

Assembly code may also be placed into library units, provided it is called from the Pascal level. Assemble the code file(s), and then create a Pascal unit which refers to each .PROC and .FUNC of the assembly code as "external". To create an intrinsic unit which contains assembly code, link the external references to the compiled P-code unit structure, and then install this completed unit in the SYSTEM.LIBRARY on your boot disk.

Regular units which contain references to assembly code are not linked prior to installation in the library file. Instead, install the unlinked Pascal unit structure in the library (either SYSTEM.LIBRARY or your own library file), and keep the assembly code file(s) on hand. When the host program is linked with this regular unit, also link the assembly routines into the final code file.

Issued 29 Oct 81

Pascal

Page 6100.021.01

MAKING A COPY OF THE "BASICS" DISKETTE

The Language System and DOS 3.3 diskette named BASICS is a 16 sector PASCAL format diskette and should be copied using the PASCAL Filer Transfer or the DOS 3.3 COPY program. FID will not be able to copy this diskette.

Issued 29 Oct 81

Pascal

Page 6100.022.01

CHAINING IN PASCAL

Pascal version 1.0 does not allow one program to call (run) another program. A program containing segment procedures can be used to simulate chaining.

Version 1.1 of Apple Pascal contains the library unit CHAINSTUFF, which will allow chaining without the use of segment procedures. Please note that this type of chaining does not preserve the variable space. Refer to the Apple Pascal Language Manual addendum for more information about chaining.

Issued 29 Oct 81

Pascal

Page 6100.023.01

TERMINAL DRIVER PACKAGE FOR PASCAL

P.I.T.S. is a terminal driver packages with upload and download functions.
The package will also allow Pascal to support the Hayes Micromodem II.
For more information, please contact:

Craig Vaughan
Software Sorcery Inc.
7927 Jones Branch Drive, Suite 400
McLean, VA 22102
(703) 385-2944

Issued 29 Oct 81

Pascal

Page 6100.024.01

BOOLEANS

In Pascal version 1.0, the Boolean operator, NOT, does not invert the Boolean variable correctly. For example, NOT FALSE doesn't equal TRUE. It is suggested that the 'NOT' be replaced by

```
IF A=TRUE THEN A:=FALSE ELSE A:=TRUE
```

when working with Booleans in version 1.0.

Booleans will work correctly in version 1.1. Please refer to pages 8-9 of the Pascal Update addendum for a description of the corrections made to Booleans.

Issued 5 Mar 82

Pascal

Page 6100.025.01

PASCAL LONG INTEGER COMPARES - VERSION 1.0 ONLY

An error in the implementation of Pascal long integers results in a stack crash during a compare operation (version 1.0 only). The attached program will repair the LONGINTEGER module in the 1.0 SYSTEM.LIBRARY. The problem has been corrected in Pascal version 1.1, so that long integers will work as expected.

THIS PROGRAM IS PROVIDED FOR VERSION 1.0 SYSTEMS ONLY, AND SHOULD NOT BE USED ON VERSION 1.1.

```
(*$I-*)
PROGRAM FIXCOMPARE;

TYPE DISKINFO = RECORD
    DADDR: INTEGER;
    LENG : INTEGER
    END;

    NAME = PACKED ARRAY [1..8] OF CHAR;

    SEGDIC = RECORD
        DINFO: ARRAY [0..15] OF DISKINFO;
        SEGNAME: ARRAY [0..15] OF NAME;
        FILLER: ARRAY [1..416] OF INTEGER
        END;

    BLOCK = PACKED ARRAY [0..511] OF 0..255;

VAR I,J: INTEGER;
    NOTFOUND: BOOLEAN;
    F: FILE;
    S: STRING;
    BLOCKZERO: SEGDIC;
    DATA: BLOCK;

PROCEDURE READERROR;
BEGIN
    WRITE ('BAD BLOCK IN LIBRARY');
    EXIT (PROGRAM)
END;
```

(Continued)

```
BEGIN
  NOTFOUND:=TRUE;
  REPEAT
    WRITE (^NAME OF LIBRARY FILE:);
    READLN (S);
    RESET (F,S);
  UNTIL EOF OR (IORESULT=0);
  IF BLOCKREAD (F,BLOCKZERO,1,0)<>1 THEN READERROR;
  FOR I:=0 TO 15 DO
    BEGIN
      IF BLOCKZERO.SEGNAME[I]=^LONGINTI^ THEN
        BEGIN
          NOTFOUND:=FALSE;
          J:=BLOCKZERO.DINFO[I].DADDR+1;
          IF BLOCKREAD (F,DATA,1,J)<>1 THEN READERROR;
          IF DATA[495]=244 THEN
            IF DATA[494]=208 THEN
              BEGIN
                WRITELN (^LONG INTEGER PATCH BEING MADE^);
                DATA[494]:=240;
                IF BLOCKWRITE (F,DATA,1,J)=1 THEN
                  WRITELN (^PATCH COMPLETE^)
                ELSE
                  WRITELN (^ERROR WHILE WRITING PATCH, SEGMENT ^,I);
              END
            ELSE
              IF DATA[494]=240 THEN
                WRITELN (^SEGMENT ^,I,
                  ^ - LONG INTEGER UNIT HAS ALREADY BEEN FIXED^)
              ELSE WRITELN (^CAN^T RECOGNIZE^,
                ^ LONG INTEGER UNIT IN SEGMENT ^,I)
              ELSE WRITELN (^CAN^T RECOGNIZE^,
                ^ LONG INTEGER UNIT IN SEGMENT ^,I)
            END
          END;
        IF NOTFOUND THEN WRITELN (^NO LONG INTEGER UNIT FOUND^)
        ELSE
          WRITELN (^PROGRAM COMPLETE^)
        END.
```

THE KEYPRESS FUNCTION

KEYPRESS is a widely used function which resides in the unit APPLESTUFF. In many cases, it is the only routine being called from the unit, so it may be more efficient to refer to the routine without also loading the rest of the unit.

The attached listing is the function KEYPRESS, which can be assembled for linkage to your host program as an "external" function. Follow the instructions provided with the examples shown on pages 136-182 of the Pascal Operating System Reference Manual.

```

        .FUNC KEYPRESS,0      ;0 words of parameters passed
;*****
;*
;*   FUNCTION KEYPRESS: BOOLEAN; EXTERNAL
;*
;*****
;
RETURN   .EQU 0              ;Storage for return address
CONCKVEC.EQU 0BF0A          ;Fixed address in BIOS
RPTR    .EQU 0BF18          ;Fixed buffer pointer
WPTR    .EQU 0BF19          ;Fixed buffer pointer
VERSION .EQU 0BF21          ;System version number
KEYBOARD.EQU 0C000          ;Keyboard hardware
CONCK   .EQU 0FF5C          ;Way to get CONCK in old system
    
```

(Continued)

```

        PLA
        STA RETURN
        PLA
        STA RETURN+1
        PLA
        PLA
        PLA
        PLA
        PLA
        LDA #0
        PHA
        LDA KEYBOARD
        BMI TRUE
        LDA VERSION
        BNE $1
        JSR CONCK
        JMP $2
$1      JSR CONCKVEC
$2      LDA RPTR
        CMP WPTR
        BEQ EMPTY
TRUE    LDA #1
        BNE KPDONE
EMPTY  LDA #0
KPDONE PHA
        LDA RETURN+1
        PHA
        LDA RETURN
        PHA
        RTS
        .END
    
```

;Pop 4 bytes stack bias for function

;Return MSB zero

;Jump if not original Pascal version

;Check console

;Char in buffer?

;Yes, return KEYPRESS=TRUE

;Always taken

;No, return KEYPRESS=FALSE

;Push LSB result

;Restore return address

(Continued)

The following sample program illustrates the use of KEYPRESS as an externally linked routine. Follow the instructions given on pages 136-182 of the Pascal Operating System Reference Manual for assembling and linking external code.

```
PROGRAM PRESSTEST;

VAR I: INTEGER;

FUNCTION KEYPRESS: BOOLEAN; EXTERNAL;

BEGIN
  I:=0;
  REPEAT
    WRITELN (I);
    I:=I+1;
  UNTIL KEYPRESS
  END.
```

Issued 29 Oct 81

Pascal

Page 6100.027.01

THE SCAN FUNCTION

The Pascal SCAN function scans a range of memory, searching for a specified target. The function takes the form

SCAN (LIMIT, PEXPR, SOURCE)

where LIMIT is the number of bytes to be scanned, PEXPR defines the target of the scan, and SOURCE is the variable being scanned. Pages 51-52 of the Pascal Language Manual describe this function in greater detail.

SCAN can be used on any variable except file types. When scanning a string (S, for example), SOURCE should be specified as S[1], to avoid having SCAN also look at the length parameter stored in S[0], which will usually yield an erroneous value.

Issued 29 Oct 81

Pascal

Page 6100.028.01

TRUNC AND ROUND FUNCTIONS

The Pascal TRUNC and ROUND functions are present in Apple Pascal. The ROUND function is of the form $I:=ROUND(R)$ where I is an integer and R is a real number. Although the compiler will accept other forms of this command, this is the only legitimate form.

TRUNC takes the form $I:=TRUNC(R)$ where I is an integer and R is a real number, and $I:=TRUNC(L)$ where L is a long integer. In all cases, integer values must be in the range -32767 to +32767. If the long integer L is greater than MAXINT, a value range error will be returned. The compiler may accept other forms of this function, but these are the only correct forms according to UCSD definitions.

Issued 5 Mar 82

Pascal

Page 6100.029.01

TTL0UT

The Pascal TTL0UT procedure in the version 1.0 unit APPLESTUFF (in the SYSTEM.LIBRARY) does not function properly. The routine can be assembled and linked externally, using the listing and example shown on pages 136-182 of the Apple Pascal Operating System Reference Manual.

This problem has been corrected for version 1.1, so that TTL0UT will function correctly from APPLESTUFF. If desired, TTL0UT can still be assembled and linked externally, to avoid using all of APPLESTUFF.

Issued 29 Oct 81

Pascal

Page 6100.030.01

THE CALC UTILITY

APPLE3:CALC.CODE allows the evaluation of simple arithmetic expressions, using +, -, *, / and (). Trigonometric functions and exponentiation will cause the program to abort with an "unimplemented instruction" error. To exit the program, simply press RETURN in response to the prompt. Please refer to pages 216-218 of the Pascal Operating System Reference Manual for a more detailed discussion of the CALC program.

Issued 29 Oct 81

Pascal

Page 6100.031.01

PROPOSED ISO STANDARDS FOR PASCAL

For information on the proposed ISO standard for Pascal, please write to:

Dr. Carol Sledge, Chairman X3J9 Committee
On-Line Systems Inc.
115 Evergreen Heights Drive
Pittsburgh, PA 15229
(412) 931-7600

CONVERTING STRINGS TO NUMERIC VARIABLES

Apple Pascal doesn't allow input editing for integer or real variables. This can cause a lot of trouble if the wrong data is entered by mistake: either the program is stuck with bad data, or worse yet, the system may crash.

STRINGSTUF is an intrinsic unit which is designed to avoid this problem. All data is entered in the form of strings, then converted to the appropriate data format by the unit. This allows input editing while the data is in string form.

NOTE: The STRINGSTUF unit and accompanying demo program are designed to run in Pascal version 1.1 only. Long integers are not supported in STRINGSTUF.

STRINGSTUF may be located in any unused segment number 17-31. Install the unit in the SYSTEM.LIBRARY following the instructions in the Pascal Operating System Reference Manual, pages 186-193.

```
(*$S+*)
```

```
(* $LPRINTER:*) (* Get a compile listing..optional *)
```

```
UNIT STRINGSTUF; INTRINSIC CODE 26;
```

```
INTERFACE
```

```
TYPE STRING255=STRING[255];
```

```
FUNCTION STRFP (VAR STR:STRING255; VAR FP:REAL): BOOLEAN;
```

```
FUNCTION STRINT (VAR STR:STRING255; VAR INT:INTEGER): BOOLEAN;
```

```
IMPLEMENTATION
```

```
FUNCTION STRFP; (* String to Real *)
```

```
CONST MAXREAL=1.70E37; (* Max/10 *)
```

```
MINREAL=1.2E-37; (* Min/10 *)
```

```
VAR DEC,DEX,EDP,INX,LEN: INTEGER;
```

```
DP,EX,IM,MN,MX,SN: BOOLEAN;
```

```
CH: CHAR;
```

```
NUMERIC,EXPONENT,MODIFIER: SET OF CHAR;
```

(Continued)

```

PROCEDURE TERMINATE;
VAR I: INTEGER;
BEGIN
  IF MX THEN DEX:=-DEX;
  EDP:=EDP+DEX-DEC;
  IF EDP<0
    THEN FOR I:=1 TO -EDP DO
      IF FP>=MINREAL THEN FP:=FP/10.0
      ELSE FP:=0 (* Underflow => 0 *)
    ELSE FOR I:=1 TO EDP DO
      IF FP<=MAXREAL THEN FP:=FP*10.0
      ELSE EXIT (STRFP); (* Overflow *)
  IF MN THEN FP:=-FP;
  STRFP:=TRUE;
  EXIT (STRFP) (* Successful conversion *)
END;

PROCEDURE SEARCH;
BEGIN
  WHILE INX<=LEN DO
    IF STR[INX] IN NUMERIC
      THEN BEGIN
        (*$R-*)
          WHILE (INX>1) AND (STR[INX-1] IN EXPONENT+MODIFIER)
            (*$R+*)
              DO INX:=INX-1;
          EXIT(SEARCH) (* Found start of number *)
        END
      ELSE INX:=INX+1;
  EXIT (STRFP) (* Non-numeric string *)
END;

BEGIN (*STRFP*)
  NUMERIC:=[^0^..9^];
  EXPONENT:=[^E^,^e^];
  MODIFIER:=[^+^,^-^,^.^,^,^];
  DP:=FALSE; EX:=FALSE; IM:=TRUE;
  MN:=FALSE; MX:=FALSE; SN:=FALSE;
  DEC:=0; DEX:=0; EDP:=0; INX:=1;
  LEN:=LENGTH(STR); FP:=0;
  STRFP:=FALSE;
  SEARCH; (* Find start of number *)
  WHILE INX<=LEN DO BEGIN
    CH:=STR[INX];
    IF CH IN NUMERIC+EXPONENT+MODIFIER
      THEN BEGIN
        IF CH IN NUMERIC
          THEN IF EX
            THEN BEGIN
              IF DEX<1000 THEN (* Exponent *)

```

(Continued)

```

        DEX:=DEX*10+ORD(CH)-ORD('^0');
        SN:=TRUE
    END

    ELSE BEGIN
        IF FP<1.0E8
            THEN
                FP:=FP*10+ORD(CH)-ORD('^0') (* Mantissa *)
            ELSE EDP:=EDP+1;
            IF DP THEN (* Digits to right of DP *)
                DEC:=DEC+1; IM:=FALSE;
            SN:=TRUE
        END
    ELSE CASE CH OF
        '+': IF SN THEN (* Duplicate '+' sign *)
            TERMINATE
            ELSE SN:=TRUE;
        '-': IF SN THEN (* Duplicate '-' sign *)
            TERMINATE
            ELSE BEGIN
                IF EX THEN MX:=TRUE
                ELSE MN:=TRUE;
                SN:=TRUE
            END;
        '.': IF DP OR EX THEN (* Duplicate '.' *)
            TERMINATE
            ELSE DP:=TRUE;
        'E', 'e': IF EX THEN TERMINATE (* Duplicate 'E' *)
            ELSE BEGIN
                IF IM THEN (* Implied mantissa *)
                    FP:=1.0;
                EX:=TRUE;
                SN:=FALSE
            END;
    END; (*CASE*)
    INX:=INX+1
    END
    ELSE TERMINATE (* End of number *)
    END;
    TERMINATE (* End of string *)
    END;

FUNCTION STRINT; (* String to Integer *)

VAR FP: REAL;

BEGIN
    STRINT:=STRFP (STR,FP); (* First convert to real *)
    IF ABS(FP)<=MAXINT

```

(Continued)


```
      THEN INT:=ROUND(FP) (* then round to integer *)
      ELSE BEGIN
        STRINT:=FALSE; (* Integer out of range *)
        INT:=0
      END
    END;

  BEGIN (* Unit Initialization *)
  END.
```

(Continued)

This sample program illustrates the use of STRINGSTUF. The compiler \$V- option is required to override the normal string length checking.

```
PROGRAM STRINGTEST;

USES STRINGSTUF; (* tests StringStuf library unit *)

VAR INPUT,STR: STRING;
    INT: INTEGER;
    FP: REAL;

BEGIN
    PAGE (OUTPUT);
    WRITELN (^STRINGSTUF STRING => NUMERIC CONVERSION: ^);
    REPEAT
        WRITELN;
        WRITE (^STRING : ^);
        READLN (INPUT);
        (*$V-*)
        IF STRFP (INPUT,FP) THEN
            BEGIN
                WRITELN (^ REAL: ^,FP);
                IF STRINT (INPUT,INT)
                    THEN WRITELN (^INTEGER: ^,INT)
                    ELSE WRITELN(^INTEGER: OUT OF RANGE.^)
            END
        ELSE WRITELN(^NO NUMERIC VALUE IN STRING.^);
        (*$V+*)
    UNTIL INPUT='^'
END.
```

INTRINSIC UNIT FOR SILENTYPE PROCEDURES

Appendix D (pages 53-59) of the Silentype reference manual indicates that all the parameter procedures can be placed into an intrinsic library unit. This is a handy and valuable unit to have if you plan to use the Silentype printer in a Pascal program.

To extend the usability of the Silentype unit, two procedures have been added to this listing, and another has been changed. These procedures will be discussed here; please refer to the Silentype manual for a description of the remainder of the routines.

PROCEDURE SETPAGE (PAGE: INTEGER)

This procedure will allow you to change from one hi-resolution graphics screen to the other, although only the first hi-res page is used in the Pascal system.

Values: 1=first page
2=second page

PROCEDURE COLDSTART

The COLDSTART procedure will reset all the Silentype parameters to the default values as specified in the printer manual. This procedure is equivalent to turning off the power to the printer.

Values: none required.

PROCEDURE WARMSTART

The RESTORE procedure as listed in the Silentype manual has been revised and renamed WARMSTART. Included in the revision is the SETPAGE procedure. This procedure differs from COLDSTART in that the parameters set by WARMSTART are user-definable; that is, they may reflect the particular setup that you find most applicable to your needs. (For instance, you may want a darkness setting of 3 instead of 5.)

(Continued)

INITIALIZATION CODE

The initialization code consists of a call to the COLDSTART procedure. This will be executed when the unit is first brought into the system from the library. This will insure that the printer does not have any unwanted values remaining from previous usages in either Pascal or in Basic. If desired, you may change this to WARMSTART or other code, or leave the initialization section empty (but the BEGIN and END must be present).

INSTALLING THE UNIT

The Silentype unit is ready to type in, compile, and install in your copy of SYSTEM.LIBRARY. You may wish to change the INTRINSIC CODE value, or the contents of WARMSTART to suit your individual needs. Refer to pages 186-193 of the Apple Pascal Operating System Reference manual for instructions for the LIBRARY program which will place the compiled Silentype unit into your SYSTEM.LIBRARY.

```
(*$$+*)  
UNIT SILENTYPE; INTRINSIC CODE 27;
```

INTERFACE

```
PROCEDURE ROMENABLE;  
PROCEDURE SETBYTEVALUE (LOC,VALUE: INTEGER);  
FUNCTION BYTEVALUE (LOC: INTEGER): INTEGER;  
PROCEDURE SEND (CH: CHAR);  
PROCEDURE SETUNIDIRECT;  
PROCEDURE SETBIDIRECT;  
FUNCTION UNIDIRECT: BOOLEAN;  
PROCEDURE SETNEGATIVE;  
PROCEDURE SETPOSITIVE;  
FUNCTION NEGATIVE: BOOLEAN;  
PROCEDURE SETPAGE (PAGE: INTEGER);  
PROCEDURE SETDARK (DARKNESS: INTEGER);  
FUNCTION DARK: INTEGER;  
PROCEDURE SETFORM (LENGTH: INTEGER);  
FUNCTION FORM: INTEGER;  
PROCEDURE SETSPACE (LENGTH: INTEGER);  
FUNCTION SPACE: INTEGER;  
FUNCTION LEFTMARGIN: INTEGER;  
FUNCTION RIGHTMARGIN: INTEGER;  
PROCEDURE SETLEFTMARGIN (POSITION: INTEGER);  
PROCEDURE SETRIGHTMARGIN (POSITION: INTEGER);  
PROCEDURE PRINTBUFFER;  
PROCEDURE CLEARBUFFER;  
PROCEDURE FORMFEED;
```

(Continued)

```

PROCEDURE PRINTPIC;
PROCEDURE COLDSTART;
PROCEDURE WARMSTART;

```

IMPLEMENTATION

```

PROCEDURE ROMENABLE;
  CONST ROMSOFF= -12289;
        ROMON= -16128;
  TYPE WINDOW= PACKED ARRAY [0..0] OF 0..255;
  VAR ADDR: INTEGER;
        P: ^WINDOW;
  BEGIN
    ADDR:=ROMSOFF;
    MOVELEFT(ADDR,P,2);
    P^[0]:=0;
    ADDR:=ROMON;
    MOVELEFT(ADDR,P,2);
    P^[0]:=0
  END;

PROCEDURE SETBYTEVALUE;
  TYPE WINDOW= PACKED ARRAY [0..0] OF 0..255;
  VAR ADDR: INTEGER;
        P: ^WINDOW;
  BEGIN
    ROMENABLE;
    ADDR:=LOC;
    MOVELEFT(ADDR,P,2);
    P^[0]:=VALUE
  END;

FUNCTION BYTEVALUE;
  TYPE WINDOW= PACKED ARRAY [0..0] OF 0..255;
  VAR ADDR: INTEGER;
        P: ^WINDOW;
  BEGIN
    ROMENABLE;
    ADDR:=LOC;
    MOVELEFT(ADDR,P,2);
    BYTEVALUE:=P^[0]
  END;

PROCEDURE SEND;
  CONST PRINTUNIT= 6;
  BEGIN
    UNITWRITE(PRINTUNIT,CH,1,0,12)
  END;

```

(Continued)

```
PROCEDURE SETUNIDIRECT;  
  CONST MAXBYTE= 255;  
        DIRECTION= -12529;  
  BEGIN  
    SETBYTEVALUE(DIRECTION,MAXBYTE)  
  END;
```

(Continued)

```
PROCEDURE SETBIDIRECT;
  CONST MINBYTE= 0;
        DIRECTION= -12529;
  BEGIN
    SETBYTEVALUE(DIRECTION,MINBYTE)
  END;

FUNCTION UNIDIRECT;
  CONST MAXBYTE= 255;
        MINBYTE= 0;
        DIRECTION= -12529;
  BEGIN
    CASE BYTEVALUE(DIRECTION) OF
      MINBYTE: UNIDIRECT:=FALSE;
      MAXBYTE: UNIDIRECT:=TRUE
    END
  END;

PROCEDURE SETNEGATIVE;
  CONST MINBYTE= 0;
        FLIP= -12524;
  BEGIN
    SETBYTEVALUE(FLIP,MINBYTE)
  END;

PROCEDURE SETPOSITIVE;
  CONST MAXBYTE= 255;
        FLIP= -12524;
  BEGIN
    SETBYTEVALUE(FLIP,MAXBYTE)
  END;

FUNCTION NEGATIVE;
  CONST MAXBYTE= 255;
        MINBYTE= 0;
        FLIP= -12524;
  BEGIN
    CASE BYTEVALUE(FLIP) OF
      MINBYTE: NEGATIVE:=TRUE;
      MAXBYTE: NEGATIVE:=FALSE
    END
  END;

PROCEDURE SETPAGE;
  CONST GRAPHPAGE= -12525;
  BEGIN
    IF PAGE=2 THEN PAGE:=64
      ELSE PAGE:=32;
    SETBYTEVALUE(GRAPHPAGE,PAGE)
  END;
```

(Continued)

```
PROCEDURE SETDARK;
  CONST INTEN= -12528;
  BEGIN
    IF DARKNESS<0 THEN DARKNESS:=0;
    IF DARKNESS>7 THEN DARKNESS:=7;
    SETBYTEVALUE(INTEN,DARKNESS)
  END;

FUNCTION DARK;
  CONST INTEN= -12528;
  BEGIN
    DARK:=BYTEVALUE(INTEN)
  END;

PROCEDURE SETFORM;
  CONST FORMLength= -12531;
  BEGIN
    IF LENGTH<0 THEN LENGTH:=0;
    IF LENGTH>255 THEN LENGTH:=255;
    SETBYTEVALUE(FORMLength,LENGTH)
  END;

FUNCTION FORM;
  CONST FORMLength= -12531;
  BEGIN
    FORM:=BYTEVALUE(FORMLength)
  END;

PROCEDURE SETSPACE;
  CONST INCR= -12530;
  BEGIN
    IF LENGTH<1 THEN LENGTH:=1;
    IF LENGTH>252 THEN LENGTH :=252;
    SETBYTEVALUE(INCR,LENGTH)
  END;

FUNCTION SPACE;
  CONST INCR= -12530;
  BEGIN
    SPACE:=BYTEVALUE(INCR)
  END;

FUNCTION LEFTMARGIN;
  CONST LMAR= -12527;
  BEGIN
    LEFTMARGIN:=BYTEVALUE(LMAR)
  END;
```

(Continued)


```
FUNCTION RIGHTMARGIN;
  CONST RMAR= -12526;
  BEGIN
    RIGHTMARGIN:=BYTEVALUE(RMAR)
  END;

PROCEDURE SETLEFTMARGIN;
  CONST LMAR= -12527;
  BEGIN
    IF POSITION>=RIGHTMARGIN THEN POSITION:=RIGHTMARGIN-1;
    IF POSITION<0 THEN POSITION:=0;
    SETBYTEVALUE(LMAR,POSITION)
  END;

PROCEDURE SETRIGHTMARGIN;
  CONST RMAR= -12526;
  BEGIN
    IF POSITION<=LEFTMARGIN THEN POSITION:=LEFTMARGIN+1;
    IF POSITION>83 THEN POSITION:=83;
    SETBYTEVALUE(RMAR,POSITION)
  END;

PROCEDURE PRINTBUFFER;
  CONST CF= 6; (* ASCII FOR CONTROL-F *)
  BEGIN
    SEND(CHR(CF))
  END;

PROCEDURE CLEARBUFFER;
  CONST PRINTUNIT= 6;
  BEGIN
    UNITCLEAR(PRINTUNIT)
  END;

PROCEDURE FORMFEED;
  CONST FF= 12; (* ASCII FOR FORM-FEED *)
  BEGIN
    SEND(CHR(FF))
  END;

PROCEDURE PRINTPIC;
  CONST CQ= 17; (* ASCII FOR CONTROL-Q *)
  VAR BIDIRECT: BOOLEAN;
  BEGIN
    IF UNIDIRECT=TRUE THEN BIDIRECT:=FALSE ELSE BIDIRECT:=TRUE;
    IF BIDIRECT THEN SETUNIDIRECT;
    SEND(CHR(CQ));
    IF BIDIRECT THEN SETBIDIRECT
  END;
```

(Continued)

```
PROCEDURE COLDSTART;
  CONST MINBYTE= 0;
        DEFAULT= -12506;
  BEGIN
    SETBYTEVALUE(DEFAULT,MINBYTE)
  END;

PROCEDURE WARMSTART;
  BEGIN
    SETBIDIRECT;
    SETPOSITIVE;
    SETPAGE(1);
    SETDARK(5);
    SETFORM(40);
    SETSPACE(2);
    SETLEFTMARGIN(2);
    SETRIGHTMARGIN(81);
  END;

BEGIN (* INITIALIZATION SECTION *)
  COLDSTART;
END.
```

OPERAND FORMATS

A useful and often important piece of information concerns the internal structure of Pascal variables. This information can be very useful when sending data (especially complex data formats such as strings) to an assembly routine from a Pascal host program. This article describes a few of the more commonly used variable types. For a complete description of the more complex variables, including records and arrays, see pages 227-228 of the Apple Pascal Operating System Reference Manual.

Machine language (assembly) routines are commonly used when speed is critical, and when the code must access other assembly routines such as PROMs or I/O drivers which can't be re-assembled as part of the program. Also, bit manipulations such as right-shift are much easier to do in assembly than in Pascal.

In the UCSD Pascal system, it's a fairly simple matter to create short assembly programs which can be linked into a Pascal host program. In some cases, it may be sufficient to merely call the assembly routine; however, most routines require data in order to be useful. The means by which data is passed to or from these routines is called a "parameter".

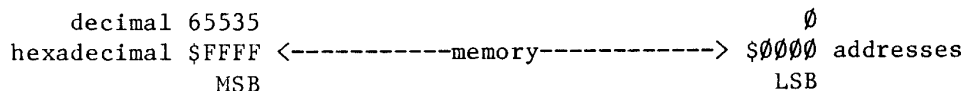
A parameter is a temporary variable created by Pascal for the purpose of passing data to or from a subroutine. The term "VAR parameter" implies that the address of the actual variable is passed to the subroutine as a parameter instead of its value.

Certain types of variables may be passed by value, but any variable may be passed by name by simply declaring it to be a VAR parameter. Pascal does not allow parameters of variable length (with the exception of certain sets and long integers) to be passed on the CPU stack, since this could exceed the stack capacity and crash the operating system, so these parameters are automatically used as if defined as VAR parameters. A good explanation of the various ways of passing parameters may be found in Peter Grogono's book, "Programming in Pascal".

(Continued)

Before delving into the details, let's define some terms and conventions which we'll use later on:

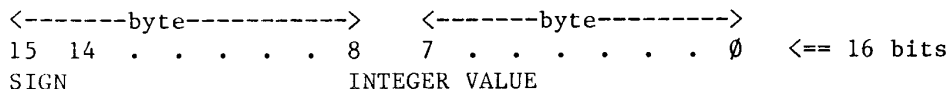
BIT = a binary digit (0 or 1). A bit is the smallest unit of information which can be stored in a computer.
 NYBBLE = 4 bits (half a byte). A hexadecimal digit is one nybble (pronounced "nibble").
 BYTE = 8 bits (2 nybbles). This is the unit of storage which the 6502 processor uses.
 WORD = 2 bytes (16 bits). A word is the unit of information which Pascal uses.
 LSB = least significant bit
 MSB = most significant bit



This diagram of memory structure will be used in describing the variable formats. Usually, when you write down a number, you write it from left to right. However, Pascal reads data from memory from right to left starting at the least significant byte.

INTEGERS

Integers, in UCSD Pascal, are whole numbers in the range of -32768 to +32767. They are stored in one word (2 bytes). Negative integers are represented in "two's complement", which means that they appear to have positive values (> 32767). By subtracting 2^{16} (65536) from this positive value, the negative integer is revealed. Similarly, large positive integers are stored as a complementary negative numbers (remember Integer Basic?). The sign bit (MSB) is 0 if positive, 1 if negative.



Example: the number 3 is represented in binary as:

MSB 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 LSB

However, -3 shows up as

MSB 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 LSB

which also reads as 65533 (or 65536-3)!

Integers may be passed by value or as VAR parameters.

(Continued)

REALS

Real number, in UCSD Pascal, are floating point numbers in the range of +/- 1.17550E-38 to +/-3.40282E+38. Real numbers use four bytes (2 words). The binary representation is similar to the proposed IEEE standard for floating point numbers:



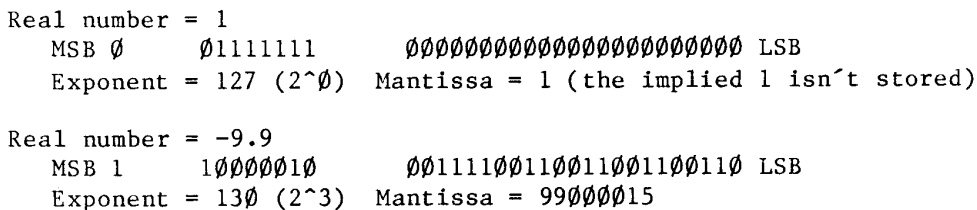
"Mantissa" is the name given to the decimal portion of a number which is expressed in scientific (exponential) notation. The "exponent" indicates the power to which the mantissa is raised. The exponent is represented in base 2 (2^n). In decimal, the number 3 x 10^2 can be seen as a mantissa of 3, an exponent of 2, in base 10 (decimal).

The sign bit refers to the sign of the mantissa, and is 0 if positive, 1 if negative. The exponent is "offset" by 127; that is, a value of 127 in the exponent field corresponds to an exponent of 0. Similarly, if the value is 1, the exponent is -126, and if the field is 254, the exponent is +127. A value of 0 indicates that the real number is 0.

The mantissa of the real number is stored in normalized format in bits 0-22. "Normalizing" a number means adjusting it so that the highest bit is significant (a 1). The exponent indicates how many times (and in which direction) the value was shifted during normalization.

Notice that the MSB of the mantissa of any non-zero number which has been normalized is always a one. The number zero can be treated as a special case by simply setting the exponent to zero. So, to gain additional precision, the mantissa has an implied "1" which is not stored, resulting in a functional 24-bit mantissa, even though only 23 bits are actually used. This gives slightly more than 6 decimal places (single precision) accuracy.

To make this clearer, let's look at some examples:



In the second example, the real number (in binary) appears as 1001.1110011 etc.. During normalization, the decimal point is moved to the left 3 times (incrementing the exponent), and the most significant bit becomes implied. The sign bit is 1, indicating that the number is negative.

(Continued)

Real numbers may be passed by value, or may be defined as VAR parameters and passed by address.

CHARACTERS

Characters, by ASCII definition, are simply integers in the range of 0 to 255. Characters take up one word of storage. The ASCII value of the character is stored in the least significant byte. The most significant byte is not used by Pascal and should be ignored.

15 8 7 0 <= 16 bits
 unused ASCII

EXAMPLE: the character "A" has an ASCII value of 65 (hexadecimal 41). Represented in binary, this would be:

MSB x x x x x x x x 0 1 0 0 0 0 1 LSB
 <----- not used-----> 4 (hex) 1

Characters can be passed as either actual parameters (pass by value) or VAR parameters (pass by address).

STRINGS

A string is a packed array of characters which can be from one to 256 bytes long. The first byte of a string always contains a number from 0 to 255 which indicates the length of the string. One character is stored per byte, and the string ends on a word boundary; that is, if the last character in the string is the first byte of a new word, the other byte of the word is also reserved, but is not used by the string.

Each character of the string can be accessed in a packed array of characters; however, you cannot access the length byte (the 0th element). Doing so will promptly generate the message: "Value Range Error".

EXAMPLE: The string "ABCD" has a length of 4 and would look like this:

S[4] S[3] S[2] S[1] S[0]
 MSB 01000100 01000011 01000010 01000001 00000100 LSB
 "D" "C" "B" "A" 4

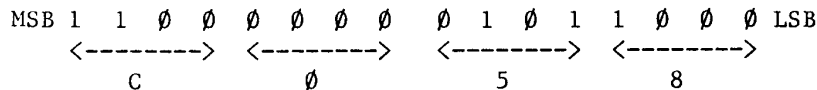
Pascal always passes strings by address, since the length may vary.

(Continued)

POINTERS

Address pointers are UNSIGNED integers which occupy 1 word of storage. The format is the same as for integers, except that the values range from 0 to 65535. The value of a pointer, in this implementation of Pascal, is the memory address of the object being described.

EXAMPLE: The address of AN0 (one of the annunciator ports) is hex C058 (49240 decimal). This would be stored as:



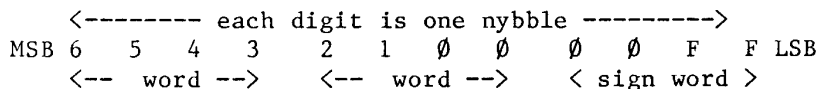
Pointers, like integers, may be passed by value or by reference (VAR parameter).

LONG INTEGERS

Long integers are a special type of variable which was first defined at UCSD as part of their extensions to the Pascal language. They are primarily used to handle calculations involving numbers which cannot be represented accurately in floating point (real) format and are too large to store in integer format.

Long integers are stored in BCD (binary coded decimal), one digit per nybble. One entire word is reserved for the sign of the long integer, and the variable must end on a word boundary. Four digits can be contained in one word, so the smallest definable long integer takes up two words of memory. The numbers are padded with leading zeroes when necessary to fill up the last word. The sign will be 0 if positive and 255 if negative (one byte is used).

To illustrate this, let's take a specific example: the long integer -123456 will take 3 words: one for the sign, and two for the digits, since they are stored in multiples of 4. The format will look like this:



Long integers should always be passed by address because they have a length which depends on their definition.

(Continued)

BOOLEANS

The Boolean, or binary, variable can have two values: TRUE and FALSE. This is most commonly used in determining yes/no conditions such as equality or set inclusion. This variable is stored in one word, although only the LSB (least significant bit) is used. TRUE is indicated by a 1, and FALSE shows as a 0.

```

MSB 15 . . . . . 8 7 . . . . . 0 LSB
                                Boolean
    
```

UCSD Pascal does not allow direct printing of Boolean variables. For example;

```

PROGRAM PRINTBOOLEAN;
VAR A: BOOLEAN;
BEGIN
  A := FALSE;
  WRITELN (A); (* this is illegal *)
  IF A = FALSE THEN WRITELN (^FALSE^) ELSE WRITELN (^TRUE^);
  (* this is correct *)
END.
    
```

Booleans are most efficient in packed arrays, where each bit of the word is utilized. DRAWBLOCK is probably the best-known example of this use. For an excellent example of the use of boolean packed arrays, look at the program GRAFDEMO on the Apple Pacal diskette APPLE3.

Boolean variables may be passed by value or by address.

OTHER TYPES

In addition to the previously mentioned standard types, Pascal allows the programmer to define a wide variety of non-standard variable types. Probably the most popular example of this is the set.

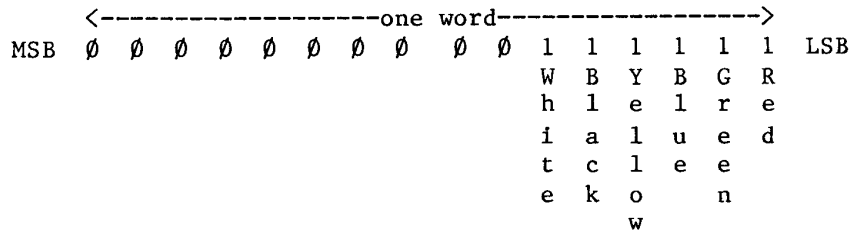
A set is an arbitrary collection of elements, where each element is assigned an ordinal position (that is, represented by a number). Each element of the set is represented by a name which can be any word of your own choosing (except for Pascal reserved words or other variable definitions already in use). Each name is then associated with one bit in the data definition beginning with bit 0. The set is stored in memory as a series of bits which are identified by the ordinal position of the element in the type definition. A set must end on a word boundary, so, for example, 17 elements would take up 2 words, even though only one bit of the second word is actually used.

(Continued)

EXAMPLE:

```
TYPE COLORS = (RED, GREEN, BLUE, YELLOW, BLACK, WHITE);
COLORSET = SET OF COLORS;
```

is a set of colors. Red occupies position 0, and white is position 5.



Sets may be passed either by address or by value, with certain restrictions. See page 203 of the Pascal reference manual for details.

In general, complex record types consist of one or more standard types which are stored as described. For the last word on Pascal data types, read Niklaus Wirth's Report in "User Manual and Report" by Jensen and Wirth.

REFERENCES

- Apple Pascal Reference Manual, by Apple Computer Inc. 1979.
- Apple Pascal Language Reference Manual, by Apple Computer, 1980.
- Apple Pascal Operating System Reference Manual, by Apple Computer, 1980.
- Programming in Pascal, by Peter Grogono, Addison Wesley, 1978.
- User Manual and Report, by Kathleen Jensen and Niklaus Wirth, Springer-Verlag, 1974.

PASCAL RUN-TIME ERRORS

Run-time errors generate a message plus a set of numbers that indicate which instruction was being executed when the error occurred. "S" stands for "SEGMENT", "P" for "PROCEDURE", and "I" for "INSTRUCTION COUNT". These can be correlated with the textfile by using the System List option (*\$L+*) or (*\$L<filename>*) when doing a compile. This will produce an annotated listing of the text, with the S, P, and I numbers included.

For example, here is a very simple program which has been compiled with the listing option:

```

(1)      (2)      (3:4)      (5)<--TEXT----->
  1      1      1:D      1 (*$LPRINTER:*)
  2      1      1:D      1 PROGRAM EXAMPLE;
  3      1      1:D      3
  4      1      1:D      3 VAR S:STRING;
  5      1      1:D      44
  6      1      1:Ø      Ø BEGIN
  7      1      1:1      Ø READLN(S);
  8      1      1:1      21 WRITELN(S)
  9      1      1:Ø      4Ø END.

```

KEY:

- (1) Line number of text
- (2) Segment number (S#) - S# values which do not appear in your listing indicate that the error has occurred in that segment of the operating system. S#Ø is SYSTEM.PASCAL, and 17-31 are usually SYSTEM.LIBRARY segments.
- (3) Procedure number (P#)- the procedure number within the segment designated by the S#.
- (4) Nesting level (D=declaration)
- (5) Instruction count (I#) - this is the instruction count from the beginning of the procedure. The number indicates the count at the beginning of each line, so a value between two lines simply means that the error occurred in the middle of the line.

A word of warning: When using Apple Pascal Version 1.Ø, do NOT use \$L+ as your listing option, as this will result in the loss of your code file and possibly your operating system. Instead, name a disk file that will be placed on a DIFFERENT volume from the destination of the code file, or better yet, use (*\$LPRINTER:*) to put the listing directly onto your printer. The \$L+ option works correctly in version 1.1.

REAL NUMBER FORMAT

Real numbers, in UCSD Pascal, are floating point numbers in the range of +/- 1.17550E-38 to +/-3.40282E+38. Real numbers use four bytes (2 words). The binary representation is similar to the proposed IEEE standard for floating point numbers:



"Mantissa" is the name given to the decimal portion of a number which is expressed in scientific (exponential) notation. The "exponent" indicates the power to which the mantissa is raised. The exponent is represented in base 2 (2^N). In decimal, the number 3 x 10^2 can be seen as a mantissa of 3, an exponent of 2, in base 10 (decimal).

The sign bit refers to the sign of the mantissa, and is 0 if positive, 1 if negative. The exponent is "offset" by 127; that is, a value of 127 in the exponent field corresponds to an exponent of 0. Similarly, if the value is 1, the exponent is -126, and if the field is 254, the exponent is +127. A value of 0 indicates that the real number is 0.

The mantissa of the real number is stored in normalized format in bits 0-22. "Normalizing" a number means adjusting it so that the highest bit is significant (a 1). The exponent indicates how many times (and in which direction) the value was shifted during normalization.

Notice that the MSB of the mantissa of any non-zero number which has been normalized is always a one. The number zero can be treated as a special case by simply setting the exponent to zero. So, to gain additional precision, the mantissa has an implied "1" which is not stored, resulting in a functional 24-bit mantissa, even though only 23 bits are actually used. This gives slightly more than 6 decimal places (single precision) accuracy.

Real numbers may be formatted for output using field-width designations. As described on pages 36-37 of the Apple Pascal language Reference Manual, the output specification has the following form:

```
REAL : FIELDWIDTH : FRACTIONLENGTH
```

where FIELDWIDTH is the minimum number of characters to be written, including the decimal point (default=1), and FRACTIONLENGTH is the number of digits to be written after the decimal place (default=5). Thus, a field specification of R:8:3 would indicate the real variable R is to be printed within a field size of 8 total, with 3 of those digits appearing to the right of the decimal. A FRACTIONLENGTH of zero is not allowed.

If the field size needed to display the variable accurately exceeds the formatting specification, the formatting will be ignored. If the size is smaller than FIELDWIDTH, the field will be padded with blanks to the left of the variable (right-justification).

LIBRARY UNITS

Modular programming means the separation of procedures and functions, or groups of them, from the main program. Source language modules are called units, and are incorporated in libraries for use with Pascal programs. Units may consist of procedures, functions, or a combination of these, in Pascal and assembly language.

Separate compilation has several advantages in the development of any program, because it allows you to approach the task as a group of smaller tasks which are linked together in a logical manner. The host program must contain a USES statement to utilize routines from the unit.

There are two principal kinds of units: Regular units, and Intrinsic units. Separate units are not included in the Apple Pascal implementation, and will not be discussed.

REGULAR UNITS

When a host program USES a regular unit, the unit's code is physically inserted into the host's codefile by the Linker. Once linked, the files need not be relinked unless either the unit or the host program is modified and recompiled.

Regular units, since they become part of the host file, may have references to file names. A regular unit under Pascal version 1.1 can use another regular unit, or it may use an intrinsic unit. RESTRICTION: In version 1.0 of the Apple Pascal system, regular units may NOT use intrinsic units.

Regular units may be installed in the SYSTEM.LIBRARY, or in any other library file. If installed in an alternate library, the uses statement should include the compiler option \$U <library name> before the unit name.

INTRINSIC UNITS

An intrinsic unit is pre-linked; that is, it contains sufficient information to allow the host program to use it without invoking the Linker. The code for an intrinsic unit remains in the SYSTEM.LIBRARY, and is loaded into memory from the library before the host program begins its actual execution. This keeps the size of the host program down; it also allows the unit and host program to be modified and recompiled individually without the need to link.

(Continued)

Intrinsic units must be installed in the SYSTEM.LIBRARY. An intrinsic unit can USE another intrinsic unit, but may NOT use a regular unit.

RESTRICTION: Intrinsic units may not have references to files (such as data files) in Pascal version 1.0.

ASSEMBLY ROUTINES AS PART OF UNITS

Assembly language routines may be placed into library units. With intrinsic units, the unit is compiled, the machine language routines assembled, then the assembled code is linked to the unit PRIOR to installing the unit into the SYSTEM.LIBRARY.

Regular units may also contain machine language routines; however, these routines are NOT linked to the unit before it is installed in the library. Instead, the host program, the unit, and the assembly routines are linked together at the same time.

ADDITIONAL NOTES ON THE CONSTRUCTION OF A UNIT

Any unit which does not contain at least one procedure in the INTERFACE section is not allowed to have an IMPLEMENTATION section. You must include the initialization section (BEGIN..END.), however.

Procedures listed in the INTERFACE section are public to the host program as well as to the unit. Those procedures listed only in the IMPLEMENTATION section cannot be accessed by the calling program (private). If no procedures are listed publicly, then none can be called from the host, and the IMPLEMENTATION section is not allowed.

Any intrinsic unit which contains a global variable, either public (defined in the INTERFACE) or private (defined in the IMPLEMENTATION), MUST have a data segment. Lack of a needed data segment will result in a system error.

(Continued)

GENERAL FORMAT OF UNITS

The following example is designed to illustrate the general structure of a unit. The line numbers at the left of the page are for reference, and are not part of the actual structure. Discussion of the format will refer to the line numbers.

```
1:  (*$S+*)
2:  UNIT < name >; INTRINSIC CODE xx DATA yy; (* DATA yy is used
      only if a data segment is required. *)

3:  INTERFACE

4:  USES < name of unit to be used >;
      CONST < definitions >;
      TYPE < definitions >;

5:  VAR < definitions >;

6:  PROCEDURE ONE (I:Integer);
      PROCEDURE TWO (I:Integer); (* External procedure *)
      FUNCTION THREE (I:Integer) : Integer;
      FUNCTION FOUR (I:Integer) : Integer; (* External function *)

7:  IMPLEMENTATION

8:  CONST < definitions >;
      TYPE < definitions >;

9:  VAR < definitions >;

10: PROCEDURE ONE;
      BEGIN
          END;

      PROCEDURE TWO; EXTERNAL;

      FUNCTION THREE;
          BEGIN
              END;

      FUNCTION FOUR; EXTERNAL;

11: BEGIN
      (* initialization section *)
      END.
```

(Continued)

DISCUSSION

1: The swapping option is required when compiling ANY unit, regardless of its size. This is the most common cause of compiler failures when working with units. The option should be the first line of text, preceding the UNIT header and any other compiler option.

2: The word INTRINSIC is required if the unit is to be intrinsic. DATA is optional, used only if a data segment is necessary (see #5 below). Regular units use only the UNIT < name > portion of this line.

3: INTERFACE is required. It defines the start of this section, which must contain some entries (a totally empty INTERFACE section is not allowed). This section contains the information which is public to the host program as well as the unit (and visible from the LIBMAP program).

4: These entries are optional. If used, they are public (see #3). If a nested unit is used, it MUST be declared at this point, and the host program must also declare it in its USES statement before declaring this unit (see pages 75-81 in the Pascal Language Reference Manual).

5: VARs are also optional. If VARs are used in an intrinsic unit, a DATA segment MUST be declared, since these are global variables.

6: These are the public statements of the unit's procedures and functions. All parameters MUST be declared in this section, and must not appear in the IMPLEMENTATION. EXTERNAL may not be specified at this point (see #10).

7: IMPLEMENTATION is required. This section contains the code generating statements, and any private types or variables to be declared. If an item is declared in this section only, it is local to the unit and cannot be accessed from the host program directly, and is not visible to LIBMAP.

8: CONST and TYPE definitions are optional. If included at this point, they are private and will not be visible to the calling program.

9: VARs are optional, and are private if defined at this point. A DATA segment is required if global VARs are specified in the IMPLEMENTATION section of an intrinsic unit, even though they are private.

10: These are the actual code generating procedures. Procedures and functions already declared in the INTERFACE section may not have parameters listed here, as this would be a duplication. EXTERNAL references are specified at this time.

11: This is the initialization section. Code in this portion is optional, but the BEGIN and END statements must be present in any event. The final END statement must be followed by a "." to indicate the end of the text. Code placed in the initialization section will be executed immediately upon access to the unit (through the USES statement in the host program), and

(Continued)

Issued 21 May 82

Pascal II

Page 6100.037.05

ignored thereafter. An example of this is TURTLEGRAPHICS, where the initialization code is responsible for allocating the hi-res page so that variables will not be lost.

Issued 19 Feb 82

Pascal

Page 6100.038.01

BIBLIOGRAPHY

Low Resolution Graphics in Pascal, APPLE ORCHARD, Spring 1981

Modifying Pascal BIOS to Work with the Paymar Lower Case Adapter
APPLE ORCHARD, Spring 1981

Issued 21 May 82

Pascal II

Page 6100.039.01

RESIDENT COMPILER OPTION

The Linker incorrectly resolves Regular Units resident in other Regular Units. The Compiler assigns temporary segment numbers to Regular Units at compile time. These temporary segment numbers are used by the Linker to resolve the resident command, with varying results (always incorrect).

The suggested work-around is to specify the segment number as resident rather than the segment (unit) name. For example {\$R 8}.

Issued 25 May 82

Pascal

Page 6100.990.01

ERRATA - APPLE PASCAL OPERATING SYSTEM REFERENCE MANUAL

A2L0028

030-0100-00

Page 105

The first paragraph is a duplicate of the last paragraph on page 104.

Page 107

Maximum Editor file size should be 38 blocks.

Page 130

SYSTEM.SWAPDISK is misspelled.

Page 140

The example for replace should read: /R/<CTRL-L>//<RET>/.

Page 150

".CODE" does not have to be specified for Output file from Linker.

Page 199

The HAZEL.MISCINFO file found on Apple 3: is designed for the Hazeltine 1510 terminal, and will not work completely on the 1500 as set up.

Page 203

"HAS RANDOM CURSOR ADDRESSING" refers the reader to Section 4.7. This information is on pages 210 through 213.

Page 286

Line 9 states that RESET does a warm boot. This is incorrect, RESET does a cold boot in Pascal Version 1.1.

Issued 22 Oct 81

Pascal

Page 6100.991.01

ERRATA - APPLE PASCAL LANGUAGE REFERENCE MANUAL
A2L0027
030-0101-00

Page 22

There is no mention of the CHR (X) function.

Page 86

ORD will accept any ordinal type (integer, character, or user defined type) and return the ordinate of the argument. We should mention that the most common use for ORD is to return the ASCII value of a character.

Page 148

The two step boot cannot use APPLE1: for the first step because RESET does a cold boot in Version 1.1. Using APPLE3: is OK for the first part of the boot.

 Issued 1 Jun 82

Pascal ///

Page 6150.000.01

INDEX TO PASCAL ///

ALPHABETIC LISTING

Doc	Title
001	Changing Text Modes
003	Comparison to Apple][Pascal
002	Filer - Errors Listing a Directory to the .Silentype Driver

NUMERIC LISTING

Doc	Issue Date	Previous Date	Pages
000	JUN 1 82	MAY 19 82	1
001	MAY 19 82	-	1
002	MAY 19 82	-	1
003	MAY 21 82	-	4

CHANGING TEXT MODES

The WRITE and WRITELN procedures can't be used to send the necessary control sequences to change the text mode to 40 columns black and white, for instance. You must use the UNITWRITE procedure instead. The correct syntax is:

```
PROGRAM DEMO;
VAR ANS :STRING;

PROCEDURE CONTROL (NUMBER:INTEGER);
VAR TRYIT :CHAR;
BEGIN
    TRYIT := CHR (NUMBER);
    UNITWRITE (1, TRYIT, 1, , 12);
END;

BEGIN
    CONTROL (16);
    CONTROL (0);
    READLN (ANS);
END.
```

Issued 18 May 82

Pascal ///

Page 6150.002.01

FILER - ERRORS LISTING A DIRECTORY TO THE .SILENTYPE DRIVER

The original version of the driver for the Silentype printer does not work properly with the Profile driver. It is easy to tell what version of the Silentype driver you have using the System Utilities diskette. Use the System Configuration Program to edit the Silentype driver. The last item is the Version ID number. This number should be 1.04 or greater. Version 1.04 is available on the System Utilities Data diskette that comes with the System Software Package.

COMPARISON TO APPLE][PASCAL

The great majority of Apple][Pascal programs can be recompiled and executed on the Apple /// without modification.

The following summary of significant differences between Apple][Pascal and Apple /// Pascal is presented in the Apple /// Pascal Programmer's Manual, Volume 2 starting on page 128.

OTHERWISE CLAUSE IN CASE STATEMENT

Apple /// Pascal provides an OTHERWISE clause in the CASE statement. The OTHERWISE clause, if present, contains a statement that is executed if none of the cases in the CASE statement are executed. See Chapter 5 of the Apple /// Pascal Programmer's Manual.

SOS PATHNAMES

SOS Pathnames are different from the Pascal filenames used on the Apple][. Apple /// Pascal supports both kinds of names, as explained in the Apple /// Pascal Introduction, Filer, and Editor manual.

SOS DEVICE DRIVER SUPPORT

All SOS device drivers are supported by Apple /// Pascal as "I/O units." See Chapters 10-12 and the Standard Device Drivers Handbook.

GRAPHICS

The Apple /// screen graphics modes differ significantly from the Apple][, and the graphics screen is driven through the SOS graphics driver (see Standard Device Drivers Handbook). Therefore, a new unit named PGRAF is supplied as a high-level interface to the graphics driver.

TURTLEGRAPHICS is available only for compatibility with Apple][- see Appendix K of the Apple /// Pascal Programmer's Manual.

NEW PROCEDURES

DATE, TIMEOFDAY, CLOCKINFO, and SETTIME are procedures provided in the APPLESTUFF unit for reading and setting the Apple /// system's internal date and time. See Appendix D of the Apple /// Pascal Programmer's Manual.

(Continued)

NEW DATA TYPES

The BYTESTREAM and WORDSTREAM types are provided for use as types of VAR parameters in procedure and function definitions. See Chapter 3 of the Apple /// Pascal Programmer's Manual.

REAL ARITHMETIC

For operations on values of type REAL, Apple /// Pascal conforms to the proposed IEEE floating-point standard. Under Default conditions, the difference between this and the arithmetic of Apple][Pascal is invisible unless the program performs operations with exceptional results (such as division by zero). See Appendix F of the Apple /// Pascal Programmer's Manual for complete details.

LIBRARY FILES AND UNITS

In addition to the SYSTEM.LIBRARY file, each codefile under Apple /// Pascal can have a "program library" file associated with it. This makes the use of library units more convenient and allows a program to have up to 48 segments at run time.

When compiling a unit, it is no longer necessary in all cases to use the Compiler's swapping option.

MEMORY ORGANIZATION

The memory organization of the Apple /// under SOS and Pascal is different from the memory organization of the Apple][under Pascal. This makes no difference to most programs. However, the amount of memory available for a user program will be somewhat greater on the 128K Apple /// than on the Apple][.

Memory organization might adversely affect an Apple][program if it depends on pointer values created while running on an Apple][and stored in a diskette file. Any such values will of course be incorrect on the Apple ///.

Similarly, an Apple][program that depends on explicit Apple][hardware addresses will not work on the Apple ///. This might affect Apple][Pascal programs that are designed to drive the Silentye printer; such programs should be revised to use the Apple /// Silentye driver described in the Standard Device Drivers Handbook.

(Continued)

THE UNITSTATUS PROCEDURE

For device oriented I/O, the UNITSTATUS procedure is supported. See Chapter 12 of the Apple /// Pascal Programmer's Manual.

RUNTIME SEGMENT TABLE

The runtime segment table allows for 64 segments instead of 32. See Chapter 15 of the Apple /// Pascal Programmer's Manual.

CONDITIONAL COMPILATION

The Apple /// Pascal Compiler allows conditional compilation. See Appendix F of the Apple /// Pascal Programmer's Manual.

THE CHAINSTUFF UNIT

Since Apple /// Pascal has no "system swapping" mode, the SWAPON and SWAPOFF procedures are absent from the CHAINSTUFF unit.

COMPILING APPLE][CODE

The Apple /// Pascal Compiler can compile code to run on the Apple][. See Appendix F of the Apple /// Pascal Programmer's Manual.

FILE VARIABLE SIZE

Every declared file in an active procedure requires 1,100 bytes of memory.

COMPILER OPTIONS

Option names can be spelled out.

Because Compiler options always end with a comma, they can be chained together (except for the INCLUDE option). Therefore, the COMMENT option cannot contain a comma, and the RESIDENT option does not accept a list.

PROCEDURE COMPLEXITY

A more complex procedure may be compiled in the Apple /// than on the Apple][because of the Apple ///'s larger memory.

(Continued)

SYSTEM GLOBALS

Users of the {USER-} option may find that their programs are not portable.

 Issued 12 May 82

Parallel Printer Interface

Page 6200.000.01

INDEX TO PARALLEL PRINTER INTERFACE

ALPHABETIC LISTING

Doc	Title
002	Controlling the Most Significant Bit
990	Errata - Parallel Printer Interface Manual

NUMERIC LISTING

Doc	Issue Date	Previous Date	Pages
000	MAY 12 82	OCT 28 81	1
002	DEC 14 81	-	3
990	OCT 28 81	-	2

CONTROLLING THE MOST SIGNIFICANT BIT

INTRODUCTION:

This driver allows the user control bit 8 from an Apple Parallel Interface with a POKE. Bit 8 is used by some printers to select expanded or normal print mode or to enable alternate or graphics character sets. DOS 3.2 or DOS 3.3 is required to use this routine. It will work in the Apple /// in Emulation mode.

SOFTWARE ENTRY

First you must decide which slot the interface will go in and enter the driver. The routine is customized for this slot number and won't work properly if used with a different configuration. Enter the driver using the values from the table for words in brackets, < >.

SLOT	1	2	3	4	5	6	7
CODE	C1	C2	C3	C4	C5	C6	C7

Enter the monitor with CALL -155 and type

```

3B0:A9 <SLOT>
:20 95 FE
:A9 80
:20 ED FD
:A9 C5
:85 36
:A9 03
:85 37
:4C EA 03
:29 7F
:0D CD 03
:4C 02 <CODE>
:80
    
```

(Continued)

To check your typing, type

3B0L

and compare your listing to the one below for an interface plugged into slot 1.

```

03B0-  A9 01      LDA  #01
03B2-  20 95 FE   JSR  $FE95
03B5-  A9 80      LDA  #80
03B7-  20 ED FD   JSR  $FDED
03BA-  A9 C5      LDA  #C5
03BC-  85 36      STA  $36
03BE-  A9 03      LDA  #03
03C0-  85 37      STA  $37
03C2-  4C EA 03   JMP  $03EA
03C5-  29 7F      AND  #7F
03C7-  0D CD 03   ORA  $03CD
03CA-  4C 02 C1   JMP  $C102
03CD-  80        ???
    
```

Now return to BASIC with 3D0G

SAVING THE PROGRAM TO DISK:

The driver should be in memory before the printer is used. Save the driver by typing

BSAVE CEN 730, A\$3B0, L\$1E

(Continued)

USING THE PRINTER:

The first time you want to use the printer you must load the driver and initialize the interface. From command mode type

```
BLOAD CEN 730  
CALL 944
```

This may be done from a program by entering

```
100 PRINT D$;"BLOAD CEN 730" : CALL 944
```

assuming that D\$ is a control-D.

If you want to switch back to the video monitor for output type

```
PR#0
```

or in a program enter

```
200 PRINT D$;"PR#0"
```

Then to reconnect the printer, all that is required is

```
CALL 954
```

or from a program

```
300 CALL 954
```

SETTING THE PRINT MODES:

To set normal print mode POKE 973,0

To set expanded print mode POKE 973,128

Issued 28 Oct 81

Parallel Printer Interface

Page 6200.990.01
-----ERRATA - PARALLEL PRINTER INTERFACE MANUAL
030-0005-01

Page 6

Throughout the Parallel Interface Manual, a negative going acknowledge is shown as ACK (with a bar over it) and a positive going acknowledge as ACK (without the bar). Therefore, when choosing a jumper block for printers that have a negative going strobe and negative going acknowledge signal find the jumper block on page 6 that is labeled (bar STR)-(ACK).

Page 17

The command "Ctrl-I I RETURN" has no effect with the Centronics version of this card. Use "Ctrl-I O RETURN" instead.

Page 17

The command "Ctrl-I K RETURN" has no effect with the Centronics version of this card because it never generates a linefeed.

Page 18

Notes: For users of Applesoft BASIC... Applesoft allows the "PR#" command. The rest of the note is still valid but of limited utility.

Page 18

Using Printer Commands in BASIC Programs:

>10 PR#1	Turns off Printer Card.	should read
>10 PR#1	Turns on Printer Card.	

Page 18

For the Centronics version of the card, Using Printer Commands in BASIC Programs:

>30 PRINT "^I I"	should be
>30 PRINT "^I O"	
>40 PRINT "^I K"	does not turn off the Line Feed as there never was one.

(Continued)

Page 19

For the Centronics version of the card, the section "Example Of Control From a BASIC Program" needs updating:

```
(20 PRINT "^IK");      is never needed or valid
>50 PRINT "^II";       should be replaced with
>50 PRINT "^IO";
```

Page 19

For the Centronics version of the card, Listing Programs Containing Print Commands

```
>^IK is never needed or valid
```

Page 21

The address to POKE from Basic is really

```
POKE (-16256 + N * 16)
```

 Issued 12 May 82

Pilot

Page 6400.000.01

INDEX TO PILOT

ALPHABETIC LISTING

Doc	Title
002	Controlling Memory Mapped I/O
003	How to Read Apple Pilot Data Files from Pascal
001	Linking from Pilot to Pascal

NUMERIC LISTING

Doc	Issue Date	Previous Date	Pages
000	MAY 12 82	OCT 1 81	1
001	SEP 28 81	-	2
002	SEP 28 81	-	1
003	SEP 28 81	-	2

Issued 28 Sep 81

Pilot

Page 6400.001.01

LINKING FROM PILOT TO PASCAL

The LP:filename command will leave the PILOT interpreter, and execute the Pascal program in the file "filename". If you are in a PILOT lesson, and wish to use a Pascal program which is located in the file MYPROG.CODE, insert the following command in your lesson:

LP:MYPROG

The LP stands for Link to Pascal. Note that you do not have to put the ".CODE" suffix on the file name; PILOT will automatically append it.

IMPORTANT: You MUST use Pascal version 1.0 to compile your Pascal program. If you use any UNITS from the SYSTEM.LIBRARY, you must compile using the SYSTEM.LIBRARY file provided on the PILOT LESSON1: student diskette. The (*\$U name *) compiler option will allow you to do this.

When the LP: command is performed, control leaves the PILOT interpreter. Any open PILOT data files are closed when this happens, and the status of PILOT variables and your location within the PILOT lesson will no longer remain in memory.

When the Pascal program ends, the system will reboot, and you will re-enter your PILOT "HELLO" lesson. Thus, if you wish to return to a particular point in a PILOT lesson, you must have your PILOT program "leave a trail" for PILOT to pick up.

As an example, let's say you want PILOT to return to lesson MECHANICS, at the label BOLTS. Your lesson might do this:

```
D:N$(50)
T:Going off to Pascal...

R:Open file "restart"
C:N$="RESTART"
FOX:1,N$

R:Leave a trail in the form of the
R:appropriate L: command
C:N$="L:MECHANICS,BOLTS"
FO:1,N$

R Go off to Pascal file MYPROG.CODE
LP:MYPROG
```

(Continued)

The HELLO lesson on the PILOT disk will look for the RESTART file, as follows:

D:N\$(50)

C:N\$="RESTART"

FIX:1,N\$

R: If error flag is raised, the RESTART file

R: does not exist, so link to the normal first lesson

LE:FIRSTLESSN

R: Otherwise, read in the record containing our

R: restart lesson and label, and link there.

FI:1,N\$

XI:N\$

CONTROLLING MEMORY MAPPED I/O

Many Apple II compatible I/O devices are memory mapped, and can be controlled directly from Apple PILOT. The PILOT system variable %M is actually a memory map, which is controlled by a subscript. The value of the subscript defines the location in the I/O device control space which is addressed by %M.

The I/O locations begin with address C080 (hex). This constant offset is added automatically to all %M locations. %M(16) thru %M(127) will address the device control space of slots 1 thru 7 (access to slot 0 is not permitted). %M(128) thru %M(1919) refer to the I/O select space of slots 1-7. The I/O select space is typically used for ROM memory, but can be used in some cases for auxiliary device control information.

For example, the PILOT command

```
C: %M(16) = 0
```

will "poke" a zero into I/O location 16, which is the first available byte (relative location zero) in the device select space of slot 1. Similarly, the command

```
C: X = %M(20)
```

will "peek" location 20 (relative location 4 in slot 1), and return the result in variable X.

%M may be used generally wherever the system variable %A is legal (for example). It cannot be used to manipulate program memory from within PILOT lessons, but does provide some measure of external device control.

CAUTION: The %M variable is a very powerful construct. It is assumed that the author takes full responsibility for its use. Some I/O devices can be damaged by indiscriminate "peeking" and "poking". In particular, DO NOT try to manipulate any of the device controls for slot 6 (the disk controller). It is entirely possible to erase entire tracks on one or both of your PILOT disks (Author and Lesson), and render them permanently unuseable.

HOW TO READ APPLE PILOT DATA FILES FROM PASCAL

From the Apple PILOT user's viewpoint, data files appear to be text files (i.e. files of strings), which are also randomly accessible. The file is actually block structured, however, with two strings of length 255 per block. Records 0 and 1 are contained in relative block zero, 2 and 3 are in block 1, etc. The Pascal construct which defines this file format is:

```
VAR PILOTFILE: File of String [255];
```

The data record associated with this file will have the description PILOTFILE^, and will appear to be a string of length 255. The file may be sequentially or randomly accessed using the commands GET, PUT and SEEK (see the Pascal Language Reference manual for details).

To open an existing file, use the RESET command. A new data file may be created using REWRITE (these are the same as the PILOT commands FIX: and FOX:, respectively). Always CLOSE a new file using the LOCK option, or the data file will not be retained permanently in the diskette directory.

The PILOT command LP: PASCALPGM will Link to the Pascal program contained in the file PASCALPGM.CODE. If this program requires any library intrinsic units, it MUST be compiled using the library found on the Apple PILOT Author Diskette (or Lesson Diskette), and using the Pascal 1.0 compiler. This is because the Apple PILOT system uses a special 48K run-only version of Pascal 1.0, which requires different intrinsics due to its unique storage allocation.

The linkage from PILOT to Pascal is one-way only. Data cannot be transmitted from PILOT to Pascal, except by data files (on diskette). When the Pascal program ends, the lesson diskette will re-boot automatically and begin execution of the HELLO program (if any). Please note also that programs running in this environment have approximately 10K less user storage than in the standard Pascal system.

The first compiled procedure in the outer block of any Pascal program running under the Apple PILOT 48K system should be

```
PROCEDURE SYSERROR;
BEGIN
END;
```

System errors will automatically divert control to this procedure, with IORESULT set to the corresponding error number. The default action is: (1) if the error is recoverable, abort execution of the procedure currently being executed, and resume execution of its calling procedure, or (2) if it is not recoverable, terminate the program with an appropriate error message. If you desire any additional error checking or recovery, it should be done in this procedure.

(Continued)

As an example, here is a listing of a Pascal program which will print the contents of any Apple PILOT data file:

```
PROGRAM DATALIST;

VAR PILOTFILE: File of String [255];
    FILENAME: String;
    RECNO: Integer;

PROCEDURE SYSERROR;
BEGIN
END;

PROCEDURE PAUSE;
BEGIN
    WRITELN;
    WRITE (^Press RETURN to continue...^);
    READLN;
    WRITELN
END;

BEGIN
    PAGE (OUTPUT);
    WRITE (^List which data file? ^);
    READLN (FILENAME);
    WRITELN;
    FILENAME := CONCAT (FILENAME,^.DATA^);
    RESET (PILOTFILE, FILENAME);
    RECNO := 0;
    WHILE NOT EOF (PILOTFILE) DO BEGIN
        WRITELN (^Record ^,RECNO,^: ^,PILOTFILE^);
        GET (PILOTFILE);
        RECNO := RECNO + 1;
        IF RECNO MOD 20 = 0 THEN PAUSE
    END;
    WRITELN;
    WRITELN (^End of file: ^,RECNO,^ records listed.^);
    PAUSE
END.
```

 Issued 12 May 82

Plan80

Page 6600.000.01

INDEX TO PLAN80

ALPHABETIC LISTING

Doc	Title
001	Plan80 and the Apple ///

NUMERIC LISTING

Doc	Issue Date	Previous Date	Pages
000	MAY 12 82	OCT 29 81	1
001	OCT 29 81	-	1

Issued 29 Oct 81

Plan80

Page 6600.001.01

PLAN80 AND THE APPLE ///

The Plan80 package won't work on the Apple /// in emulation mode because it is written in Pascal and there is no Language Card in the Apple ///.

 Issued 1 Jun 82

Profile

Page 6650.000.01

INDEX TO PROFILE

ALPHABETIC LISTING

Doc	Title
001	Interaction with the Apple /// Built-in Memory Test
002	Problem Saving Files from the Original Visicalc

NUMERIC LISTING

Doc	Issue Date	Previous Date	Pages
000	JUN 1 82	MAY 12 82	1
001	MAY 21 82	-	1
002	DEC 14 81	-	1

Issued 21 May 82

Profile

Page 6650.001.01

INTERACTION WITH THE APPLE /// BUILT-IN MEMORY TEST

The built-in memory test in the Apple /// that starts at \$F6E6 will not work properly with the Profile connected or even with the Profile interface card installed. With everything connected it will successfully complete the first pass and then start beeping about once a second. With the interface card installed and the cable to the Profile disconnected it will loop through the memory test, possibly showing memory errors and printing "ROM ERROR" below the diagnostic display. Removing the interface card from the Apple /// will allow the memory test to run properly.

The Apple /// Confidence Disk will work properly!

Issued 14 Dec 81

Profile

Page 6650.002.01

PROBLEM SAVING FILES FROM THE ORIGINAL VISICALC

The original version (SOS dated 21 Nov 80) of Visicalc /// will return a DISK FULL error if you try to save a worksheet with a new pathname. It works OK with an existing pathname and the current version (SOS dated 4 Sep 81) will save to new files as well as existing files.

 Issued 12 May 82

Programmer's Aid #1

Page 6700.000.01

INDEX TO PROGRAMMER'S AID #1

ALPHABETIC LISTING

Doc	Title
003	Applesoft and the Programmer's Aid
002	Memory Test and DOS
001	Memory Test Bug

NUMERIC LISTING

Doc	Issue Date	Previous Date	Pages
000	MAY 12 82	SEP 28 81	1
001	SEP 21 81	-	1
002	SEP 21 81	-	1
003	SEP 21 81	-	1

Issued 21 Sep 81

Programmer's Aid #1

Page 6700.001.01

MEMORY TEST BUG

The memory test in the Programmer's Aid #1 has a problem. The routine that prints the error address is wrong in the ROM. According to the listing on page 86 of the manual, \$D670 contains 20 8A D6 when the ROM contains 4C CB 02. This causes the program to jump to \$02CB where there normally isn't any program. You can patch around this problem by entering the following instructions in the monitor before testing any memory:

```
02CB:20 8A D6 4C 73 D6
```

Issued 21 Sep 81

Programmer's Aid #1

Page 6700.002.01

MEMORY TEST AND DOS

Any attempt to test the memory used by DOS will cause the Apple to hang unless DOS is disabled first. That is because all input from the keyboard and output to the screen goes through DOS and DOS gets erased during the memory test. DOS can be disabled by typing FE89G FE93G from the monitor.

Issued 21 Sep 81

Programmer's Aid #1

Page 6700.003.01

APPLESOFT AND THE PROGRAMMER'S AID

The Programmer's Aid #1 uses some of the same addresses as the Applesoft ROMS. This makes it impossible to use the PA#1 while in Applesoft. The only way to have access to the memory test is to enter Integer Basic before going into the monitor. The sequence of commands would be:

```
INT
CALL -155
```

Issued 12 May 82

Psort

Page 6800.000.01

INDEX TO PSORT

ALPHABETIC LISTING

Doc	Title
001	Psort and the Apple ///

NUMERIC LISTING

Doc	Issue Date	Previous Date	Pages
000	MAY 12 82	OCT 29 81	1
001	OCT 29 81	-	1

Issued 29 Oct 81

Psort

Page 6800.001.01

PSORT AND THE APPLE ///

The Psort package won't work on the Apple /// in emulation mode because it is written in Pascal and there is no Language Card in the Apple ///.

Issued 12 May 82

The Shell Games

Page 7100.000.01

INDEX TO THE SHELL GAMES

ALPHABETIC LISTING

Doc	Title
990	Errata - The Shell Games

NUMERIC LISTING

Doc	Issue Date	Previous Date	Pages
000	MAY 12 82	SEP 28 81	1
990	APR 26 82	SEP 28 81	1

Issued 26 Apr 82

The Shell Games

Page 7100.990.01

ERRATA - THE SHELL GAMES 030-0066-00

Page 11

Item 4 mentions that the listing of "Animals and Their Young" is on page 9 when it's really on page 8.

Page 26

Paragraph 3 mentions that problem 5 resides from 9025 to 9029 when it is actually from 9030 to 9034.

Page 34

Modifying Professor True states that you are given 5 strings while you really only get 4, D1\$ to D4\$.

Page 35-36

You must save the program and re-run it before any flag changes will take effect.

 Issued 12 May 82

Silentype

Page 7200.000.01

INDEX TO SILENTYPE

ALPHABETIC LISTING

Doc	Title
003	Bibliography
990	Errata - Silentype Manual
002	Silentype and the TAB Command
001	Silentype Spacing Errors

NUMERIC LISTING

Doc	Issue Date	Previous Date	Pages
000	MAY 12 82	OCT 01 81	1
001	SEP 21 81	-	1
002	SEP 28 81	-	1
003	FEB 19 82	-	1
990	APR 28 82	SEP 29 81	1

Issued 21 Sep 81

Silentype

Page 7200.001.01

SILENTYPE SPACING ERRORS

If the RIGHT margin is set less than 40 and the screen is OFF, extra spaces may be introduced into the printout. Turn the screen on by sending a control "T" to the printer to correct this.

The Silentype margins must be reset after every PR#1. Some features, such as print intensity, do not need to be set every time and may lead to the belief that the system is supposed to work that way for everything.

Issued 28 Sep 81

Silentype

Page 7200.002.01

SILENTYPE AND THE TAB COMMAND

TAB does not work properly with this interface. The Integer Basic version is limited to 40 columns and an Applesoft TAB(20) will sometimes output 20 spaces instead of going to column 20. Use POKE 36,T instead of TAB (T) (where T is the tab value). This works with either Basic.

```
PRINT "FIRST ITEM";: POKE 36,55: PRINT "SECOND ITEM"
```

```
FIRST ITEM
```

```
SECOND ITEM
```

Issued 19 Feb 82

Silentype

Page 7200.003.01

BIBLIOGRAPHY

Double-Size Graphics for the Silentype, APPLE ORCHARD, Spring 1981

 Issued 28 Apr 82

Silentype

Page 7200.990.01

ERRATA - SILENTYPE MANUAL
 A2L0034
 030-0095-00

Page 11

This procedure for printing a Hi-Res image doesn't work. When you go to the Filer, Pascal clears the Hi-Res buffer. I suggest that the program be modified as shown below:

```

PROGRAM SPIRO;

USES TURTLEGRAPHICS, APPLESTUFF;
VAR ANGLE, DISTANCE : INTEGER;
    CH : CHAR;                                (* ADDED *)

BEGIN
  ANGLE := 0;
  WHILE NOT KEYPRESS DO
  BEGIN
    INITTURTLE;
    PENCOLOR (WHITE);
    FOR DISTANCE := 0 TO 99 DO
    BEGIN
      MOVE (2 * DISTANCE);
      TURN (ANGLE);
    END;
    ANGLE := ANGLE + 5;
  END;
  CH:=CHR (17);                                (* ADDED *)
  UNITWRITE (6,CH,1,0,12);                    (* ADDED *)
  TEXTMODE
END.
```

Page 13

The discussion on "ECHOING TO THE SCREEN" assumes that you have already done a PR#1 before you do the T command.

Page 18

The discussion in the last paragraph assumes that you have already done a PR#1 before you do the Q command.

Page 38

Line 4, "for the right margin is 2" should be "for the left margin is 2"

Page 39

Program line 6040 should end with THEN GOTO 6070

 Issued 1 Jun 82

Super Serial Card

Page 7300.000.01

INDEX TO SUPER SERIAL CARD

ALPHABETIC LISTING

Doc	Title
990	Errata - Super Serial Card Manual

NUMERIC LISTING

Doc	Issue Date	Previous Date	Pages
000	JUN 1 82	-	1
990	MAY 21 82	-	1

Issued 21 May 82

Super Serial Card

Page 7300.990.01

ERRATA - SUPER SERIAL CARD MANUAL

Page 36

The last two lines of the paragraph titled "A Terminal Mode Example" should read:

"BASIC program to an Applesoft program, substitute CHR\$(4) for D\$ and CHR\$(1) for A\$, and leave out program lines 40 and 42."

Issued 24 May 82

Supermap

Page 7400.000.01

INDEX TO SUPERMAP

ALPHABETIC LISTING

Doc	Title
001	Program Errors

NUMERIC LISTING

Doc	Issue Date	Previous Date	Pages
000	MAY 24 82	-	1
001	MAY 24 82	-	1

Issued 24 May 82

Supermap

Page 7400.001.01

PROGRAM ERRORS

Both Cincinnati and Bismarck are misspelled. The program is being revised to reflect the correct spelling.

APPLE COMPUTER TECHNICAL NOTES

Issued 24 May 82

Tax Planner

Page 7500.000.01

INDEX TO TAX PLANNER

ALPHABETIC LISTING

Doc	Title
001	Double Linefeeds When Printing
990	Errata - Tax Planner Manual

NUMERIC LISTING

Doc	Issue Date	Previous Date	Pages
000	MAY 24 82	MAY 12 82	1
001	MAY 24 82	-	1
990	OCT 29 81	-	1

Issued 24 May 82

Tax Planner

Page 7500.001.01

DOUBLE LINEFEEDS WHEN PRINTING

Tax Planner will cause double linefeeds when interfaced to certain printers with the Apple Parallel Printer Interface. The reason is that Tax Planner is a Pascal program and Pascal sends out a linefeed after each carriage return. The only fix for this is to stop the printer from auto-linefeeding or change the P9-00 PROM on the interface card with a P1-02 PROM. Changing the PROM will allow normal operation under both DOS and Pascal.

Issued 29 Oct 81

Tax Planner

Page 7500.990.01

ERRATA - TAX PLANNER MANUAL
030-0171-00

Page 49

Item VI should read:

LET

Excess Itemized Deductions = class 1 deductions
-.6 * (Adj gross income - class 2 deductions)

Issued 1 Jun 82

Universal Parallel Card

Page 7650.000.01

INDEX TO UNIVERSAL PARALLEL CARD

ALPHABETIC LISTING

Doc	Title
001	Epson MX-80 Printer

NUMERIC LISTING

Doc	Issue Date	Previous Date	Pages
000	JUN 1 82	-	1
001	MAY 21 82	-	1

Issued 21 May 82 Universal Parallel Interface Card Page 7650.001.01

EPSON MX-80 PRINTER

Table 2-2 which was to appear on page 10 for the UPIC manual was left out. Refer to an identical table 4-2 on page 19.

The Epson printer will not work properly with an Apple][parallel interface cable. The cable specified in table 2-1 on page 8 must be wired.

The switch on the UPIC must be set to Auto.

The SOS driver configuration block must be set to the values for the Epson MX-80 printer in table 4-2 on page 19.

 Issued 12 May 82

Visicalc ///

Page 7800.000.01

INDEX TO VISICALC ///

ALPHABETIC LISTING

Doc	Title
001	Column Insertion Problem
002	Printing from Visicalc ///
003	Problem Saving Files to Profile

NUMERIC LISTING

Doc	Issue Date	Previous Date	Pages
000	MAY 12 82	SEP 28 81	1
001	SEP 18 81	-	1
002	SEP 18 81	-	1
003	DEC 14 81	-	1

Issued 18 Sep 81

Visicalc ///

Page 7800.001.01

COLUMN INSERTION PROBLEM

If column 254 has been used in a given session, Visicalc assumes that there is no more room and won't allow column insertions. To recover, save the file to disk and re-load it. This will reset Visicalc's pointer.

Issued 18 Sep 81

Visicalc ///

Page 7800.002.01

PRINTING FROM VISICALC ///

The driver file that comes on the Visicalc /// diskette is set up with the Silentye as the default printer and another driver called .QUME that will drive the RS-232 port. There are two ways to get output to the Qume. The first is to use the System Configuration Program to rename the existing .PRINTER, .SILENTYPE and rename .QUME to .PRINTER. The second way is to send the information to the Qume as if it were a file.

.PRINTER	.QUME
/P	/P
P	F
	.QUME

At this point you will be prompted for the lower right coordinate of the part of the worksheet that you want printed.

Issued 14 Dec 81

Visicalc ///

Page 7800.003.01

PROBLEM SAVING FILES TO PROFILE

The original version (SOS dated 21 Nov 80) of Visicalc /// will return a DISK FULL error if you try to save a worksheet with a new pathname on the Profile. It works OK with an existing pathname and the current version (SOS dated 4 Sep 81) will save to new files as well as existing files.

Issued 12 May 82

VT-100 Emulator

Page 8000.000.01

INDEX TO VT-100 EMULATOR

ALPHABETIC LISTING

Doc	Title
001	VT-100 Emulator and the Micromodem

NUMERIC LISTING

Doc	Issue Date	Previous Date	Pages
000	MAY 12 82	OCT 29 81	1
001	OCT 29 81	-	1

Issued 29 Oct 81

VT-100 Emulator

Page 8000.001.01

VT-100 EMULATOR AND MICROMODEM

The VT-100 Emulator will not work with the Hayes Microcomputer Micromodem. It requires an Apple Communications Interface in slot 2. No other configuration will work.

INDEX TO BULLETIN BOARD SYSTEMS

ALPHABETIC LISTING

Doc	Title
001	Apple Bulletin Board Systems
002	Community Bulletin Board Systems
003	FORUM-80 Systems
007	Miscellaneous Bulletin Board Systems
004	NET-WORKS Systems
005	People's Message Systems
006	Remote CP/M Systems

NUMERIC LISTING

Doc	Issue Date	Previous Date	Pages
000	MAY 12 82	OCT 29 81	1
001	OCT 29 81	-	3
002	SEP 21 81	-	1
003	OCT 29 81	-	1
004	OCT 29 81	-	1
005	OCT 29 81	-	1
006	OCT 29 81	-	1
007	OCT 29 81	-	2

APPLE COMPUTER TECHNICAL NOTES

 Issued 29 Oct 81

Bulletin Board Systems

Page 9300.001.01

APPLE BULLETIN BOARD SYSTEMS

The following is a list of personal bulletin board systems that was downloaded from the Source in June. This information is undoubtedly out of date, so you might want to look on the Source yourself for the current information. Our thanks to Peoples' Message System, Santee CA (714) 449-5689 for letting us include this list.

ABBS ABACUS II, Toledo OH.....	(419)	865	1594	
ABBS A.C.E.S., Ft. Lauderdale FL.....	(305)	524	2237	
ABBS ACG-NJ, NJ.....	(201)	753	1225	
ABBS Addison, Dallas TX.....	(214)	661	2969	*24
ABBS AFB, Bell Gardens CA.....	(213)	773	6591	
ABBS RAUG, Akron OH.....	(216)	867	7463	*24
ABBS Akron Digital Group, Akron OH.....	(216)	745	7855	*24
ABBS Ames IA.....	(515)	294	8204	
ABBS Apple Bits, Kansas City MO.....	(913)	341	3502	*24
ABBS Apple Crate I, Vancouver WA.....	(206)	524	0203	
ABBS Apple Crate II, Vancouver WA.....	(206)	244	5438	
ABBS Apple Bin, Tacoma WA.....	(206)	937	0444	
ABBS Apple Cider, Las Vegas NV.....	(702)	454	3417	
ABBS Apple Group N.J., Piscataway NJ.....	(201)	968	1074	
ABBS Apple-Med, Iowa City IA.....	(319)	353	6528	
ABBS Apple Orchard, Vacaville CA.....	(707)	448	9055	
ABBS Atlanta GA.....	(404)	939	8429	
ABBS Augusta GA.....	(404)	793	1045	
ABBS Baileys Computer Store, Augusta GA.....	(404)	790	8614	
ABBS Boston MA.....	(617)	354	4682	
ABBS Bowling Green OH.....	(419)	352	4477	
ABBS Byte Shop, Miami FL.....	(305)	486	2983	
ABBS CCNJ, Pompton Plains NJ.....	(201)	835	7228	
ABBS Cleveland OH.....	(216)	779	1338	
ABBS Compumart, Ottawa Canada.....	(613)	725	2243	
ABBS Computerland, Jackson MS.....	(601)	362	8755	
ABBS Computer Center, Birmingham AL.....	(205)	945	1489	
ABBS Computer Corner, Amarillo TX.....	(806)	355	5610	
ABBS Computer Forum, Sante Fe Springs CA.....	(714)	739	0711	
ABBS Computer Forum, Sante Fe Springs CA.....	(213)	921	2111	
ABBS Computer Lab, Memphis TN.....	(901)	761	4743	
ABBS Computer Room, Kalamazoo MI.....	(616)	382	0101	
ABBS Computer Store, Toledo OH.....	(419)	531	3845	
ABBS Computer World, Costa Mesa CA.....	(714)	751	1422	*24
ABBS Dayton OH.....	(513)	223	3672	
ABBS Denver CO.....	(303)	759	2625	
ABBS Desert Technology, Phoenix AZ.....	(602)	957	4428	*24
ABBS Detroit MI.....	(313)	477	4471	
ABBS Downers Grove IL.....	(312)	964	7768	
ABBS Dubuque IA.....	(319)	557	9618	
ABBS El Paso TX.....	(915)	533	6255	
ABBS El Paso TX.....	(915)	533	7039	

(Continued)

APPLE TECH NOTES

Copyright (C) 1981 by Apple Computer, Inc.

"AppleIACTNJuly82_9300-001-01" 211 KB 1962-10-11 dpi: 300h x 300v pix: 1966h x 3001v

APPLE COMPUTER TECHNICAL NOTES

Page 9300.001.02

Bulletin Board Systems

Issued 29 Oct 81

ABBS Eugene OR.....	(503)	485	4546	
ABBS Fort Lauderdale FL.....	(305)	486	2983	
ABBS Fort Walton Beach/Destin FL.....	(904)	243	1257	
ABBS Fremont CA -1.....	(415)	792	8406	
ABBS Fremont CA -2.....	(415)	794	9314	
ABBS Gamemaster, Chicago IL.....	(312)	475	4884	*24
ABBS GLTC.DV1, Hollywood FL.....	(305)	989	9647	
ABBS Hayward CA.....	(415)	881	5662	
ABBS Illini Microcomputer, Naperville IL.....	(312)	420	7995	
ABBS Inner Circle, Dallas TX.....	(214)	530	0858	*24
ABBS Ketchikan AK.....	(907)	225	6789	
ABBS Lafayette CA.....	(415)	284	9524	
ABBS Laguna Niguel CA.....	(714)	495	6458	
ABBS Lawton OK.....	(405)	353	2554	
ABBS Lincoln NE.....	(402)	423	8086	*24
ABBS Louisville KY.....	(502)	426	2975	*24
ABBS MC^3, Decatur IL.....	(217)	429	5505	
ABBS McGraw-Hill, New York City NY.....	(212)	997	2186	
ABBS Memphis TN.....	(901)	725	5691	
ABBS Michigan Apple-Fone, Southfield MI.....	(313)	357	1422	
ABBS Morris Microproducts, Inglewood CA.....	(213)	673	2206	
ABBS MSS, Washington DC.....	(703)	255	2192	
ABBS Munch, Stamford CT.....	(203)	356	1921	
ABBS NJAUC, Marlton NJ.....	(609)	983	5970	
ABBS Oak Brook Computer, Oak Brook IL.....	(312)	941	9009	
ABBS OKC, Oklahoma City OK.....	(405)	528	8009	
ABBS Pacific Palasades, Los Angeles CA.....	(213)	459	6400	
ABBS PCnet, San Francisco CA.....	(415)	863	4703	
ABBS Peoria IL.....	(309)	692	6502	
ABBS Phoenix AZ.....	(602)	898	0891	
ABBS Portland OR.....	(503)	641	8555	*24
ABBS Rainbow Computing, Los Angeles CA.....	(213)	349	5728	
ABBS Saddlebrook NJ.....	(201)	843	4563	
ABBS San Antonio TX.....	(512)	737	0214	*24
ABBS San Francisco CA.....	(415)	661	0705	
ABBS Seattle WA.....	(206)	248	2600	
ABBS Softronics, Los Angeles CA.....	(213)	278	3873	
ABBS Software Sorcery, McLean VA.....	(703)	255	2192	
ABBS South of Market, San Francisco CA.....	(415)	821	1714	-SO
ABBS Spokane WA.....	(509)	456	8900	
ABBS St. Louis MO.....	(314)	838	7784	*24
ABBS The Moon, Dallas TX.....	(214)	931	3437	*24
ABBS Torrance CA.....	(213)	316	5706	
ABBS Turnersville NJ.....	(609)	228	1149	
ABBS Vermont, Essex Junction VT.....	(802)	879	4981	*24
ABBS West Palm Beach FL.....	(305)	689	3234	
ABBS Winesap, Dallas TX.....	(214)	824	7455	*24
ABBS Youngs Elect Svc., College Station TX...	(713)	693	3462	*24
ABBS Zim Computers, Minneapolis MN.....	(612)	561	6311	

(Continued)

APPLE TECH NOTES

Copyright (C) 1981 by Apple Computer, Inc.

"AppleIACTNJuly82_9300-001-02" 213 KB 1962-10-11 dpi: 300h x 300v pix: 1965h x 2989v

Compliments of Peoples' Message System, Santee CA.
(714) 449-5689

*24 Denotes 24-hour operation
-S0 Sexually oriented messages

COMMUNITY BULLETIN BOARD SYSTEMS

The following is a list of personal bulletin board systems that was downloaded from the Source in June. This information is undoubtedly out of date, so you might want to look on the Source yourself for the current information. Our thanks to Peoples' Message System, Santee CA (714) 449-5689 for letting us include this list.

CBBS AMRAD, Washington DC.....	(703)	734	1387	*24
CBBS Atlanta GA.....	(404)	394	4220	*24
CBBS Baton Rouge LA.....	(504)	273	3116	*24
CBBS Bloomington IN.....	(812)	334	2522	
CBBS Cambridge MA.....	(617)	864	3819	*24
CBBS Chicago IL.....	#1 (312)	545	8086	*24
CBBS Columbus OH.....	(614)	274	5758	
CBBS Corpus Christi TX.....	(512)	855	1512	
CBBS Detroit MI.....	(313)	288	0335	*24
CBBS LICA LIMBS, Long Island NY.....	(516)	561	6590	*24
CBBS Long Island NY.....	(516)	334	3134	*24
CBBS NW, Portland OR.....	(503)	646	5510	*24
CBBS RAMS, Rochester NY.....	(716)	244	9531	
CBBS Proxima, Berkeley CA.....	(415)	357	1130	
CBBS Richfield MN.....	(612)	869	5780	
CBBS Sacramento CA.....	(916)	483	8718	*24
CBBS Santa Clara CA.....	(408)	241	1956	
CBBS Stamford CT.....	(203)	348	6353	*24
CBBS Waco TX.....	(817)	776	1375	

Compliments of Peoples' Message System, Santee CA.
 (714) 449-5689

*24 Denotes 24-hour operation
 #1 Denotes original system of that type

APPLE COMPUTER TECHNICAL NOTES

 Issued 29 Oct 81

Bulletin Board Systems

Page 9300.003.01

FORUM-80 BULLETIN BOARD SYSTEMS

The following is a list of personal bulletin board systems that was downloaded from the Source in June. This information is undoubtedly out of date, so you might want to look on the Source yourself for the current information. Our thanks to Peoples' Message System, Santee CA (714) 449-5689 for letting us include this list.

Family Historians Forum 80, Fairfax VA.....	(703)	978	7561	
FORUM-80 Augusta GA.....	(803)	279	5392	
FORUM-80 Chicago IL.....	(312)	782	8180	
FORUM-80 Cleveland OH.....	(216)	486	4176	
FORUM-80 Denver CO.....	(303)	771	3826	*24
FORUM-80 Denver CO.....	(303)	399	8858	
FORUM-80 Ft. Lauderdale FL.....	(305)	772	4444	*24
FORUM-80 Hull England.....	(011)	44	482	859169
FORUM-80 Kansas City MO.....	#1 (816)	861	7040	
FORUM-80 Las Vegas NV.....	(702)	362	3609	*24
FORUM-80 Leavenworth KS.....	(913)	651	3744	
FORUM-80 Loosdrecht Holland.....	(010)	31351	2633	
FORUM-80 Memphis TN.....	(901)	362	2222	*24
FORUM-80 Monmouth, Brielle NJ.....	(201)	528	6623	*24
FORUM-80 Montgomery AL.....	(205)	272	5069	
FORUM-80 Nashua NH.....	(603)	882	5041	
FORUM-80 Orange County, Anaheim CA.....	(714)	952	2110	
FORUM-80 Orlando FL.....	(305)	830	8194	
	(305)	862	6917	
FORUM-80 Pontiac MI.....	(313)	335	8456	
FORUM-80 Seattle WA.....	(206)	723	3282	
FORUM-80 Shreveport LA.....	(318)	631	7107	*24
FORUM-80 Tulsa OK.....	(918)	224	5347	
FORUM-80 Union NJ.....	(201)	688	7117	
FORUM-80 Westford MA.....	(617)	692	3973	
FORUM-80 Wichita KS.....	(316)	682	2113	*24
FORUM-80 Wichita Falls TX.....	(817)	855	3916	

Compliments of Peoples' Message System, Santee CA.
 (714) 449-5689

*24 Denotes 24-hour operation
 #1 Denotes original system of that type

NET-WORKS

The following is a list of personal bulletin board systems that was downloaded from the Source in June. This information is undoubtedly out of date, so you might want to look on the Source yourself for the current information. Our thanks to Peoples' Message System, Santee CA (714) 449-5689 for letting us include this list.

NET-WORKS St. Louis MO.....	(314)	781	1308	
NET-WORKS Apple Jacks, Fontana CA.....	(714)	823	1451	
NET-WORKS C.A.M.S., Decatur IL.....	(217)	429	5541	
NET-WORKS Chicago IL.....	(618)	877	8080	
NET-WORKS Chicago IL.....	(312)	289	1198	
NET-WORKS Clah, Arlington Heights IL.....	(312)	255	6489	
NET-WORKS Computer Emporium, Des Moines IA...	(515)	279	8863	
NET-WORKS Dallas TX.....	(214)	361	1386	*24
NET-WORKS Hoffman Estates IL.....	(312)	882	9237	
NET-WORKS Magnetic Fantasies, Los Angeles CA.	(213)	465	1431	
NET-WORKS Woodland Hills CA.....	(213)	346	1849	

Compliments of Peoples' Message System, Santee CA.
 (714) 449-5689

*24 Denotes 24-hour operation

PEOPLE'S MESSAGE SYSTEMS

The following is a list of personal bulletin board systems that was downloaded from the Source in June. This information is undoubtedly out of date, so you might want to look on the Source yourself for the current information. Our thanks to Peoples' Message System, Santee CA (714) 449-5689 for letting us include this list.

PMS - Cincinnati OH.....	(513)	671	2753	
PMS - **IF**, Anaheim CA.....	(714)	772	8868	*24
PMS - Ellicott City MD.....	(301)	465	3176	
PMS - Gulfcoast, Freeport TX.....	(713)	233	7943	*24
PMS - Indianapolis IN.....	(317)	862	6191	*24
PMS - Los Angeles CA.....	(213)	291	9314	*24
PMS - Minneapolis MN.....	(612)	929	6699	*24
PMS - Mission Valley, San Diego CA.....	(714)	295	8280	*24
PMS - NIAUG, Lake Forest IL.....	(312)	295	6926	*24
PMS - Palo Alto CA.....	(415)	493	7691	*24
PMS - Portola Valley CA.....	(415)	851	3453	*24
PMS - San Diego CA.....	(714)	582	9557	*24
PMS - Santa Cruz CA.....	(408)	476	6181	
PMS - Santee CA.....	#1 (714)	449	5689	*24
PMS - Shrewsbury NJ.....	(201)	747	6768	
PMS - Apple Guild, Weymouth MA.....	(617)	767	1303	*24

Compliments of Peoples' Message System, Santee CA.
 (714) 449-5689

*24 Denotes 24-hour operation
 #1 Denotes original system of that type

APPLE COMPUTER TECHNICAL NOTES

Issued 29 Oct 81

Bulletin Board Systems

Page 9300.006.01

REMOTE CP/M SYSTEMS

The following is a list of personal bulletin board systems that was downloaded from the Source in June. This information is undoubtedly out of date, so you might want to look on the Source yourself for the current information. Our thanks to Peoples' Message System, Santee CA (714) 449-5689 for letting us include this list.

RCP/M CBBS Columbus OH.....	(614)	268	2227	*24
RCP/M CBBS Pasadena CA.....	(213)	799	1632	*24
RCP/M CBBS Vancouver BC Canada.....	(604)	687	2640	*24
RCP/M CCCC Lake Forest IL.....	(312)	234	9257	
RCP/M CP/M Net Simi Valley CA.....	(805)	527	9321	
RCP/M Detroit MI.....	(313)	584	1044	-RB
RCP/M Keith Petersen, Royal Oak MI.....	(313)	588	7054	-RB
RCP/M Logan Square, Chicago IL.....	(312)	252	2136	
RCP/M MAUDE, Milwaukee WI.....	(414)	241	8364	*24
RCP/M MCBBS, MI.....<pass=Sorcerer>..	(313)	535	9186	-RB
RCP/M Mississauga Ontario.....	(416)	826	5394	
RCP/M RAPM Chicago IL.....	(312)	384	4762	*24
RCP/M RBBS Allentown PA.....	(215)	398	3937	*24
RCP/M RBBS Huntsville AL.....	(205)	895	6749	-RB
RCP/M RBBS Hyde Park IL.....	(312)	955	4493	
RCP/M RBBS Baltimore MD.....	(301)	655	0393	-RB
RCP/M RBBS Cranford NJ.....	(201)	272	1874	
RCP/M RBBS Larkspur CA.....	(415)	383	0473	-RB
RCP/M RBBS Long Island NY.....	(516)	698	8619	-RB
RCP/M RBBS Mill Valley CA.....	(415)	383	0473	
RCP/M RBBS P&S Computer Svc, NY.....	(914)	279	5693	-RB
RCP/M RBBS Prodigy Systems, Baltimore MD.....	(301)	337	8825	*24
RCP/M RBBS Rochester NY.....	(716)	334	4604	
RCP/M RBBS Westland MI.....	(313)	729	1905	-RB
RCP/M RBBS Yelm WA.....	(206)	458	3086	-RB
RCP/M Rick Martinek, Milwaukee WI.....	(414)	774	2683	-RB
RCP/M SJBBS Bearsville NY.....	(914)	679	6559	-RB
RCP/M SJBBS Johnson City NY.....	(607)	797	6416	
RCP/M Steve Koper MI.....	(313)	584	1044	-RB
RCP/M SuperBrain, Lexington MA.....	(617)	862	0781	
RCP/M TCBBS Dearborn MI.....	(313)	846	6127	*24
RCP/M Terry O'Brien, Vancouver BC.....	(604)	584	2543	
RCP/M TRS-80 Chicago IL.....	(312)	949	6189	

Compliments of Peoples' Message System, Santee CA.
(714) 449-5689

*24 Denotes 24-hour operation
-RB Denotes call, let ring once and call back

APPLE COMPUTER TECHNICAL NOTES

 Issued 29 Oct 81

Bulletin Board Systems

Page 9300.007.01

MISCELLANEOUS BULLETIN BOARD SYSTEMS

The following is a list of personal bulletin board systems that was downloaded from the Source in June. This information is undoubtedly out of date, so you might want to look on the Source yourself for the current information. Our thanks to Peoples' Message System, Santee CA (714) 449-5689 for letting us include this list.

8BBS Santa Clara CA.....	(408)	296	5799	*24
AABB CP/M, New York NY.....	(212)	787	5520	
ARBB Seattle WA.....	(206)	546	6239	
ARBIS-80 Akron OH.....	(216)	724	1963	*24
Aviators Bulletin Board, Sacramento CA.....	(916)	393	4459	
BBS-80, Cincinnati OH.....	(513)	244	2983	
BBS Pensacola FL.....	(904)	477	8783	
BILLBOARD 80, San Jose CA.....	(408)	263	0248	
Bronx BBS, New York NY.....	(212)	933	9459	
BTBBS Centereach NY.....	(516)	588	5836	
BULLET-80, Danbury CT.....	(203)	744	4644	
CMS-80 Fort Lauderdale FL.....	(305)	462	8677	
COMM-80 Orange County, Fullerton CA.....	(714)	526	3687	
Communique-80, Livingston NJ.....	(201)	992	4847	
Compusystems, Columbia SC.....	(803)	771	0922	
Computer Arts Message System, Newhall CA.....	(805)	255	6445	
Connection-80, Fremont CA.....	(415)	651	4147	
Download-80 Fraser MI.....	(313)	294	8248	
HARD-80 Bethel CT.....	(203)	743	9281	*24
HEX, Silver Spring MD.....	(301)	593	7033	
Kinky Kumputer, San Francisco CA.....	(415)	647	9524	-SO
Lehigh Press BB, Lehigh PA.....	#1 (215)	435	3388	
Market 80, Kansas City MO.....	(816)	931	9316	
Medical FORUM-80, Mt. Clemens MI.....	(313)	465	9531	
M.O.M., Manhattan New York NY.....	(212)	245	4363	
MSG-80 Everett WA.....	(206)	334	7394	
New England Comp. Soc., Maynard MA.....	(617)	897	0346	
Orange County Data Exchange, Garden Grove CA.....	(714)	537	7913	
PBBS Co-operative Comp Svc, Palatine IL.....	(312)	359	9450	*24
Personal Message System-80, Deerfield Bch FL.....	(305)	427	6300	*24
PHOTO-80, Haledon NJ.....	(201)	790	6795	

(Continued)

APPLE TECH NOTES

Copyright (C) 1981 by Apple Computer, Inc.

"AppleIACTNJuly82_9300-007-01" 176 KB 1962-10-11 dpi: 300h x 300v pix: 1954h x 3001v

APPLE COMPUTER TECHNICAL NOTES

Page 9300.007.02

Bulletin Board Systems

Issued 29 Oct 81

Pet Bulletin Board, Ypsilanti MI.....(313) 484 0732 *24
Potomac Micro Magic Inc., Falls Church VA....(703) 379 0303
Powercom 2.2, Largo FL.....(813) 577 3095 *24
PSBBS Washington DC.....(202) 337 4694
PSBBS Gaithersburg MD.....(301) 840 8588 *24
PSBBS Software Associates, Reston VA.....(703) 620 4990

Remote Northstar Atlanta GA.....#1 (404) 939 1520 *24
Remote Northstar Nasa, Greenbelt MD.....(301) 344 9156
Remote Northstar Santa Barbara CA.....(805) 964 4115
Remote Northstar Santa Barbara CA.....(805) 682 7878
Remote Northstar Virginia Beach VA.....(804) 340 5246

SEACOMM-80.....(206) 763 8879
SLAMS St. Louis MO.....(314) 839 4307
SLUMS St. Louis MO.....(314) 394 7233

Triple Cities Bulletin Board, Endicott NY....(607) 754 5571 *24

Westside Download, Detroit MI.....(313) 533 0254

XBBS Dayton, OH.....(513) 863 7681 *24

Compliments of Peoples[^] Message System, Santee CA.
(714) 449-5689

*24 Denotes 24-hour operation
#1 Denotes original system of that type
-SO Sexually oriented messages

APPLE COMPUTER TECHNICAL NOTES

Issued 24 May 82

Apple PROMs

Page 9600.000.01

INDEX TO APPLE PROMS

ALPHABETIC LISTING

Doc	Title
001	List of Apple PROMs

NUMERIC LISTING

Doc	Issue Date	Previous Date	Pages
000	MAY 24 82	MAY 12 82	1
001	MAY 24 82	SEP 18 82	1

LIST OF APPLE PROMS

The following list can help you identify which Apple PROM you have.

341-0001	E0	Integer BASIC
341-0002	E8	Integer BASIC
341-0003	F0	Integer BASIC
341-0004	F8	Old Monitor
341-0005	P1-02	General Printer
341-0006	P2	
341-0009	P5	13 Sector Disk
341-0010	P6	13 Sector Disk
341-0011	D0	Applesoft
341-0012	D8	Applesoft
341-0013	E0	Applesoft
341-0014	E8	Applesoft
341-0015	F0	Applesoft
341-0016	D0	Programmers Aid #1
341-0017	P7-04	Serial Card
341-0018	P8	Serial Card
341-0019	P9	Centronics Printer
341-0020	F8	Auto-Start ROM
341-0027	P5A	16 Sector Disk
341-0028	P6A	16 Sector Disk
341-0048	P8A-00	Serial Card Software Handshake

INDEX TO VENDOR LIST

ALPHABETIC LISTING

Doc	Title
002	Hardware Products
001	Introduction to Vendor List
004	Miscellaneous Products
003	Software Products
005	Printer Manufacturers
006	Vendors - Numeric
007	Vendors - A
008	Vendors - B
009	Vendors - C
010	Vendors - D
011	Vendors - E
012	Vendors - F
013	Vendors - G
014	Vendors - H
015	Vendors - I
016	Vendors - J
017	Vendors - K
018	Vendors - L
019	Vendors - M
020	Vendors - N
021	Vendors - O
022	Vendors - P
023	Vendors - Q
024	Vendors - R
025	Vendors - S
026	Vendors - T
027	Vendors - U
028	Vendors - V
029	Vendors - W
030	Vendors - X
031	Vendors - Y
032	Vendors - Z

NUMERIC LISTING

Doc	Issue Date	Previous Date	Pages
000	MAY 12 82	OCT 1 81	2
001	SEP 21 81	-	1
002	SEP 29 81	-	10
003	SEP 29 81	-	14
004	SEP 29 81	-	2
005	SEP 29 81	-	3

(Continued)

APPLE COMPUTER TECHNICAL NOTES

Page 9900.000.02

Vendor List

Issued 12 May 82

006	SEP 29 81	-	1
007	SEP 29 81	-	3
008	SEP 29 81	-	2
009	SEP 29 81	-	4
010	SEP 21 81	-	2
011	SEP 29 81	-	2
012	SEP 29 81	-	1
013	SEP 29 81	-	1
014	SEP 29 81	-	2
015	SEP 29 81	-	2
016	SEP 29 81	-	1
017	SEP 29 81	-	1
018	SEP 29 81	-	1
019	SEP 29 81	-	4
020	SEP 29 81	-	1
021	SEP 29 81	-	1
022	SEP 29 81	-	2
023	SEP 29 81	-	1
024	SEP 29 81	-	2
025	SEP 29 81	-	4
026	SEP 29 81	-	2
027	SEP 29 81	-	1
028	SEP 29 81	-	2
029	SEP 29 81	-	1
030	SEP 29 81	-	1
031	SEP 29 81	-	1
032	SEP 29 81	-	1

Issued 21 Sep 81

Vendor List

Page 9900.001.01

INTRODUCTION TO VENDOR LIST

The vendor list was compiled from the various hobbyist magazine ads. It is intended as an overall guide to what products are available from which vendors. More complete directories of software are available from:

Vital Information
350 Union Station
Kansas City, MO 64108
(913) 384-3860

WIDL Video
5245 W. Diversey Av
Chicago, IL 60639
(312) 622-9606

The Book Co.
14013 Old Harbor Lane, Suite 312
Marina Del Rey, CA 90291

Skarbeks Software Directory
11990 Dorsett Rd
Maryland Heights, MO 63043
(314) 567-3291

We are always interested in more information. If you have any products that need to be added to the list, please forward technical information to:

Apple Computer, Inc.
10260 Bandley Dr.
Cupertino, CA 95014

Attn. Technical Communications
Mail Stop 4B

Issued 29 Sep 81

Vendor List

Page 9900.002.01

HARDWARE PRODUCTS

16K RAM Expansion

Andromeda Inc.
Computer Stop
Computer Technology Associates
Microsoft Consumer Products
Prometheus Products
R.H. Electronics
Videx

32K RAM Expansion

GS Computer Enterprises

9 Track Tape Drives

Electrovalue Industrial

AC Line Filter/Isolator

Dymarc Industries
Electronic Specialists
MFJ Enterprises
Protean Enterprises
RKS Enterprises

AC Standby Power Supply

Welco Ind

Amateur Radio

Macrotronics

Arithmetic processor

California Computer Systems

Bar Code Reader

Advanced Business Technology

(Continued)

APPLE TECH NOTES

Copyright (C) 1981 by Apple Computer, Inc.

"Apple|ACTN|July82_9900-002-01" 65 KB 1962-10-11 dpi: 300h x 300v pix: 1971h x 2989v

Braille I/O

Telesensory Systems

Clock/Calendar

Applied Engineering
California Computer Systems
CompuTime
Intelligent Control Systems
Lax Computer Products
Mountain Computer
SciTronics Inc
Thunderware
Westside Electronics

Digitizer/Graphics Tablet

Houston Instrument
Versa Computing

Disk, Cartridge

Cameo Electronics
Lobo Drives

Disk, 5" Floppy

Lobo Drives
Micro-Sci

Disk, 8" Floppy

Lobo Drives
Matchless Systems
Programma International

Disk, Winchester

Corvus
Konan
Lobo Drives

(Continued)

Issued 29 Sep 81

Vendor List

Page 9900.002.03

Disk Controller/DOS

Sorrento Valley Associates

Expansion Chassis

Mountain Computer

Extender Card

John Bell Engineering
SSM Microcomputer Products

Fan

MR Engineering
R.H. Electronics

Game I/O Expansion

CJM Industries
Programma

Hobbyist

John Bell Engineering
MR Engineering
Passport Designs

Industrial, Optical Position Indicator

United Detector Technology

Interface, Analog Input

California Computer Systems
Computer Technology Associates
Interactive Microware
Interactive Structures
Tecmar

(Continued)

Interface, Analog Output

Interactive Structures
Tecmar

Interface, Analog I/O

John Bell Engineering

Interface, IEEE-488

California Computer Systems
SSM Microcomputer Products

Interface, Isolated Parallel Power

Interactive Structures

Interface, Parallel Digital I/O

California Computer Systems
Interactive Structures
John Bell Engineering
SSM Microcomputer Products

Interface, Parallel I/O & Serial Asynchronous RS-232

Prometheus Products
SSM Microcomputer Products

Interface, Parallel Printer

John Bell Engineering
Micro-Ware

Interface, ROM

California Computer Systems
Mountain Computer

(Continued)

Issued 29 Sep 81

Vendor List

Page 9900.002.05

Interface, Selectric/Typewriter

Escon Products
Iplex International
Rochester Data
Vertical Data Systems

Interface, Serial Asynchronous RS-232

John Bell Engineering
California Computer Systems
SSM Microcomputer Products

Interface, Dual Serial Asynchronous RS-232

Prometheus Products

Interface, Serial Synchronous RS-232

California Computer Systems

Joystick/Paddle

Stanley van Dusen
Micromate Electronics
Programma International

Keyboard

RCA

Keyboard Enhancer

Dockside Computing
Lazer Micro Systems
Videx

Keypad

Advanced Business Technology

(Continued)

APPLE TECH NOTES

Copyright (C) 1981 by Apple Computer, Inc.

"Apple|ACTN|July82_9900-002-05" 67 KB 1962-10-11 dpi: 300h x 300v pix: 1959h x 2989v

Light Pen

3-G Co.
Symtec

Lower Case Adapter

Dan Paymar
Lazer Micro Systems
M.D. Software
MUSE
Videx

Message System

Rank Electro-Media

Mobile Power Supply

CPU Inc

Modem, Acoustic Couplers

Micromate Electronics
Novation
Ohio Data Products
Omnitec Data
Prentice Corp
Racal-Vadic
U.S. Robotics

Modem, Direct Connect

Hayes Microcomputer Products
Micromate Electronics
Novation

Modem, Direct Couplers

Hayes Microcomputer Products
Livermore Data Systems
Micromate Electronics
Novation
Universal Data Systems

(Continued)

Issued 29 Sep 81

Vendor List

Page 9900.002.07

Multi-User System

Corvus Systems
Nestar Systems

Music Keyboard

Passport Designs
Syntauri Inc.

Music Synthesizer

ALF Products
Mountain Computer
Passport Designs

Optical Card Reader

Mountain Computer

Paper Tape Reader/Punch

GNT Automatic

Plotter

Houston Instruments
Mauro Engineering

Processor Card, 6809

Stellation Two

Processor Card, Z-80

Carl Dick, Distributor
Microsoft Consumer Products

Prototyping Card

California Computer Systems
SSM Microcomputing Products

(Continued)

APPLE TECH NOTES

Copyright (C) 1981 by Apple Computer, Inc.

"AppleACTNJuly82_9900-002-07" 63 KB 1962-10-11 dpi: 300h x 300v pix: 1960h x 2989v

RESET Key Guard

Compdent

RF Modulator

ATV Research
M&R Enterprises

Remote Home Control

John Bell Engineering
Intelligent Control Systems
Mountain Computer
SciTronics
Symtec
Thunderware
Trillium Group

ROM Programmer

MicroProducts Inc.
Mountain Computer

ROM Simulator

Lamar Instruments

Sound Effects

MR Engineering
Symtec

Speech Synthesis/Recognition

Mountain Computer
Neutronics R&D Ltd
Street Electronics
Voicetek
Votrax

(Continued)

Issued 29 Sep 81

Vendor List

Page 99000.002.09

Switch, RS-232

Cybertech

Timer

California Computer Systems

Turtles

Terrapin, Inc.

Video 80 Column Display

Bit-3 Computers
Computer Stop
M&R Enterprises
Spies Laboratories
Videx

Video Hi-Res Enhancer

Ray Dahlby Electronics
Symtec

Video Image Digitizing

Computer Station
Micro Works

Video Monitor, B/W

Amdek Corp
Micro Products Unltd.
SMD

Video Monitor, Color

Video Marketing

Video Monitor, RGB

Video Marketing

(Continued)

APPLE TECH NOTES

Copyright (C) 1981 by Apple Computer, Inc.

"Apple|ACTN|July82_9900-002-09" 59 KB 1962-10-11 dpi: 300h x 300v pix: 1948h x 2977v

Video Standardizer

Adwar Video
Symtec

Video Tape Recorder Controller

BCD Assoc.

Video Terminal

Hazeltine Corp.
Maryland Computer Services
SOROC Technology
RCA

Video Terminal with Speech Output

Maryland Computer Services

Voice Entry Devices

Heuristics
Voicetek

Voice Entry Terminal

Scott Instruments

Issued 29 Sep 81

Vendor List

Page 9900.003.01

SOFTWARE PRODUCTS

Amateur Radio

High Sierra Software

Applesoft Extensions

Hayden Book Co
Dakin5

Applewriter Extensions and Graphics Package

Data Transforms

Architectural

Londe, Parker, Michels

Assembler

Hayden Book Co

Assembler, Macro

Eastern House Software

Astronomy

Information Unlimited Software
Synergistic Software

Authoring Systems (Educational)

Educational Services of Alaska
Muse Software

Aviation

Illinois Computer Mart

(Continued)

APPLE TECH NOTES

Copyright (C) 1981 by Apple Computer, Inc.

"Apple|ACTN|July82_9900-003-01" 60 KB 1962-10-11 dpi: 300h x 300v pix: 1960h x 2983v

Business, Accounts Payable

Continental Software
Peachtree Software
Westware Software

Business, Accounts Receivable

BPI Systems
Continental Software
Peachtree Software
Small Business Computer Systems
Westware Software

Business, Depreciation

Aardvark Software
Moneydisk

Business, Estate Planning

Aardvark Software

Business, Financial Planning/Forecasting

Charles Mann & Assoc.
Continental Software
Dakin 5
Decision Master
Denver Software
Educational Programming Systems
ESP Computer Resources
Hayden Book Co.
DR Jarvis Computing
Powersoft
Spectrum Software

(Continued)

Issued 29 Sep 81

Vendor List

Page 9900.003.03

Business, General Ledger

BPI Systems
Compumax
Continental Software
Peachtree Software
Serendipity Systems
Small Business Computer Systems
Westware Software

Business, Inventory

BPI Systems
Peachtree Software
Serendipity Systems
Softech International Corporation
Software Technology for Computers
Westware Software

Business, Invoices

Micro Lab

Business, Job Cost System

BPI Systems

Business, Mailing List

Avant-Garde Creations
Computer Services Company
Continental Software
Hayden Book Co.
Peachtree Software
Powersoft
Software Technology for Computers
Synergistic Software

Business, Order Entry

Compumax

(Continued)

Business, Payroll

BPI Systems
Broderbund Software
Peachtree Software
Software Technology for Computers
Westware Software

Business, Planning Aids

Personal Software
Spectrum Software

Business, Purchasing

Serendipity Systems

Business, Tax Planning/Preparation

Aardvark Software
Howard Software Services

Check Register

Powersoft
Spectrum Software

Data Base, Telecommunications

CompuServe
Dow Jones Information Services

(Continued)

Database Manager

Decision Systems
ESP Computer Resources
Hayden Book Co.
High Technology
Highlands Computer Services
Information Unlimited Software
Microlab
Personal Software
Programma International
Rocky Mountain Software
Software Publishing Corp
Software Technology for Computers
Stoneware Microcomputer Products
United Software of America
Westware Software

Decision Aids

DecisionMaster

Design Aids, Circuit

Spectrum Software

Design Aids, Circuit Simulator

Spectrum Software

Design Aids, Logic

Spectrum Software

Design Aids, Logic Simulator

Spectrum Software

Disassembler

Hayden Book Co
Rak-Ware

(Continued)

DOS Command Editor

Beagle Bros. Micro Software

Educational Software

Avant-Garde Creations
BCD Associates
Compu TA
Computer Services Company
Computer Systems Design Group
Educational Corseware
Edu-Comp, Inc.
Edu-Ware Services
Hartley Software
High Sierra Software
Mastertype
Math City
Micro Learningware
Shafer Software
Stoneware

Electronic Design/Control

High Sierra Software
Hayden Book Co.
Howard S. Sams & Co., Inc.
Spectrum Software

Energy Management

Hayden Book Co

Flight Simulation

subLogic

(Continued)

Issued 29 Sep 81

Vendor List

Page 9900.003.07

Foreign Language Applications

Computer Translation, Inc.
Tercer Medio

Foreign Language Study

Computer Translation, Inc.
Powersoft
Synergistic Software

(Continued)

APPLE TECH NOTES

Copyright (C) 1981 by Apple Computer, Inc.

"Apple|ACTN|July82_9900-003-07" 40 KB 1962-10-11 dpi: 300h x 300v pix: 1971h x 3001v

Games / Entertainment

Adventure International
 Avant-Garde Creations
 Automated Simulations
 Beagle Bros. Micro Software
 Broderbund Software
 BudgeCo.
 Color Software
 Continental Software
 Creative Computing Software
 Crystal Computer
 Crystalware
 Dynacomp
 Edu-Ware
 Galaxy
 Hayden Book Co.
 High Sierra Software
 Highlands Computer Services
 Instant Software
 Interlude
 Micro Lab
 MicroWare
 Muse Software
 Mytopia Gameware Institute
 On-Line Systems
 Powersoft
 Programma
 Programs Unlimited
 Quality Software
 Rainbow Computing
 Sirius Software
 Spectrum Software
 Strategic Software
 Stoneware Microcomputer Products
 Strategic Simulations
 subLogic
 Synergistic Software
 Systems Design Software
 TSE-Hardside
 United Software of America
 Western MicroData Enterprises

(Continued)

Graphics

Business & Professional Software
Computer Station
Connecticut Information Systems
Datasoft
Data Transforms
Micro-Ware
Rocky Mountain Software
Sirius Software
SmartWare
Spectrum Software
Versa Computing
West Coast Consultants

Graphics, Drawing

Co-op Software
Micro Lab
Sirius Software

Graphics, Histograms

Hayden Book Co

Graphics, Plotting

Interactive Microware
Muse Software
Personal Software
Spectrum Software

Graphics, Printer Dump Software

Computer Station

Graphics, Shape Table Generator

Co-op Software

Handicapped Applications

Rocky Mountain Software

(Continued)

Home Applications, Budget

Continental Software
Soft Touch
Spectrum Software

Home Applications, Recipe

Soft Touch

Languages, Compiled Basic

Hayden Book Co.
Online Systems

Languages, Forth

Micro Motion
Softape

Languages, Pascal

Abacus Software

Languages, Z80 Cobol

Apple Computer/ Micro Focus
Microsoft

Languages, Z80 FORTRAN

Microsoft

Languages, Z80 Pascal

Prometheus Products

(Continued)

Mathematics/Numerical Analysis

Action-Research Northwest
Interactive Microware
Serendipity Systems
Spectrum Software
Systems Design Software

Medical/Dental

Charles Mann & Assoc.
CompuSoCo
Hayden Book Company
Health Data Products, Inc.
Med Logic Systems
Prosoft

Medical Billing

Charles Mann & Assoc.
Professional Medical Software

Monitor Program Extender

Image Computer Products

Multi-User/Multi-Tasking

Omega Research

Music

Computer Applications Tomorrow

Pascal Programs

Abacus Software
Advanced Business Technology
Arizona Computer Systems
Broderbund
Link Systems

(Continued)

Program Editor/Cross-Reference

Highlands Computer Services
Sensible Software

Real Estate

Continental Software
Howard Software Services

Scheduling/Appointments/Calendar

Organic Software
Spectrum Software
Stoneware Microcomputer Products

Scientific Applications

High Sierra Software

Spelling Correction, Pascal

Intelligent Computer Systems Corp

Statistics

Human Systems Dynamics
Rosen Grandon Assoc.
Serendipity Systems
Spectrum Software

Stock/Quotes

H & H Scientific
Hayden Book Company
Investors Software
Rocky Mountain Software
RTR Software
Sophisticated Microsystems
Stockwatch
Telesphere Corp

(Continued)

Telecommunications, Terminal Driver

Advanced Business Technology
Harvey's Space Ship Repair
Home Banking System
Microcom
Personal Software
Rocky Mountain Software
Software Sorcery
Southeastern Software
Southwestern Data Systems
Telephone Software Connection

Telecommunications, Telex & TWX

Microcom

Text Editor/Word Processor

Artsci, Inc.
Aurora Systems
Charles Mann & Assoc.
Data Transforms
Harvey's Space Ship Repair
Hayden Book Co.
Information Unlimited Software
M.D. Software
Muse Software
On-Line Systems
Organic Software
Peachtree Software
Personal Business Systems
Programma International
Rainbow Computing
Select Information Systems
Southwestern Data Systems
Sympathetic Software
Systems Design Software

Utilities, Applesoft

Dakin5
Highlands Computer Services
Soft CTRL Systems

(Continued)

Utilities/Diagnostics

High Sierra Software
Nikrom Technical Products
XPS Inc.

Utilities, Disk

Beagle Bros. Micro Software
Decision Systems
Highlands Computer Services
Image Computer Products
M.D. Software
Omega Software Products
Practical Software LTD
Rocky Mountain Software
Soft CTRL Systems
Sympathetic Software
Western MicroData Enterprises
VK Utilities

Utilities, Machine Language Level

Compu-Tron Software Services

Utilities, Pascal

Advanced Business Technology
Denver Software Co.
Sensible Software

Utilities, ROM

Highlands Computer Services
Soft CTRL Systems

Visicalc Extender

Aurora Systems

Issued 29 Sep 81

Vendor List

Page 9900.004.01

MISCELLANEOUS PRODUCTS

Books

Computer Book Club
Dilithium Press
Hayden Book Co
Mc Graw Hill
Osborne
Sybex

Computer Summer Camp

Computer Camp

Computer Case

Cases Inc
Computer Case Co

Computer Cabinet

Softsel

Data Format Conversion Service, Apple II <=> DEC RT-11

Prestige Marketing Corp

Data Format Conversion Service, Apple II <=> IBM

Prestige Marketing Corp

Data Format Conversion Service, Apple II <=> TRS-80

Prestige Marketing Corp

Disk 13 Sector & 16 Sector Switch

Microcomputer Center
Micro-Ware

(Continued)

APPLE TECH NOTES

Copyright (C) 1981 by Apple Computer, Inc.

"AppleI ACTN July82_9900-004-01" 66 KB 1962-10-11 dpi: 300h x 300v pix: 1954h x 2978v

Disk Mass Reproduction

Dysan

Documentation

Computer Station

Forms

Nebs Computer Forms
Regent Standard Forms

Magazines

Byte
Creative Computing
Interface Age
Kilobaud/Microcomputing
onComputing
Robotics Age

Newsletter

Apple Educators' Newsletter

EPR0M Programming

Logic Technology Services

Software Directories

Skarbeks Software Directory
The Book 1981
Vital Information
Vanloves Software Directory

Want Ads

Ad-Line

PRINTER MANUFACTURERS

Alphacom
 2323 S. Basocm
 Campbell, CA 95008
 (408) 559-8000

Anadex, Inc.
 9825 De Soto Ave.
 Chatsworth, CA 91311
 (213) 998-8010

Anderson-Jacobson, Inc.
 521 Charcot Ave.
 San Jose, CA 95131
 (408) 263-8520

Axiom Corp.
 5932 San Fernando Rd.
 Glendale, CA 91202
 (213) 245-9244

Capitol Circuits
 Printer Products Div.
 24 Denby Rd
 Allston, MA 02134
 (617) 787-2030

Centronics
 Route 111
 Hudson, NH 03051
 (603) 883-0111

C. Itoh Electronics
 5301 Beethoven St.
 Los Angeles, CA 90066
 (213) 390-7778

DataSouth Computer Corp.
 4740 Dwight Evans Rd.
 Charlotte, NC 28210
 (704) 523-8500

Dip Inc
 745 Atlantic Av
 Boston, MA 02111
 (617) 482-4214

(Continued)

Epson America Inc.
23844 Hawthorne Blvd.
Torrance, CA 90505
(213) 378-2220

Howard Industries Inc
2031 E. Cerritos Av, Bldg 7K
Anaheim, CA 92806
(714) 778-3443

Integral Data Systems
Milford, NH 03055
(603) 673-9100

InterSell
465 Fairchild Dr, #214
Mountain View, CA 94043
(415) 964-5460

Malibu
2301 Townsgate Rd
Westlake Village, CA 91361
(805) 496-1990

Micro Peripherals Inc.
4426 S. Century Dr.
Salt Lake City, UT 84107
(801) 263-3081

Microtek Inc
9514 Chesapeake Dr
San Diego, CA 92132
(714) 278-0633

NEC Information Systems Inc.
5 Militia Dr.
Lexington, MA 02173
(617) 862-3120

Okidata Printers/Eakins Assoc. Inc.
999 Independence Ave.
Mtn. View, CA 94043
(415) 969-4533

Pertec
12910 Culver Blvd.
Los Angeles, CA 90066
(213) 822-9222

(Continued)

Issued 29 Sep 81

Vendor List

Page 9900.005.03

Printronic, Inc.
17421 Derian Ave.
Irvine, CA 92714
(714) 549-8272

Qantex
60 Plant Ave.
Hauppauge, NY 11787
(516) 582-6060

Qume Corp.
2350 Qume Dr.
P.O. Box 50039
San Jose, CA 95150
(408) 942-4200

Trendcom
484 Oakmead Pkwy
Sunnyvale, CA 94086
(408) 737-0747

Victor Data Systems
3900 N. Rockwell St
Chicago, IL 60618
(312) 539-8200

XYmec
17905 Sky Park Circle, Suite J
Irvine, CA 92714
(714) 557-8501

Issued 29 Sep 81

Vendor List

Page 9900.006.01

VENDORS - NUMERIC

3-G Co.
Route 3, Box 28A, Dept. BT
Gaston, OR 97119
(503) 662-4492

VENDORS - A

Aardvark Software
783 N. Water Street
Milwaukee, WI 53202
(414) 289-9988

Abacus Software
PO Box 7211
Grand Rapids, MI 49510

ABW Corp.
PO Box M 1047
Ann Arbor, MI 48106
(313) 971-9364

Action-Research Northwest
11442 Marine View Drive SW
Seattle, WA 98146
(206) 244-9360

Addmaster Corp.
Box 387
416 Junipero Serra Drive
San Gabriel, CA 91776
(213) 285-1121

Ad-Line
PO Box 15842
Philadelphia, PA 19103
(215) 462-4415

Advanced Business Technology (Abtech)
12333 Saratoga-Sunnyvale Road
Saratoga, CA 95070
(408) 446-2013

Advanced Logic Systems
1026 West Maude Ave, Suite 305
Sunnyvale, CA 94086
(408) 730-0306

Adventure International
PO Box 3435
Longwood, FL 32750
(305) 862-6917

(Continued)

Adwar Video Corp.
100 Fifth Avenue
New York, NY 10011
(212) 691-0976

ALF Products
1448 Estes
Denver, CO 80215
(303) 234-0871

Amdek Corp.
2420 East Oakton Street, Suite E
Arlington Heights, IL 60005
(312) 364-1180

Andromeda Inc.
PO Box 19144
Greensboro, NC 27410
(919) 852-1482

Apparat, Inc.
4401 South Tamarac Parkway
Denver, CO 80237
(303) 741-1778

Application Software Development
504 Lakemead Way
Redwood City, CA 94062
(415) 366-6124

Applied Analytics Inc.
5406 Roblee Drive
Upper Marlboro, MD 20870

Applied Engineering
PO Box 470301
Dallas, TX 75247
(214) 492-2027

Arizona Computer Systems Inc.
PO Box 125
Jerome, AZ 86331
(602) 634-7301

Artsci
10432 Burbank Blvd
North Hollywood, CA 91601
(213) 985-2922

(Continued)

Issued 29 Sep 81

Vendor List

Page 9900.007.03

ATV Research
13B Broadway
Dakota City, NE 68731
(402) 987-3771

Aurora Systems
2040 East Washington Avenue
Madison, WI 53704
(608) 249-5875

Avant-Garde Creations
PO Box 30161
Eugene, OR 97403
(503) 345-3043

VENDORS - B

BCD Associates
1216 Blackwelder Avenue
Oklahoma City, OK 73106
(405) 524-7403

Beagle Bros. Micro Software
Dept. F
4315 Sierra Vista
San Diego, CA 92103
(714) 296-6400

John Bell Engineering
PO Box 338, Dept. 8
Redwood City, CA 94064
(415) 367-1137

Bit-3 Computer Corp.
1890 Huron Street
St. Paul, MN 55113
(612) 926-6997

The Book Company
16720 Hawthorne Blvd.
Lawndale, CA 90260
(213) 371-4012

BPI Systems
1600 W. 38th, Suite 444
Austin, TX 78731
(512) 454-2801

Broderbund Software
Box 3266
Eugene, OR 97403
(503) 343-9024

BSR Ltd.
Route 303
Blauvelt, NY 10913
(914) 358-6060

BudgeCo.
428 Pala Avenue
Piedmont, CA 94611
(415) 658-8141

(Continued)

Business & Professional Software
PO Box 11 Kendall Square Branch
238 Main Street
Cambridge, MA 02142
(617) 491-3377

BYTE Books
70 Main Street
Peterborough, NH 03458

Byte
PO Box 590
Martinsville, NJ 08836

Issued 29 Sep 81

Vendor List

Page 9900.009.01

VENDORS - C

California Computer Systems
250 Caribbean Drive
Sunnyvale, CA 94086
(408) 734-5811

Cameo Electronics Inc.
1626 Clementine
Anaheim, CA 92802
(714) 535-1682

Cases, Inc.
PO Box 33820
Seattle, WA 98133
(206) 365-5210

Cavri Systems
26 Trumbull Street
New Haven, CT 06511
(203) 562-9873
(203) 562-4979

Charles Mann & Associates
7594 San Remo Trail
Yucca Valley, CA 92284
(714) 365-9718

CJM Industries Inc.
PO Box 2367
Reston, VA 22090
(703) 620-2444

Color Software
PO Box 24214
Indianapolis, IN 46224

Compdent
542 Lake Street
Hancock, MI 49930

Compumax
PO Box 1139
Palo Alto, CA 94301
(415) 321-2881

CompuSoCo
PO Box 2325
26251 Via Roble
Mission Viejo, CA 92690

(Continued)

CompuServe
Personal Computing Division
5000 Arlington Centre Blvd
Columbus, OH 43220
(614) 457-8600

Compu TA
9533 Grossmont Blvd.
La Mesa, CA
(714) 469-3017

CompuTalker
1730 21st Street
Santa Monica, CA 90404
(213) 392-5230

Computer Applications Tomorrow
PO Box 605
Birmingham, AL 48012

Computer Book Club
Blue Ridge Summit, PA 17214

Computer Camp
1235 Coast Village Road, Suite G
Santa Barbara, CA 93108
(805) 965-7777

Computer Case Co.
5650 Indian Mound Court
Columbus, OH 43213
(614) 868-9464

Computer Mart
PO Box 1664
Lake Havasu, AZ 86403
(602) 855-3357

Computer Services Co.
14109 SE 168th Street
Renton, WA 98055
(206) 255-7410

Computer Station
12 Crosswoods Plaza
Granite City, IL 62040
(618) 452-1860

(Continued)

Computer Stop
16919 Hawthorne Blvd
Lawndale, CA 90260
(213) 371-4010

Computer Systems Design Group
3632 Governor Drive
San Diego, CA 92122
(415) 856-1954

CompuTime
PO Box 5343
Huntington Beach, CA 92646
(714) 536-5000

Computer Translation, Inc.
Dept. BPI
PO Box 7004 University Station
Provo, UT 84602
(801) 224-1169

Compu-Tron Software Services
1414 South Fairplain Avenue
Whittier, CA 90601
(213) 336-7353

Connecticut Information Systems
218 Huntington Road
Bridgeport, CT 06608
(203) 579-0472

Continental Software
12101 Jefferson Blvd
Culver City, CA 90230
(213) 371-5612

Co-Op Software
PO Box 432
West Chicago, IL 60185
(312) 231-0912

Corvus
2029 O'Toole Avenue
San Jose, CA 95131
(408) 946-7700

Cover Craft
PO Box 555
Amherst, NH 03031
(603) 889-6811

(Continued)

CPU Inc
5161 Atlanta Highway
Montgomery, AL 36109

Creative Computing Software
PO Box 789-M
Morristown, NJ 07960
(201) 540-0445

Cybertech
PO Box 504
Allen, TX 75002
(214) 727-3900

Cyborg Corp.
342 Western Avenue
Boston, MA 92135
(617) 782-9820

Crystal Computer
12215 Murphy Avenue
San Martin, CA 95046
(408) 683-0696

Crystalware
12215 Murphy Avenue
San Martin, CA 95046
(408) 683-0696

VENDORS - D

D.J. 'AI' Systems Ltd.
Station Road
Ilminster, Somerset
TA19 9BQ, England
04605 4117

Dakin5
PO Box 21187
Denver, CO 80221
(800) 525-0463

Data-Safe Products Inc.
1926 Margaret Street
Philadelphia, PA 19124
(215) 535-3004

Datasoft Inc.
16606 Schoenborn Street
Sepulveda, CA 91343
(213) 894-9154

Data Transforms
905 E. Fifth Avenue
Denver, CO 80213
(303) 772-8774

Decision Master
10428 Westpark
Houston, TX 77042
(800) 231-5768 ext. 306
(800) 392-2348 in TX

Decision Systems
PO Box 13006
Denton, TX 76203

Denver Software Co.
36 Steele Street, Suite 19
Denver, CO 80206
(303) 321-4551

Digital Systems Engineering
12503 King's Lake Drive
Reston, VA 22091

(Continued)

dilithium Press
PO Box 606
Beaverton, OR 97075
(503) 243-1158

Dockside Computing
PO Box 5030
Westlake Village, CA 91362

Dow Jones Information Services
PO Box 300
Princeton, NJ 08540
(800) 257-5114

Dymarc Industries Inc.
7133 Rutherford Road
Baltimore, MD 21207
(301) 298-3130

Dynacomp Inc
1427 Monroe Avenue
Rochester, NY 14618
(716) 442-8960

Dysan Corp.
5440 Patrick Henry Dr
Santa Clara, CA 95050
(408) 988-3472

VENDORS - E

Eastern House Software
 3239 Linda Drive
 Winston-Salem, NC 27106
 (919) 924-2889
 748-8446

Eclectic Corp
 2830 Walnut Hill Lane
 Dallas, TX 75229

Educational Courseware
 3 Nappa Lane
 Westport, CT 06880

Educational Services of Alaska
 Box 145
 McGrath, AK 99627
 (907) 524-3892

Educational Programming Systems
 1328 Baur Blvd.
 St. Louis, MO 63132
 (314) 991-0300

Edu-Comp, Inc.
 14109 SE 168th Street
 Renton, WA 98055
 (206) 255-7410

Edu-Ware Services
 22222 Sherman Way #102
 Canoga Park, CA 91303
 (213) 346-6783

Electronic Specialists Inc.
 171 S. Main Street
 Natick, MA 01760
 (617) 655-1532

Electronic Systems
 PO Box 21638
 San Jose, CA 95151
 (408) 448-0800

Escon Products Inc.
 12919 Alcosta Blvd.
 San Ramon, CA 94583
 (415) 820-1256

(Continued)

ESP Computer Resources
9 Ash Street
Hollis, NH 03049
(603) 465-7264

Issued 29 Sep 81

Vendor List

Page 9900.012.01

VENDORS - F

(no entries yet)

Issued 29 Sep 81

Vendor List

Page 9900.013.01

VENDORS - G

Galaxy
Dept. MI5
PO Box 22072
San Diego, CA 92122
(714) 452-1072

GNT Automatic Inc.
1560 Trapelo Road
Waltham, MA 02154
(617) 890-3305

GS Computer Enterprises
PO Box 8050
Ann Arbor, MI 48107
(313) 665-6416

VENDORS - H

H & H Scientific
13507 Pendleton Street
Oxon Hill, MD 20022
(301) 292-3100

Hartley Software
3268 Coach Lane #2A
Kentwood, MI 49508
(616) 942-8987

Harvey's Space Ship Repair
Box 3478-C
Las Cruces, NM 88003
(505) 522-1482

Hayden Book Co.
50 Essex Street
Rochelle Park, NY 07662

Hayes Microcomputer Products
5835 Peachtree Corners East
Norcross, GA 30092
(404) 449-8791

Hazeltine Corp.
Computer Terminal Equipment
Greenlawn, NY 11740
(516) 549-8800

Health Data Products Inc.
222 E. Anapamu Street
Santa Barbara, CA 93101
(805) 965-4477

Heuristics Inc.
1285 Hammerwood Avenue
Sunnyvale, CA 94086
(408) 734-8532

High Sierra Software
5541 Highway 50 East, Suite 2A
Carson City, NV 89701
(702) 883-6590

High Technology Inc.
Software Products Division
PO Box B-14665

(Continued)

8001 N. Classen Blvd.
Oklahoma City, OK 73113
(405) 840-9900

Highlands Computer Services
1422 SE 132nd
Renton, WA 98055
(206) 228-6691

Home Banking System
PO Box 72 Radio City Station
New York, NY 10101

Home Computer Center, Inc.
2927 Virginia Beach Blvd.
Virginia Beach, VA 23452
(804) 340-1977

Houston Instruments
One Houston Square
Austin, TX 78752
(512) 837-2820

Howard Software Services
7722 Hosford Avenue
Los Angeles, CA 90045
(213) 645-4069

Human Systems Dynamics
9249 Reseda Blvd, Suite 107a
Northridge, CA 91324

Issued 29 Sep 81

Vendor List

Page 9900.015.01

VENDORS - I

Illinois Computer Mart
Route 8, Sweet Corners Plaza
Carbondale, IL 62901

Image Computer Products
615 Academy Drive
Northbrook, IL 60062
(312) 564-5060

Information Unlimited Software
281 Arlington Avenue
Berkeley, CA 94707
(415) 525-9452

Information Technologies Inc.
2816 Harvard East
Seattle, WA 98102
(206) 325-0430

Innovision
PO Box 1317
Los Altos, CA 94022

Insoft O'Donnel Co.
259 Barnett Road, Unit 2
Medford, OR 97501
(503) 779-2465

Instant Software
Peterborough, NH 03458
(603) 924-7296

Intelligent Control Systems, Inc.
PO Box 14571
Minneapolis, MN 55414
(612) 699-4342

Intelligent Computer Systems Corp.
722 South 24th Street
Arlington, VA 22202
(703) 684-7389

Interactive Microware Inc.
PO Box 771
State College, PA 16801
(814) 238-8294

(Continued)

Interactive Structures
PO Box 404
112 Bala Avenue
Bala Cynwyd, PA 19004
(215) 667-1713

Interlude
10428 Westpark
Houston, TX 77042
(800) 231-5768 ext. 306

International Apple Core
PO Box 976
Daly City, CA 94017

Investors Software
PO Box 2605
San Francisco, CA 94126
(415) 981-5261

Ipex International
16140 Valerio Street
Van Nuys, CA 91406
(213) 781-0020

Issued 29 Sep 81

Vendor List

Page 9900.016.01

VENDORS - J

D.R. Jarvis Computing
1039 Cadiz Drive
Simi, CA 93065
(805) 526-0151

Jasac+
PO Box 7000-287
Palos Verdes Peninsula, CA 90274
(213) 541-5973

Issued 29 Sep 81

Vendor List

Page 9900.017.01

VENDORS - K

Kinetic Systems Inc.
PO Box 4111
Baton Rouge, LA 70821
(504) 383-5369

VENDORS - L

Lamar Instruments
2107 Artesia Blvd
Redondo Beach, CA 90278
(213) 374-1673

Lax Computer Products
4728 Manhattan Beach Blvd.
Lawndale, CA 90260
(213) 542-4505

Lazer Systems
PO Box 55518
Riverside, CA 92517
(714) 682-5268

Link Systems
1655 26th Street
Santa Monica, CA 90404
(213) 871-1511

Livermore Data Systems Inc.
2050 Research Drive
Livermore, CA 94550

LJK Enterprises Inc.
PO Box 10827
St. Louis, MO 63129
(314) 846-2313

Lobo Drives International
354 South Fairview Avenue
Goleta, CA 93117
(805) 683-1576

Logic Technology Services, Inc.
2400 E. Oakton
Arlington Heights, IL 60005
(312) 364-4670

Londe, Parker, Michels
7438 Forsyth, Suite 202
St. Louis, MO 63105
(314) 725-5501

VENDORS - M

M&R Enterprises
 PO Box 61011
 Sunnyvale, CA 94088
 (408) 738-3772

Macrotronics
 PO Box 518(A)
 Keyes, CA 95328
 (209) 667-2888

Maryland Computer Services
 502 Rock Springs Avenue
 Bel Air, MD 21014
 (301) 838-8888
 879-3366

Mast Development Co.
 2212 E. 12th Street
 Davenport, IA 52803
 (319) 326-0141

Mastertype
 PO Box 5223
 Stanford, CA 94305

Matchless Systems
 18444 S. Broadway
 Gardena, CA 90248
 (213) 327-1010

Math City
 4040 Palos Verdes Drive North
 Rolling Hills Estates, CA 90274
 (213) 541-3377

Mauro Engineering
 2220 Pack Trail
 Mount Shasta, CA 96067
 (916) 926-4406

Mc Graw Hill
 70 Main Street
 Peterborough, NH 03458
 (800) 258-5420

(Continued)

M.D. Software
7225 Clayton Road
St. Louis, MO 63117
(314) 725-1110

Med Logic Systems
2860 Walnut Avenue, Suite C
Tustin, CA 92680

MFJ Enterprises Inc.
PO Box 494
Mississippi State, MS 39762
(601) 323-5869

Microcom Inc
6 Faneuil Hall Marketplace
Boston, MA 02109
(617) 367-6362

Micro Computer Center
7900 Paragon Road
Dayton, OH 45459
(513) 435-9355

Micro Lab
3218 Skokie Valley Road
Highland Park, IL 60035
(312) 433-7550

Micro Learningware
PO Box 2134
North Mankato, MN 56001
(507) 625-2205

Micromate Electronics Inc.
2094 Front Street
East Meadow, NY 11554
(516) 794-1072

Micro Mind Inc.
917 Midway
Woodmere, NY 11598
(516) 374-6793

Micro Motion
12077 Wilshire Blvd, #506
Los Angeles, CA 90025
(213) 821-4340

(Continued)

Microproducts
30420 Via Rivera
Rancho Palos Verdes, CA 90274
(213) 541-5131

Micro Products Unlimited
PO Box 1525
Arlington, TX 76010
(817) 461-8043

Microsoft Consumer Products
400 108th Avenue NE, Suite 200
Bellevue, WA 98004
(206) 454-1315

Micro-Ware Distributing Inc.
PO Box 113
Pompton Plains, NJ 07444
(201) 839-3478

Micro Works, Inc.
PO Box 1110
Del Mar, CA 92014
(714) 942-2400

Min Microcomputer Software, Inc.
5835-A Peachtree Corners East
Norcross, VA 30092
(404) 447-4322

Monarch Computer Products, Inc.
PO Box 4081
New Windsor, NY 12550
(914) 562-3100

Moneydisk
PO Box 1531
Richland, WA 99352
(509) 943-9004

Mountain Computer
300 El Pueblo
Scotts Valley, CA 95066
(408) 438-6650

MR Engineering Co.
4730 W. Addison
Chicago, IL 60641
(312) 286-6606

(Continued)

Muse Software
330 N. Charles Street
Baltimore, MD 21201
(301) 659-7212

Mytopia Gameware Institute
PO Box 625
Sioux City, IA 51102

VENDORS - N

Nebs Computer Forms
 78 Hollis Street
 Groton, MA 01450
 (800) 225-9550

Nestar Systems Inc.
 2585 East Bayshore Road
 Palo Alto, CA 94303
 (415) 493-2223

Neutronics R&D Ltd
 333 Litchfield Road
 New Milford, CT 06776
 (203) 354-9375

Nikrom Technical Products
 25 Prospect Street
 Leominster, MA 01453

Novation Inc.
 18664 Oxnard Street
 Tarzana, CA 91356
 (213) 996-5060

VENDORS - 0

Ohio Data Products Corp.
 14600 Detroit Avenue
 Cleveland, OH 44107
 (216) 221-9000

Omega Research
 PO Box 479
 Linden, CA 95236
 (209) 334-6666

Omega Software Products
 222 S Riverside Plaza
 Chicago, IL 60606
 (312) 648-1944

Omega Software Systems Inc.
 1574 Ives Dairy Road
 North Miami, FL 33179

Omnico Computer Corp.
 3300 Buckeye Road
 Atlanta, GA 30341
 (404) 455-8460

Omnitec Data
 2405 South 20th Street
 Phoenix, AZ 85034
 (800) 528-8423

On-Line Systems
 36575 Mudge Ranch Road
 Coarsegold, CA 93614
 (209) 683-6858

Optimal Technology Inc.
 Blue Wood 127
 Earlysville, VA 22936
 (804) 973-5482

Organic Software
 1492 Windsor Way
 Livermore, CA 94550
 (415) 455-4034

Osborne/McGraw-Hill
 630 Bancroft Way, Dept B13
 Berkeley, CA 94710
 (415) 548-2805

(Continued)

Issued 29 Sep 81

Vendor List

Page 9900.022.01

VENDORS - P

Passport Designs
PO Box 478
La Honda, CA 94020
(415) 747-0614

Paymar, Dan
91 Pioneer Drive
Durango, CO 81301
(303) 259-3598

Peachtree Software
3 Corporate Square, Suite 700
Atlanta, GA 30329
(800) 835-2246 ext. 35

Personal Business Systems
4306 Upton Avenue South
Minneapolis, MN 55410

Personal Software
1330 Bordeaux Drive
Sunnyvale, CA 94086
(408) 745-7841

Powersoft
PO Box 157
Pitman, NJ 08071
(609) 589-5500

Practical Software LTD.
Dept. CC
PO Box 3000
Pomona, NY 10970

Prentice Corp.
266 Caspian Drive
Sunnyvale, CA 94086
(408) 734-9810

Prestige Marketing Corp.
909 North Coliseum Blvd, Suite 109
Fort Wayne, IN 46805
(219) 423-1517

Professional Medical Software
3604 Foothill Boulevard
La Crescenta, CA 91214
(213) 248-2884

(Continued)

Programma International
2908 N. Naomi Street
Burbank, CA 91504
(213) 954-0240

Programs Unlimited
Dept. 0581
PO Box 265
Jericho, NY 11753
(800) 645-6038

Progressive Software
PO Box 273
Plymouth Meeting, PA 19462

Prometheus Products Inc.
4509 Thompson Court
Fremont, CA 94538
(415) 791-0266

Prosoft
3604 Foothill
LaCrescenta, CA
(213) 248-2884

Protean Enterprises
PO Box 1284
Olympia, WA 98507
(206) 357-6197

Issued 29 Sep 81

Vendor List

Page 9900.023.01

VENDORS - Q

Qkit
PO Box 35879
Tucson, AZ 85740
(602) 299-9831

Quality Software
6660 Reseda Blvd, Suite 105
Reseda, CA 91335
(213) 344-6599

VENDORS - R

Racal-Vadic
222 Caspian Drive
Sunnyvale, CA 94086
(408) 744-0800

Rainbow Computing Inc.
9719 Reseda Blvd
Northridge, CA 91324
(213) 349-5560

Rak-Ware
41 Ralph Road
West Orange, NJ 07052

Rank Electro-Media LTD.
1237 - 47 Avenue NE
Calgary, Alberta, Canada T2E 6N7
(403) 276-3533

RCA
Microcomputer Products Marketing
New Holland Avenue
Lancaster, PA 17604
(717) 697-7661

Regent Standard Forms, Inc.
501 Benigno Blvd
Bellman, NJ 08031
(609) 933-0480

Retail Sciences
3 Corproate Square, Suite 700
Atlanta, GA 30329
(404) 325-8533

R.H. Electronics
3125 19th Street, Suite 173
Bakersfield, CA 93301
(805) 688-2047

RKS Enterprises, Inc.
643 South 6th Street
San Jose, CA 95112
(408) 288-5565

Robotics Age
PO Box 512
Tujunga, CA 91042

(Continued)

Rochester Data Inc.
3000 Winton Road South
Rochester, NY 14623
(716) 244-7804

Rocky Mountain Software Inc.
1038 Hamilton Street
Vancouver, BC, Canada V6B 2R9
(604) 681-3371

RTR Software Inc.
1147 Baltimore Drive, Dept. M1
El Paso, TX 79902
(915) 544-4397

VENDORS - S

Howard W. Sams & Co.
 PO Box 7092
 4300 W. 62nd Street
 Indianapolis, IN 46206

Scan-Tron Corp.
 PO Box 4273
 Burlingame, CA 94010
 (415) 697-9651

Sci Tronics
 523 S. Clewell Street
 PO Box 5344
 Bethlehem, PA 18015
 (215) 868-7220

Scott Instruments, Inc.
 815 N. Elm, Suite 5
 Denton, TX 76201
 (817) 387-9514

Select Information Systems
 919 Sir Francis Drake Blvd.
 Kentfield, CA 94904
 (415) 459-4003

Sensational Software
 PO Box 789-M
 Morristown, NJ 07960

Sensible Software
 6619 Perham Drive
 West Bloomfield, MI 48033
 (313) 399-8877

Serendipity Systems
 225 Elmira Road
 Ithaca, NY 14850
 (607) 277-4889

Shafer Software
 749 W. Fremont Avenue
 Sunnyvale, CA 94087

Sirius Software
 2011 Arden Way, #225A
 Sacramento, CA 95825
 (916) 920-1939

(Continued)

Skarbeks Software Directory
11990 Dorsett Road
Maryland Heights, MO 63043
(314) 567-3291

SMD
4545 Fuller Drive, Suite 225
Irving, TX 75062
(214) 258-6636

Smart Ware
2281 Cobble Stone Court
Dayton, OH 45431

Softsel
4079 Glencoe Avenue
Marina Del Rey, CA 90291
(213) 822-8933

Softech International Corporation
144 W. 15th Street, Suite #6
North Vancouver, BC, Canada V7M 1R5
(604) 984-0477

Soft CTRL Systems
Box 599
West Milford, NY 07480

Soft Touch
PO Box 7200
Costa Mesa, CA 92626

Software Publishing Corp.
PO Box 50575
Palo Alto, CA 94303
(415) 368-7598

Software Sorcery Inc.
7927 Jones Branch Drive, Suite 400
McLean, VA 22102
(703) 385-2944

Software Technology for Computers
PO Box 428
Belmont, MA 02178

Sophisticated Microsystems
550 Frontage Road
Northfield, IL 60093
(312) 441-5802

(Continued)

Sorrento Valley Associates
 11722 Sorrento Valley Road
 San Diego, CA 92121
 (714) 452-0101

Southeastern Software
 6414 Derbyshire Drive
 New Orleans, LA 70126
 (504) 246-8438

Southwestern Data Systems
 PO Box 582-C
 Santee, CA 92071
 (714) 562-3670

Special Delivery Software
 Apple Computer Inc.
 (408) 744-0630

Spectrum Software
 PO Box 2084
 142 Carlow Drive
 Sunnyvale, CA 94087
 (408) 738-4387

SSM Microcomputer Products
 2190 Paragon Drive
 San Jose, CA 95131
 (408) 946-7400

Stellation Two
 PO Box 2342
 Santa Barbara, CA 93120
 (805) 966-1140

Stockwatch
 13773 N. Central Expressway
 Suite 1224 Keystone Park
 Dallas, TX 75243
 (214) 235-4414

Stoneware Micro Computer Products
 50 Belvedere Street
 San Rafael, CA 94901
 (415) 454-6500

Strategic Simulations Inc.
 465 Fairchild Drive, #108
 Mountain View, CA 94043

(Continued)

Street Electronics
3152 E. La Palma Avenue, Suite C
Anaheim, CA 92806
(714) 632-9950

subLogic Communications Corp.
Box V
Savoy, IL 61874
(217) 359-8482

Sybex
2344 Sixth Street, Dept. B31
Berkeley, CA 94710
(415) 848-8233

Sympathetic Software
9531 Telhan Drive
Huntington Beach, CA 92646

Symtec, Inc.
PO Box 462
Farmington, MI 48024
(313) 272-2950

Synergistic Software
5221 - 120th Avenue SE
Bellevue, WA 98006
(206) 641-1917

Syntauri Corp.
3506 Waverly Street
Palo Alto, CA 94306
(415) 494-1017

System Design Software (SDL)
2612 Artesia Blvd, Suite B
Redondo Beach, CA 90278
(213) 374-4471

VENDORS - T

Tally
8301 South 180th Street
Kent, WA 98031

Talos Systems Inc.
7419 East Helm Drive
Scottsdale, AZ 85260
(602) 948-6540

TASA (Touch Activated Switch Arrays)
2346 Walsh Avenue
Santa Clara, CA 95050
(408) 727-8272

Tecmar, Inc.
23600 Mercantile Road
Cleveland, OH 44122
(216) 464-7410

Teksim
ABW Corp.
PO Box M 1047
Ann Arbor, MI 48106
(313) 971-9364

Telephone Software Connection
PO Box 6548
Torrance, CA 90504
(213) 329-3715

Telesensory Systems
3408 Hillview Avenue
Palo Alto, CA 94304
(415) 493-2626

Telesphere Corp.
1605 John Street
Fort Lee, NJ 07024
(201) 886-2476

TeleVideo
2149 Paragon Drive
San Jose, CA 95131
(408) 946-8500

Tercer Medio
Apartado de Correos 62533
Caracas 1060-A Venezuela

(Continued)

Terrapin Inc.
678 Massachusetts Avenue #205
Cambridge, MA 02139
(617) 492-8816

Texcom Engineering Associates
PO Box 24472
Houston, TX 77013
(713) 458-3720

Thunderware Inc.
PO Box 13322
44 Hermosa Avenue
Oakland, CA 94618
(415) 652-1737

TSE-Hardside
6 South Street
Milford, NH 03055
(603) 673-5144

Issued 29 Sep 81

Vendor List

Page 9900.027.01

VENDORS - U

United Software of America
750 3rd Avenue
New York, NY 10017
(212) 682-0347

Universal Data Systems
5000 Bradford Drive
Huntsville, AL 35805
(205) 837-8100

U.S. Robotics
203 N. Wabash, Suite 1718
Chicago, IL 60601
(312) 346-5650

VENDORS - V

Van Dusen, Stanley
33170 Little John Drive
Coarsegold, CA 93614
(209) 683-4793

Velostat
3M Corp.
Data Recording Products Division
223-5N 3M Center
St. Paul, MN 55101

Versa Computing Inc.
887 Conestoga Circle
Newbury Park, CA 91320
(805) 498-1956

Vertical Data Systems
1215 Meyerside Drive, Unit 2A
Mississauga, Ontario, Canada L5T 1H3

Victor Data Products
3900 N. Rockwell Street
Chicago, IL 60618
(312) 539-8200

Video Marketing Inc.
PO Box 339
Warrington, PA 18976
(215) 343-3000

Videx
897 NW Grant Avenue
Corvallis, OR 97330
(503) 758-0521

Vital Information
350 Union Station
Kansas City, MO 64108
(913) 384-3860

VK Utilities
4147 California Avenue SW
Seattle, WA 98116

Voicetek
PO Box 388
Goleta, CA 93116
(805) 685-1854

(Continued)

Votrax
500 Stephenson Highway
Troy, MI 48084
(313) 588-0341

Issued 29 Sep 81

Vendor List

Page 9900.029.01

VENDORS - W

Welco Industries
9027 Shell Road
Cincinnati, OH 45236
(513) 891-6600

West Coast Consultants
1775 Lincoln Blvd.
Tracy, CA 95376
(209) 835-1785

West Side Electronics
PO Box 636A
Chatsworth, CA 91311
(213) 884-4794

Western Micro Data Enterprises
PO Box G33 Postal Station G
Calgary, Alberta, Canada T3A 2W1
(403) 247-1621

Westware Software Inc.
2455 SW 4th Avenue, Suite 2
Ontario, OR 97914

Whitney Educational Service
2071 Tenth Avenue
San Francisco, CA 94116
(415) 681-4725

WIDL Video
5245 West Diversey
Chicago, IL 60639
(312) 622-9606

John Wiley & Sons Inc.
605 Third Avenue
New York, NY 10158

Issued 29 Sep 81

Vendor List

Page 9900.030.01

VENDORS - X

XPS Inc.
323 York Road
Carlisle, PA 17013

Issued 29 Sep 81

Vendor List

Page 9900.031.01

VENDORS - Y

(no entries yet)

Issued 29 Sep 81

Vendor List

Page 9900.032.01

VENDORS - Z

Zeus Computing, Al
PO Box 712
Downey, CA 90241

Ziatech
2410 Broad Street
San Luis Obispo, CA 93401
(805) 544-9011

apple tech notes





