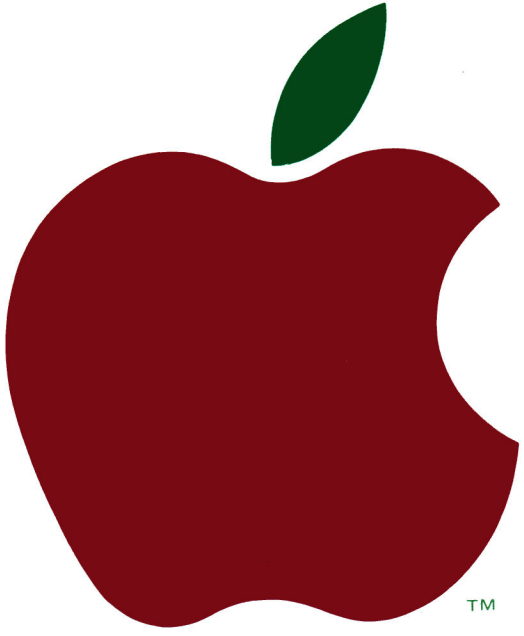
# Computer Station's

# Programmer's Guide

# to the

# APPLE II



Produced by

**Computer Station**
**3659 Nameoki Road**
**Granite City, Illinois 62040**



All material in this manual was edited from the following Apple manuals:

1. Apple II Reference Manual - Jan. 1978
2. Applesoft II Reference Manual - Aug. 1978
3. Applesoft II Programming Reference Manual

**Cover Drawing: Trademark of Apple Computer, Inc.**

# Acknowledgements

The authors would like to acknowledge the kind assistance given by APPLE COMPUTER INC. during the preparation of this Programmer's Guide. We especially appreciated the advise of Mr. Gene Carter, APPLE's Director of Dealer Marketing.

# Preface

This booklet is not meant as a replacement for the three excellent publications of APPLE COMPUTER INC., rather, it is designed as a companion. The information contained in the APPLE publications could not be condensed into one booklet this size. Instead, the command syntax, operands, and special locations have been gleaned from the reference manuals and collected to yield a Programmer's reference. We assume the programmer has already read and absorbed APPLE's publications. This programmer's guide is designed as a helpful tool for speedy information retrival without referring to the entire text found in the APPLE publications.

# Table of Contents

# System Monitor

## System Monitor Commands

Apple II contains powerful machine level monitor for use by the advanced programmer. To enter the monitor either press RESET button on keyboard or CALL -151 (Hex FF65) from Basic. Apple II will respond with an "*" (asterisk) prompt character on the TV display. This action will not kill current BASIC program which may be re-entered by a $C^{C}$ (control C). NOTE: "adrs" is a four digit hexidecimal number. Remember to press "return" button at the end of each line.

| Command Format | Example | Description |
|---|---|---|
| | | **Examine Memory** |
| adrs | *CØF2 | Examines (displays) single memory location of (adrs). |
| adrs1.adrs2 | *1Ø24.1Ø48 | Examines (displays) range of memory from (adrs1) thru (adrs2). |
| (return) | *(return) | Examines (displays) next 8 memory locations. |
| .adrs2 | *.4Ø96 | Examines (displays) memory from current location thru location (adrs2). |
| | | **Change Memory** |
| adrs:data data data | *A256:EF 2Ø 43 | Deposits data into memory starting at location (adrs). |
| :data data | *:FØ A2 12 | Deposits data into memory starting after (adrs) last used for deposits. |
| | | **Move Memory** |
| adrs1 adrs2 adrs3M | *1ØØ<BØ1Ø.B41ØM | Copy the data now in the memory range from (adrs2) to (adrs3) into memory locations starting at (adrs1). |
| | *8ØØ:Ø <8Ø1 8ØØ.8FEM | Fills from 8ØØ to 8FF with Ø's. Note 1: 1 less than last location to be filled. Note 2: Fill byte is hex number immediately after colon (:). |
| | | **Verify Memory** |
| adrs1 adrs2 adrs3V | *1ØØ<BØ1Ø.B41ØV | Verify that block of data in memory range from (adrs2) to (adrs3) exactly matches data block starting at memory location (adrs1) and displays differences if any. |
| | | **Cassette I/O** |
| adrs1 adrs2R | *3ØØ.4FFR | Reads cassette data into specified memory (adrs) range. Record length must be |

| | | |
|---|---|---|
| | | same as memory range or an error will occur. |
| adrs1.adrs2W | *8ØØ.9FFW | Writes onto cassette, data from specified memory (adrs) range. |

**Display**

| | | |
|---|---|---|
| I | *I | Set inverse video mode. (Black characters on white background). |
| N | *N | Set normal video mode. (White characters on black background). |

**Dis-assembler**

| | | |
|---|---|---|
| adrsL | *C8ØØL | Decodes 2Ø instructions starting at memory (adrs) into 65Ø2 assembly mnenonic code. |
| L | *L | Decodes next 2Ø instructions starting at current memory address. |

**Mini-assembler**

| | | |
|---|---|---|
| (Turn-on) | *F666G | Turns-on mini-assembler. Prompt character is now a "!" (exclamation point). |
| $ (monitor command) | !$C8ØØL | Executes any monitor command from mini-assembler. Note that many monitor commands change current memory address reference so that it is a good practice to retype desired address reference upon return to mini-asembler. |
| adrs: (65Ø2 MNEMONIC instruction) | !CØ1Ø:STA 23FF | Assembles a mnemonic 65Ø2 instruction into machine codes. If error, machine will refuse instruction, sound bell, and reprint line with up arrow under error. |
| (space) (65Ø2 mnemonic instruction) | ! STA Ø1 FF | Assembles instruction into next available memory location. (Note space between "!" and instruction) |
| (TURN-OFF) | !(Reset Button) | Exits mini-assembler and returns to system monitor. |

**Monitor Program Execution and Debugging**

| | | |
|---|---|---|
| adrsG | *3ØØG | Runs machine level program starting at memory (adrs) |
| adrsT | *8ØØT | Traces a program starting at memory location (adrs) and continues trace until hitting a breakpoint. Break occurs on instruction ØØ (BRK), and returns control to system monitor. Opens 65Ø2 status registers (see Note 1). |

| | | |
|---|---|---|
| adrsS | *CØ5ØS | Single steps through program beginning at memory location (adrs). Type a letter S for each additional step that you want displayed. Opens 65Ø2 status registers (see Note 1). |
| (Control E) | *E$^{C}$ | Displays 65Ø2 status registers and opens them for modification (see Note 1). |
| (Control Y) | *Y$^{C}$ | Executes user specified machine language subroutine starting at memory location ($3F8). |

## Note 1:

65Ø2 status registers are open if they are last line displayed on screen. To change them type ":" then "data" for each register.

Example: A=3C X=FF Y=ØØ P=32 S=F2

| | |
|---|---|
| *:FF | Changes A register only |
| *:FF ØØ 33 | Changes A, X, and Y registers |

To change S register, you must first retype data for A, X, Y and P.

**Hexidecimal Arithmetic**

| | |
|---|---|
| data1+data2*78+34 | Performs hexidecimal sum of data1 plus data2. |
| data1-data2 *AE-34 | Performs hexidecimal difference of data1 minus data2. |

**Set Input/Output Ports**

| | |
|---|---|
| (X) (Control P) *5P$^{C}$ | Sets printer output to I/O slot number (X). (see Note 2 below) |
| (X) (Control K) *2K$^{C}$ | Sets keyboard input to I/O slot number (X). (see Note 2 below) |

## Note 2:

Only slots 1 through 7 are addressable in this mode. Address Ø (Ex: ØP$^{C}$ or ØK$^{C}$) resets ports to internal video display and keyboard. These commands will not work unless Apple II interfaces are plugged into specified I/O slot.

**Multiple Commands**

| | |
|---|---|
| *1ØØL 4ØØG AFFT | Multiple monitor commands may be given on same line if separated by a "space". |
| *LLLL | Single letter commands may be repeated without spaces. |

# Special Control and Editing Characters

"Control" characters are indicated by a super-scripted "C" such as $G^C$. They are obtained by holding down the CTRL key while typing the specified letter. Control characters are NOT displayed on the TV screen. $B^C$ and $C^C$ must be followed by a carriage return. Screen editing characters are indicated by a super-scripted "E" such as $D^E$. They are obtained by pressing and releasing the ESC key then typing specified letter. Edit characters send information only to display screen and does not send data to memory. For example, $V^C$ moves cursor to right and copies text while $A^E$ moves cursor to right but does not copy text.

| Character | Description of Action |
|---|---|
| RESET key | Immediately interrupts any program execution and resets computer. Also sets all text mode with scrolling window at maximum. Control is transferred to System Monitor and Apple prompts with a "*" (asterisk) and a bell. Hitting RESET key does NOT destroy existing BASIC or machine language program. |
| Control B | If in System Monitor (as indicated by a "*"), a control B and a carriage return will transfer control to BASIC, scratching (killing any existing BASIC program and set HIMEM: to maximum installed user memory and LOMEM: to 2Ø8. |
| Control C | If in BASIC, halts program and displays line number where stop occurred*. Program may be continued with a CON command. If in System Monitor, (as indicated by "*"), control C and a carriage return will enter BASIC without killing current program. |
| Control G | Sounds bell (beeps speaker) |
| Control H | Backspaces cursor and deletes any overwritten characters from computer but not from screen. Apply supplied keyboards have special key "←" on right side of keyboard that provides this function without using control button. |
| Control J | Issues line feed only. |
| Control V | Compliment to $H^C$. Forward spaces cursor and copies over written characters. Apple keyboards have "→" key on right side which also performs this function. |
| **Control X** | Immediately deletes current line. |
| **ESC A** | Move cursor to right. (does not copy text) |
| **ESC B** | Move cursor to left. (does not copy text) |
| **ESC C** | Move cursor down. (does not copy text) |
| **ESC D** | Move cursor up. (does not copy text) |
| **ESC E** | Clear text from cursor to end of line. |

ESC F Clear text from cursor to end of page.

ESC @ Home cursor to top of page, clear text to end of page.

*If BASIC program is expecting keyboard input, you will have to hit carriage return key after typing control C.

**NOTES:**

# Calls — Peeks, Pokes & Colors

| Hex | BASIC Example | Description |
|---|---|---|
| | **Graphics Mode Controls** | |
| C050:0 | POKE-16304,0 | Set color graphics mode. |
| C051:0 | POKE-16303,0 | Set text mode. |
| C052:0 | POKE-16302,0 | Clear mixed graphics. |
| C053:0 | POKE-16301,0 | Set mixed graphics (4 lines text). |
| C054:0 | POKE-16300,0 | Clear display Page 2 (BASIC commands use Page 1 only) |
| C055:0 | POKE-16299,0 | Set display to Page 2 (alternate). |
| C056:0 | POKE-16298,0 | Clear HIRES graphics mode. |
| C057:0 | POKE-16297,0 | Set HIRES graphics mode. |
| F836 | CALL-1994 | Clears upper 20 lines of text page 1 to reversed @ signs. If in low-resolution graphics mode, this clears the upper 40 lines of graphics screen to black (page 1 only). |
| F832 | CALL-1998 | Clears entire text page 1 to reversed @ signs. If in page 1 low-resolution graphics mode, this clears the entire screen to black. |
| F3F2 | CALL 62450 | Clears current high-resolution screen to black. |
| F3F6 | CALL 62454 | Clears current high-resolution screen to HCOLOR most recently HPLOTted. Must by preceded by a plot. |
| | **TEXT Mode Controls** | |
| FC58 | CALL-936 | (@$^{E}$) Home cursor, clear screen. |
| FC42 | CALL-958 | (F$^{E}$) Clear from cursor to end of page. |
| FC9C | CALL-868 | (E$^{E}$) Clear from cursor to end of line. |
| FC66 | CALL-922 | (J$^{C}$) Line feed. |
| FC70 | CALL-912 | Scroll up text one line. |
| 0032:3F | POKE 50, 63 | Set inverse flag if 63. |
| 0032:FF | POKE 50,255 | Set normal flag if 255. |
| | **SCROLLING WINDOW** | |
| 0020 | POKE 32,L1 | Set **LEFT** side of scrolling window to location specified by L1 in range of 0 to 39. |
| 0021 | POKE 33,W1 | Set window **WIDTH** to amount specified by W1.L1 + W1 < 40. W1 > 0. |
| 0022 | POKE 34,T1 | Set window **TOP** to line specified by T1 in range of 0 to 23. |
| 0023 | POKE 35,B1 | Set window **BOTTOM** to line specified by B1 in range of 0 to 23. B1 > T1. |

## CURSOR POSITION

| | | |
|---|---|---|
| ØØ24 | CH=PEEK(36)<br>POKE 36,CH<br>TAB(CH+1) | Read/set cursor horizontal position in the range of Ø to 39. If using TAB, you must add "1" to cursor position read value; second and third example perform identical functions. |
| ØØ25 | CV=PEEK(37)<br>POKE 37, CV<br>VTAB(CV+1) | Read/set cursor verticle position in the range of Ø to 23. If using TAB, you must add "1" to cursor position read value; second and third example perform identical functions. |

## SOUND

| | | |
|---|---|---|
| CØ3Ø<br>CØ3Ø:Ø | X=PEEK(-16336)<br>POKE-16336,Ø | Toggle speaker. (1 click in speaker) |
| CØ2Ø | X=PEEK(-16352) | Toggle cassette-output once (1 click on casette recording) |

## KEYBOARD

| | | |
|---|---|---|
| CØØØ | X=PEEK(-16384) | Read keyboard; if X>127 then key was pressed. (7F Hex) |
| CØ1Ø:Ø | POKE-16368,Ø | Clear keyboard stobe - always after reading keyboard. |

## GAME PADDLES

| | | |
|---|---|---|
| CØ61 | X=PEEK(-16287) | Read PDL (Ø) push button switch. If X>127 then switch is "on." (7F Hex) |
| CØ62 | X=PEEK(-16286) | Read PDL (1) push button switch. If X>127 then switch is "on." (7F Hex) |
| CØ63 | X=PEEK(-16285) | Read PDL (2) push button switch. If X>127 then switch is "on." (7F Hex) |
| CØ58:1 | POKE-16296,1 | Clear Game I/O ANØ output. (OFF—3.5V high) |
| CØ59:Ø | POKE-16295,Ø | Set Game I/O ANØ output. (ON—.3V low) |
| CØ5A:1 | POKE-16294,1 | Clear Game I/O AN1 output. (OFF—3.5V high) |
| CØ5B:Ø | POKE-16293,Ø | Set Game I/O AN1 output. (ON—.3V low) |
| CØ5C:1 | POKE-16292,1 | Clear Game I/O AN2 output. (OFF—3.5V high) |
| CØ5D:Ø | POKE-16291,Ø | Set Game I/O AN2 output. (ON—.3V low) |
| CØ5E:1 | POKE-16290,1 | Clear Game I/O AN3 output. (OFF—3.5V high) |
| CØ5F:Ø | POKE-16289,Ø | Set Game I/O AN3 output. (ON—.3V low) |

## Colors As Set By Color=

Note: Colors may vary depending on TV tint (hue) setting.

### Low-Resolution Colors

Ø = Black
1 = Magenta
2 = Dark Blue
3 = Light Purple
4 = Dark Green
5 = Grey
6 = Medium Blue
7 = Light Blue
8 = Brown
9 = Orange
1Ø = Grey
11 = Pink
12 = Green
13 = Yellow
14 = Blue/Green
15 = White

### High-Resolution Colors

Ø = Black 1
1 = Green
2 = Blue
3 = White 1
4 = Black 2
5 = (depends on TV)
6 = (depends on TV)
7 = White 2

**NOTES:**

# High-Resolution Operating Subroutines

**(ROM Location shown in parenthesis)**
**Other locations are for HIRES software.**

| Hex | BASIC Example | Description |
|---|---|---|
| ØCØØ<br>(DØØØ) | CALL 3Ø72<br>(CALL-12288) | **INIT — Initializes High-Resolution Graphics mode.** |

This subroutine sets High-Resolution Graphics mode with a 28Ø x 16Ø matrix of dots in the top portion of the screen and four lines of text in the bottom portion of the screen. INIT also clears the screen.

| | | |
|---|---|---|
| ØCØE<br>(DØØE) | CALL 3Ø86<br>(CALL-12274) | **CLEAR — Clears the screen.** |

This subroutine clears the High-Resolution screen without resetting the High-Resolution Graphics mode.

| | | |
|---|---|---|
| ØC7C<br>(DØ7C) | CALL 387Ø<br>(CALL-1158Ø) | **PLOT — Plots a point on the screen.** |

This subroutine plots a single point on the screen.

| | Plot/Posn Location | Line Location | |
|---|---|---|---|
| Y (vertical) Coordinate | 8Ø2 Decimal<br>A - register | 8Ø2.<br>Y - register | Range Ø Top of Screen<br>159 Bottom of Screen |
| X (horizontal) Coordinate | 8ØØ Decimal<br>8Ø1 Decimal<br>X(X LO) Register<br>Y(Y HI) Register | 8ØØ.<br>8Ø1.<br>A(X LO)<br>X(X HI) | Range Ø (left) to<br>279 (right)<br>Loc 8ØØ=Xmod 256<br>8Ø1=X/256 |
| Color | 812<br>Ø32C Hex | | 4 Colors<br>Ø ($Ø) Black<br>85 ($55) Green<br>17Ø ($AA) Violet<br>255 ($FF) White |

| | | |
|---|---|---|
| ØC26<br>(DØ26) | CALL 3761<br>(CALL-11599) | **POSN — Positions a point on the screen.** |

This subroutine does all calculations for a PLOT, but does not plot a point (it leaves the screen unchanged). This is useful when used in conjunction with LINE or SHAPE (described later). To use this subroutine set up the X and Y coordinates just the same as for PLOT. The color in location 812 ($32C) is ignored.

| | | |
|---|---|---|
| ØC95 | CALL 3786 | **LINE — Draw a line on the screen.** |
| (DØ95) | (CALL-11574) | |

This subroutine draws a line from the last point PLOTed or POSN'ed to the other endpoint which is passed in the same manner as for a PLOT or POSN (8ØØ, 8Ø1, 8Ø2). The color of the line is set in location 812 ($32C). After the line is drawn, the new endpoint becomes the base endpoint for the next line drawn.

| | | |
|---|---|---|
| ØDBC | CALL 38Ø5 | **SHAPE — Draws a predefined shape on the screen.** |
| (D1BC) | (CALL-11555) | |

This subroutine draws a predefined shape on the screen at the point previously PLOTed or POSN'ed (8ØØ, 8Ø1, 8Ø2). The shape is defined by a table of vectors in memory.

| Hex | Decimal | Description |
|---|---|---|
| Ø324 | 8Ø4 (LO) | X mod 256 } Starting Address of Shape |
| Ø325 | 8Ø5 (HI) | X/256 } Table |
| ØØ1C | 28 | Color of Shape (see plot description) |
| Ø326 | 8Ø6 | Scaling Factor (1 = same size 2 = twice size, etc.) |
| Ø327 | 8Ø7 | Rotation Factor Ø to 64 (Ø = right side up 16 = 90° clockwise). |

**NOTES:**

# DISK COMMANDS

**[ ] indicated not needed, if default values are OK.**
**See abbreviations and prefixes for definition of single letter abbreviations.**

### Initialize

| | |
|---|---|
| INIT f,v[,Ss][,Dd] | Initializes diskette for system of same memory size only - will not boot on system of larger memory size. |

### General Program Control

| | |
|---|---|
| LOAD f [,Ss][,Dd][,Vv] | Loads program of name f from disk. |
| SAVE f [,Ss][,Dd][,Vv] | Saves program of name f to disk. |
| DELETE f [,Ss][,Dd][,Vv] | Deletes program of name f from disk. |
| RENAME f,g[,Ss][,Dd][,Vv] | Renames program f to name g |
| CATALOG [,Ss][,Dd] | Displays the volume number of diskette and list of all files and file types. File types are:<br>I - Integer BASIC program<br>A - Applesoft BASIC program<br>T - Text file - written by WRITE or APPEND commands<br>B - Bit-for-bit image of memory locations (machine code) |
| VERIFY f[,Ss][,Dd][,Vv] | Performs check on disk data to make sure it is self-consistent — does not check disk against memory contents. |
| RUN f[,Ss][,Dd][,Vv] | Loads and runs program of name f. |
| CHAIN f [,Ss][,Dd][,Vv] | For Integer Basic Programs only — loads and runs program f, but does not clear variables from previous program. |
| EXEC f[,Rr][,Ss][,Dd][,Vv] | Command file for program execution as if entered from keyboard. |
| LOCK f[,Ss][,Dd][,Vv] | When executed on file f it protects the file from deletion or erasure (write protected). |
| UNLOCK f[,Ss][,Dd][,Vv] | Removes write protect from file f. |

### Machine Code

| | |
|---|---|
| BSAVE f,Aa,Lj [,Ss][,Dd][,Vv] | Saves machine code starting at address a for length j. |
| BLOAD f[,Aa][,Ss][,Dd][,Vv] | Loads machine code. |
| BRUN f[,Aa][,Ss][,Dd][,Vv] | Loads and runs machine code. |

## Files

| | |
|---|---|
| MAXIFILES n | Specific number of files that are active at one time (1 to 16) |
| OPEN f[,Lj][,Ss][,Dd][,Vv] | Allocates 600 byte buffer and prepares to write to beginning of file. |
| CLOSE [f][,Ss][,Dd][,Vv] | Deallocates buffer for file f or all files if f is not specified and closes file indicated or if unspecified, all files. |
| READ f[,Rr][,Bb][,Ss][,Dd][,Vv] | Causes following input and get statements to obtain data from file f rather than keyboard. |
| WRITE F[,Rr][,Bb][,Ss][,Dd][,Vv] | Causes following print statements to write to file f. |
| POSITION f[,Rr][,Ss][,Dd][,Vv] | Positions file pointer to indicated line and byte (relative, not absolute). |
| APPEND f[,Ss][,Dd][,Vv] | Opens file f and writes data after last data previously written. |
| NOMON [C][,I ][,O] | Prevents writing information specified from being displayed on video screen. |
| C - disk commands<br>I - information to disk<br>O - output from disk | |
| MON [C][,I ][,O] | Enables commands previously disabled with NOMON. |
| CTRL/D | Must be used to turn off read command |

## Control

| | |
|---|---|
| FP [,Ss][,Dd][,Vv] | Activates Applesoft ROM or loads Applesoft from disk. |
| INT | Transfers control to Integer Basic. |
| POKE 72, Ø<br>CALL -151 or CALL -151<br>48:Ø | Go to monitor. |
| | **NOTE:** After using DOS commands in monitor type *48:Ø to insure monitor will work correctly. |
| 3DØG | Go to Basic and preserve program. |
| 3D3G | Returns to Basic losing Basic Program and Applesoft if not in ROM. |
| CALL 976 | Activates DOS is in memory (crashes if not) |

**ROM/RAM APPLESOFT**

To change programs

RAM to ROM (to use APPLESOFT ROM Card)

```
LOAD f
CALL 54514
SAVE f
```

ROM to RAM (no APPLESOFT ROM Card)

```
LOAD f
CALL 3314
SAVE f
```

**Abbreviations**

| | |
|---|---|
| f | — File Name 1 |
| g | — File Name 2 |
| s | — Slot Ø to 7 |
| v | — Volume Ø to 254 |
| d | — Drive 1 or 2 |
| r | — Record # Ø to 32767 |
| b | — Byte # Ø to 32767 |
| j | — Length Specifier 1 to 32767 |
| n | — # of Files 1 to 16 |
| a | — Starting Address Decimal<br>$ precedes Hex Numbers |
| FP | — Applesoft |
| INT | — Integer Basic |

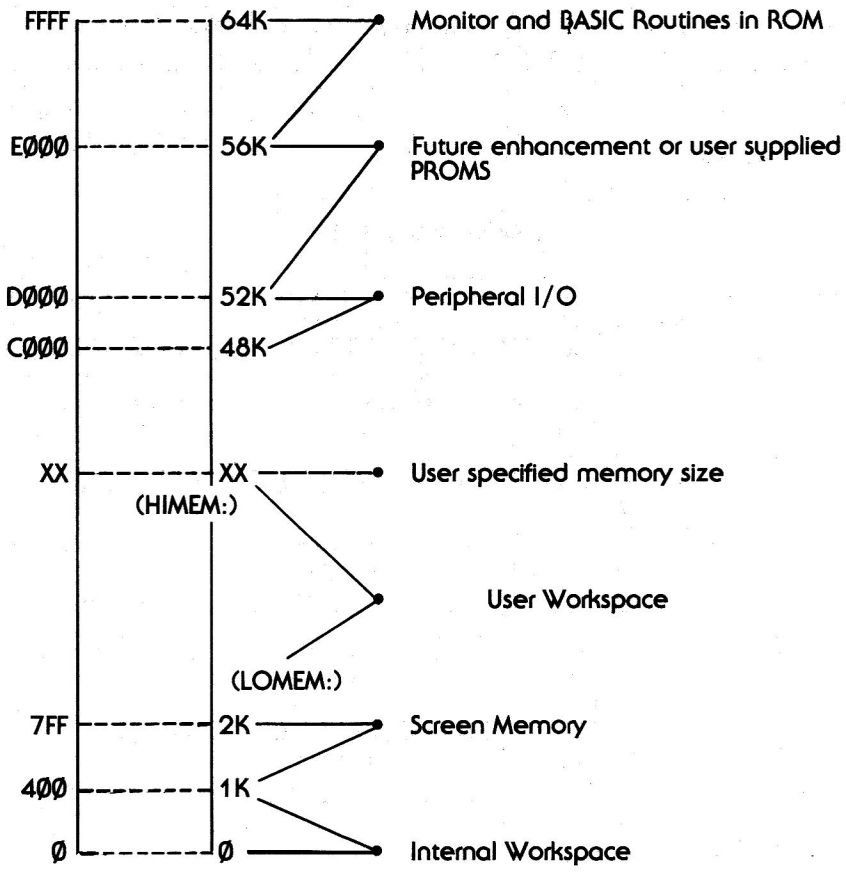
**Prefixes**

| | |
|---|---|
| A | — Address Start for Machine Code (decimal) |
| A$ | — Address Start for Machine Code (Hexadecimal) |
| L | — Record Length (Decimal) Bytes |
| L$ | — Record Length (Hexadecimal) Bytes |
| V | — Volume |
| S | — Slot **(Default last booted)** |
| D | — Drive **(Default last booted)** |
| R | — Record Number **(Default Ø)** |
| B | — Byte # **(After r-th CARRIAGE RETURN)** |
| CTRL D | — Precedes any disk function written in a program |
| [ ] | — Signifies Optional Specifiers with Prefixes |

# Memory Map

| Memory Range | Description |
|---|---|
| Ø.1FF | Program work space; not available to user. |
| 2ØØ.2FF | Keyboard character buffer. |
| 3ØØ.3FF | Available to user for short machine language programs. |
| 4ØØ.7FF | Screen display area for page 1 text or color graphics. |
| 8ØØ.2FFF | In cassette tape version, the APPLESOFT BASIC interpreter. |
| 8ØØ.XXX | If firmware APPLESOFT (Part number A2BØØØ9X) installed, user program and variable space, where XXX is maximum RAM memory to be used by APPLESOFT. This is either total system RAM memory, or less if the user is reserving part of high memory for machine language routines or high-resolution screen buffers. |
| 2ØØØ.3FFF | Firmware APPLESOFT only: high-resolution graphics display page 1. |
| 3ØØØ.XXX | Cassette tape APPLESOFT II; user program and variables where XXX is maximum available RAM memory to be used by APPLESOFT. This is either total system RAM memory, or less if the user is reserving part of high memory for machine language routines or page 2 high-resolution graphics. |
| 4ØØØ.5FFF | High-resolution graphics display page 2. |
| CØØØ.CFFF | Hardware I/O Addresses. |
| DØØØ.DFFF | Future ROM expansion. |
| DØØØ.F7FF | APPLESOFT II firmware version, with select switch "ON" (up). |
| EØØØ.F7FF | APPLE Integer BASIC. |
| F8ØØ.FFFF | APPLE System Monitor. |

## Simplified Memory Map



# APPLESOFT VARIABLE MAPS

## SIMPLE VARIABLES

| POINTERS | REAL | INTEGER | STRING POINTERS |
|---|---|---|---|
| $69-$6A | NAME (pos) 1st byte<br>(pos) 2nd byte | NAME (neg) 1st byte<br>(neg) 2nd byte | NAME (neg) 1st byte<br>(pos) 2nd byte |
| | exponent 1 byte<br>mantissa m.s. byte<br>mantissa<br>mantissa<br>mantissa l.s. byte | high byte<br>low byte<br>Ø<br>Ø<br>Ø | length 1 byte<br>address low byte<br>address high byte<br>Ø<br>Ø |

## ARRAY VARIABLES

| | REAL | INTEGER | STRING POINTERS |
|---|---|---|---|
| $6B-$6C | NAME (pos) 1st byte<br>(pos) 2nd byte | NAME (neg) 1st byte<br>(neg) 2nd byte | NAME (neg) 1st byte<br>(pos) 2nd byte |
| | OFFSET pointer to next variable: add to address of this variable name<br>low byte<br>high byte | OFFSET pointer to next variable: add to address of this variable name<br>low byte<br>high byte | OFFSET pointer to next variable: add to address of this variable name<br>low byte<br>high byte |
| | NO. OF DIMENSIONS<br>one byte | NO. OF DIMENSIONS<br>one byte | NO. OF DIMENSIONS<br>one byte |
| | SIZE Nth DIMENSION<br>high byte<br>low byte | SIZE Nth DIMENSTION<br>high byte<br>low byte | SIZE Nth DIMENSION<br>high byte<br>low byte |
| | | | |
| | SIZE 1st DIMENSION<br>high byte<br>low byte | SIZE 1st DIMENSION<br>high byte<br>low byte | SIZE 1st DIMENSION<br>high byte<br>low byte |
| | REAL (Ø, Ø, . . , Ø)<br>exponent 1 byte<br>mantissa m.s. byte<br>mantissa<br>mantissa<br>mantissa l.s. byte | INTEGER% (Ø, Ø, . . , Ø)<br>high byte<br>low byte | STRING$ (Ø, Ø, . . , Ø)<br>length 1 byte<br>address low byte<br>address high byte |
| | | | |
| | REAL (N, N, . . , N)<br>exponent 1 byte<br>mantissa m.s. byte<br>mantissa<br>mantissa | INTEGER% (N, N, . . , N)<br>high byte<br>low byte | STRING$ (N, N, . . , N)<br>length 1 byte<br>address low byte<br>address high byte |
| $6D-$6E | mantissa l.s. byte | | |

Strings are stored in order of entry, from HIMEM: down. String table points to first character of each string, at the bottom of the string in memory. As strings are changed, new pointing addresses are written; when available memory is used up, house-cleaning deletes all abandoned strings. (House-cleaning is forced by a FRE(X) ).

All arrays are stored with the right-most index ascending slowest; e.g., the numbers in the array A• (1, 1) where A• (Ø, Ø)=Ø, A• (1, Ø)=1, A• (Ø, 1)=2, A• (1, 1)=3 would be found in memory in proper sequence.

# ASCII CHARACTER CODES

DEC = ASCII decimal code
HEX = ASCII hexadecimal code
CHAR = ASCII character name
n/a = not accessible directly from the APPLE II keyboard
CTRL = depression of control key while striking character indicated.

| DEC | HEX | CHAR | WHAT TO TYPE | DEC | HEX | CHAR | WHAT TO TYPE |
|---|---|---|---|---|---|---|---|
| Ø | ØØ | NULL | CTRL @ | 48 | 3Ø | Ø | Ø |
| 1 | Ø1 | SOH | CTRL A | 49 | 31 | 1 | 1 |
| 2 | Ø2 | STX | CTRL B | 5Ø | 32 | 2 | 2 |
| 3 | Ø3 | ETX | CTRL C | 51 | 33 | 3 | 3 |
| 4 | Ø4 | ET | CTRL D | 52 | 34 | 4 | 4 |
| 5 | Ø5 | ENQ | CTRL E | 53 | 35 | 5 | 5 |
| 6 | Ø6 | ACK | CTRL F | 54 | 36 | 6 | 6 |
| 7 | Ø7 | BEL | CTRL G | 55 | 37 | 7 | 7 |
| 8 | Ø8 | BS | CTRL H or ← | 56 | 38 | 8 | 8 |
| 9 | Ø9 | HT | CTRL I | 57 | 39 | 9 | 9 |
| 1Ø | ØA | LF | CTRL J | 58 | 3A | : | : |
| 11 | ØB | VT | CTRL K | 59 | 3B | ; | ; |
| 12 | ØC | FF | CTRL L | 6Ø | 3C | < | < |
| 13 | ØD | CR | CTRL M or RETURN | 61 | 3D | = | = |
| 14 | ØE | SO | CTRL N | 62 | 3E | > | > |
| 15 | ØF | SI | CTRL O | 63 | 3F | ? | ? |
| 16 | 1Ø | DLE | CTRL P | 64 | 4Ø | @ | @ |
| 17 | 11 | DC1 | CTRL Q | 65 | 41 | A | A |
| 18 | 12 | DC2 | CTRL R | 66 | 42 | B | B |
| 19 | 13 | DC3 | CTRL S | 67 | 43 | C | C |
| 2Ø | 14 | DC4 | CTRL T | 68 | 44 | D | D |
| 21 | 15 | NAK | CTRL U or → | 69 | 45 | E | E |
| 22 | 16 | SYN | CTRL V | 7Ø | 46 | F | F |
| 23 | 17 | ETB | CTRL W | 71 | 47 | G | G |
| 24 | 18 | CAN | CTRL X | 72 | 48 | H | H |
| 25 | 19 | EM | CTRL Y | 73 | 49 | I | I |
| 26 | 1A | SUB | CTRL Z | 74 | 4A | J | J |
| 27 | 1B | ESCAPE | ESC | 75 | 4B | K | K |
| 28 | 1C | FS | n/a | 76 | 4C | L | L |
| 29 | 1D | GS | CTRL shift-M | 77 | 4D | M | M |
| 3Ø | 1E | RS | CTRL ^ | 78 | 4E | N | N |
| 31 | 1F | US | n/a | 79 | 4F | O | O |
| 32 | 2Ø | SPACE | space | 8Ø | 5Ø | P | P |
| 33 | 21 | ! | ! | 81 | 51 | Q | Q |
| 34 | 22 | " | " | 82 | 52 | R | R |
| 35 | 23 | # | # | 83 | 53 | S | S |
| 36 | 24 | $ | $ | 84 | 54 | T | T |
| 37 | 25 | % | % | 85 | 55 | U | U |
| 38 | 26 | & | & | 86 | 56 | V | V |
| 39 | 27 | ' | ' | 87 | 57 | W | W |
| 4Ø | 28 | ( | ( | 88 | 58 | X | X |
| 41 | 29 | ) | ) | 89 | 59 | Y | Y |
| 42 | 2A | * | * | 9Ø | 5A | Z | Z |
| 43 | 2B | + | + | 91 | 5B | [ | n/a |
| 44 | 2C | , | , | 92 | 5C | \ | n/a |
| 45 | 2D | — | — | 93 | 5D | ] | ] (shift-M) |
| 46 | 2E | . | . | 94 | 5E | ^ | ^ |
| 47 | 2F | / | / | 95 | 5F | _ | n/a |

# Decimal Tokens for Keywords

## XDRAW not currently used in Applesoft commands

| Hex | Decimal Token | Keyword | Hex | Decimal Token | Keyword |
|---|---|---|---|---|---|
| 8Ø | 128 | END | AA | 17Ø | LET |
| 81 | 129 | FOR | AB | 171 | GOTO |
| 82 | 13Ø | NEXT | AC | 172 | RUN |
| 83 | 131 | DATA | AD | 173 | IF |
| 84 | 132 | INPUT | AE | 174 | RESTORE |
| 85 | 133 | DEL | AF | 175 | & |
| 86 | 134 | DIM | BØ | 176 | GOSUB |
| 87 | 135 | READ | B1 | 177 | RETURN |
| 88 | 136 | GR | B2 | 178 | REM |
| 89 | 137 | TEXT | B3 | 179 | STOP |
| 8A | 138 | PR# | B4 | 18Ø | ON |
| 8B | 139 | IN# | B5 | 181 | WAIT |
| 8C | 14Ø | CALL | B6 | 182 | LOAD |
| 8D | 141 | PLOT | B7 | 183 | SAVE |
| 8E | 142 | HLIN | B8 | 184 | DEF |
| 8F | 143 | VLIN | B9 | 185 | POKE |
| 9Ø | 144 | HGR 2 | BA | 186 | PRINT |
| 91 | 145 | HGR | BB | 187 | CONT |
| 92 | 146 | HCOLOR= | BC | 188 | LIST |
| 93 | 147 | HPLOT | BD | 189 | CLEAR |
| 94 | 148 | DRAW | BE | 19Ø | GET |
| 95 | 149 | XDRAW - (n/a) | BF | 191 | NEW |
| 96 | 15Ø | HTAB | CØ | 192 | TAB( |
| 97 | 151 | HOME | C1 | 193 | TO |
| 98 | 152 | ROT= | C2 | 194 | FN |
| 99 | 153 | SCALE= | C3 | 195 | SPC( |
| 9A | 154 | SHLOAD | C4 | 196 | THEN |
| 9B | 155 | TRACE | C5 | 197 | AT |
| 9C | 156 | NOTRACE | C6 | 198 | NOT |
| 9D | 157 | NORMAL | C7 | 199 | STEP |
| 9E | 158 | INVERSE | C8 | 2ØØ | + |
| 9F | 159 | FLASH | C9 | 2Ø1 | — |
| AØ | 16Ø | COLOR= | CA | 2Ø2 | * |
| A1 | 161 | POP | CB | 2Ø3 | / |
| A2 | 162 | VTAB | CC | 2Ø4 | ^ |
| A3 | 163 | HIMEN: | CD | 2Ø5 | AND |
| A4 | 164 | LOMEN: | CE | 2Ø6 | OR |
| A5 | 165 | ONERR | CF | 2Ø7 | > |
| A6 | 166 | RESUME | DØ | 2Ø8 | = |
| A7 | 167 | RECALL | D1 | 2Ø9 | < |
| A8 | 168 | STORE | D2 | 21Ø | SGN |
| A9 | 169 | SPEED= | D3 | 211 | INT |

| Hex | Decimal Token | Keyword |
|---|---|---|
| D4 | 212 | ABS |
| D5 | 213 | USR |
| D6 | 214 | FRE |
| D7 | 215 | SCRN( |
| D8 | 216 | PDL |
| D9 | 217 | POS |
| DA | 218 | SQR |
| DB | 219 | RND |
| DC | 22Ø | LOG |
| DD | 221 | EXP |
| DE | 222 | COS |
| DF | 223 | SIN |
| EØ | 224 | TAN |
| E1 | 225 | ATN |
| E2 | 226 | PEEK |
| E3 | 227 | LEN |
| E4 | 228 | STR$ |
| E5 | 229 | VAL |
| E6 | 23Ø | ASC |
| E7 | 231 | CHR$ |
| E8 | 232 | LEFT$ |
| E9 | 233 | RIGHT$ |
| EA | 234 | MID$ |

**NOTES:**

# ERROR CODES

Error detected if PEEK(216) 127
POKE 216,Ø — resets ERRFLG
Y=PEEK(212) — Y is error code returned in decimal

## COMMANDS RELATED TO ERRORS

34Ø X = PEEK (218) + PEEK (219) * 256

This statement sets X equal to the line number of the statement where an error occurred if an ONNERGOTO statement has been executed.

| Hex | Decimal Value | Error Type Encountered |
|---|---|---|
| Ø | Ø | NEXT without FOR |
| 1Ø | 16 | Syntax |
| 16 | 22 | RETURN without GOSUB |
| 2A | 42 | Out of DATA |
| 35 | 53 | Illegal Quantity |
| 45 | 69 | Overflow |
| 4Ø | 77 | Out of Memory |
| 5A | 9Ø | Undefined Statement |
| 6B | 1Ø7 | Bad Subscript |
| 78 | 12Ø | Redimensioned Array |
| 85 | 133 | Division by Zero |
| A3 | 163 | Type Mismatch |
| BØ | 176 | String Too Long |
| BF | 191 | Formula Too Complex |
| EØ | 224 | Undefined Function |
| FE | 254 | Bad Response to an INPUT Statement |
| FF | 255 | Ctrl C Interrupt Attempted |

```
POKE 768, 1Ø4 : POKE 769, 168 : POKE 77Ø, 1Ø4 : POKE 771, 166:
POKE 772, 223 : POKE 773, 154 : POKE 774, 72  : POKE 775, 152 :
POKE 776, 72  : POKE 777, 96
```

Establishes a machine-language subroutine at location 768, which can be used in an error-handling routine. Clears up some ONERR GOTO problems with PRINT and ?OUT OF MEMORY ERROR messages. Use the command CALL 768 in the error-handling routine.

## DISK ERROR MESSAGES

### Error Code At Decimal 222 (Hex DE)

| Hex | Decimal Value | Error Messages |
|---|---|---|
| Ø4 | 4 | Write Protect |
| Ø5 | 5 | End of Data |
| Ø6 | 6 | File Not Found |
| Ø7 | 7 | Volume Mismatch |
| Ø8 | 8 | DISK I/O |
| Ø9 | 9 | Disk Full |
| ØA | 1Ø | File Locked |
| ØB | 11 | CMD Syntax |
| ØC | 12 | No File Buffs Available |
| ØD | 13 | Not Basic Program |
| ØE | 14 | Program Too Large |
| ØF | 15 | Not Binary File Error |

**NOTES:**

# Applesoft Zero Page Usage

| Location(s) (in hex) | Use |
|---|---|
| $Ø-$5 | Jump instructions to continue in APPLESOFT. (reset ØG return for APPLESOFT is equivalent to reset ctrl C return for Integer BASIC.) |
| $A-$C | Location for USR function's jump instruction. |
| $D-$17 | General purpose counters/flags for APPLESOFT. |
| $2Ø-$4F | APPLE II system monitor reserved locations. |
| $5Ø-$61 | General purpose pointers for APPLESOFT. |
| $62-$66 | Result of last multiply/divide. |
| $67-$68 | Pointer to beginnig of program. Normally set to $Ø8Ø1 for ROM version, or $3ØØ1 for RAM (cassette tape) version. |
| $69-$6A | Pointer to start of simple variable space. Also points to the end of the program plus 1 or 2, unless changed with the LOMEM: statement. |
| $6B-$6A | Pointer to beginning of array space. |
| $6D-$6E | Pointer to end of numeric storage in use. |
| $6F-$7Ø | Pointer to start of string storage. Strings are stored from here to the end of memory. |
| $71-$72 | General pointer. |
| $73-$74 | Highest location in memory available to APPLESOFT plus one. Upon initial entry to APPLESOFT, is set to the highest RAM memory location available. |
| $75-$76 | Current line number of line being executed. |
| $77-$78 | "Old line number". Set up by a ctrl C, STOP or END statement. Gives line number at which execution was interrupted. |
| $79-$7A | "Old text pointer". Points to location in memory for statement to be executed next. |
| $7B-$7C | Current line number from which DATA is being READ. |
| $7D-$7E | Points to absolute location in memory from which DATA is being READ. |
| $7F-$8Ø | Pointer to current source of INPUT. Set to $2Ø1 during an INPUT statement. During a READ statement is set to the DATA in the program it is READing from. |
| $81-$82 | Holds the last-used variable's name. |
| $83-$84 | Pointer to the last-used variable's value. |
| $85-$9C | General usage. |
| $9D-$A3 | Main floating point accumulator. |
| $A4 | General use in floating point math routines. |
| $A5-$AB | Secondary floating point accumulator. |

| | |
|---|---|
| $AC-$AE | General usage flags/pointers. |
| $AF-BØ | Pointer to end of program (**not** changed by LOMEM:) |
| $B1-$C8 | CHRGET routine. APPLESOFT calls here everytime it wants another character. |
| $B8-$B9 | Pointer to last character obtained through the CHRGET routine. |
| $C9-$CD | Random number. |
| $DØ-$D5 | High-resolution graphics scratch pointers. |
| $D8-$DF | ONERR pointers/scratch. |
| $EØ-$E2 | High-resolution graphics X and Y coordinates. |
| $E4 | High-resolution graphics color byte. |
| $E5-$E7 | General use for high-resolution graphics. |
| $E8-$E9 | Pointer to beginning of shape table. |
| $EA | Collision counter for high-resolution graphics. |
| $FØ-$F3 | General use flags. |
| $F4-$F8 | ONERR pointers. |

**NOTES:**

# Reserved Words in Applesoft & Integer Basic

[1] Integer Basic Only

[2] Applesoft Basic Only

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| &[2] | | | | | | | |
| ABS[1] | AND[2] | ASC[1] | AT[1] | ATN[1] | AUTO[1] | | |
| CALL[1] | CHR$[2] | CLEAR[2] | CLR[1] | COLOR [1] | CON[1] | CONT[2] | COS[1] |
| DATA[2] | DEF[2] | DEL[1] | DIM[1] | DRAW[2] | DSP[1] | | |
| END[1] | EXP[2] | | | | | | |
| FLASH[2] | FN[2] | FOR[1] | FRE[2] | | | | |
| GET[2] | GOSUB[1] | GOTO[1] | GR[1] | | | | |
| HCOLOR[2] | HGR[2] | HGR2[2] | HIMEM:[1] | HLIN[1] | HOME[2] | HPLOT[2] | HTAB[2] |
| IF[1] | IN#[1] | INPUT[1] | INT[2] | INVERSE[2] | | | |
| LEFT$[2] | LEN[1] | LET[1] | LIST[1] | LOAD[1] | LOG[1] | LOMEM:[1] | |
| MID$[2] | MAN[1] | | | | | | |
| NEW[1] | NEXT[1] | NORMAL[2] | NOT[2] | NOTRACE[1] | NO DSP[1] | | |
| ON[2] | ONERR[2] | OR[2] | | | | | |
| PDL[1] | PEEK[1] | PLOT[1] | POKE[1] | POP[1] | POS[2] | PRINT[1] | PR#[1] |
| READ[2] | RECALL[2] | REM[1] | RESTORE[2] | RESUME[2] | RETURN[1] | RIGHT$[2] | |
| RND[1] | ROT=[2] | RUN[1] | | | | | |
| SAVE[1] | SCALE=[2] | SCRN([1] | SGN[1] | SHLOAD[2] | SIN[1] | SPC([1] | |
| | SPEED=[2] | SQR[2] | STEP[1] | STOP[2] | STORE[2] | STR$[2] | |
| TAB([1] | TAN[2] | TEXT[1] | THEN[1] | TO[1] | TRACE[1] | | |
| USR[2] | | | | | | | |
| VAL[2] | VLIN[1] | VTAB[1] | | | | | |
| WAIT[2] | | | | | | | |
| XPLOT[2] | XDRAW[2] | | | | | | |

APPLESOFT "tokenizes" these reserved words: each word takes up only one byte of program storage. All other characters in program storage use up only one byte of program storage each.

## NOTE

The ampersand (&) is intended for the computer's internal use only; it is not a proper APPLESOFT command. This symbol, when executed as an instruction, causes an unconditional jump to location $3F5. Use reset ctrl C return to recover.

XPLOT is a reserved word that does not correspond to a current APPLESOFT command.

**See next page for special cases of reserved words.**

## Reserved words recognized by APPLESOFT only in certain contexts.

**COLOR, HCOLOR, SCALE, SPEED, and ROT**

parse as reserved words only if the next non-space character is the replacement sign, =. This is of little benefit in the case of COLOR and HCOLOR, as the included reserved word OR prevents their use in variable names anyway.

**SCRN, SPC and TAB**

parse as reserved words only if the next non-space character is a left parenthesis, (.

**HIMEM:**

must have its colon ( : ) to be parsed as a reserved word.

**LOMEM:**

also requires a colon ( : ) if it is to be parsed as a reserved word.

**ATN**

is parsed as reserved word only if there is no space between the T and the N. If a space occurs between the T and the N, the reserved word AT is parsed, instead of ATN.

**TO**

is parsed as a reserved word **unless** preceded by an A **and** there is a space betweenthe T and the O. If a space occurs between the T and the O, the reserved word AT is parsed instead of TO.

Sometimes parentheses can be used to get around reserved words:

```
                100 FOR A = LOFT OR CAT TO 15
LISTs as        100 FOR A = LOF TO RC AT TO 15
but             100 FOR A = (LOFT) OR (CAT) TO 15
LISTs as        100 FOR A = (LOFT) OR (C  AT  ) TO 15
```

## NOTES:

## APPLESOFT ROM CARD

### Applesoft/Integer Movement

[RESET] CØ8Ø [RETURN] CTRL/B [RETURN] — To initialize APPLESOFT.

[RESET] CØ81 [RETURN] CTRL/B [RETURN] — To initialize Integer Basic.

## ALGEBRAIC OPERATORS

| | |
|---|---|
| = | Assigns value to variable (LET optional) |
| — | Negation |
| ^ | Exponentiation |
| * | Multiplication |
| / | Division |
| + | Addition |
| — | Subtraction |

## RELATIONAL AND LOGICAL OPERATORS

| | |
|---|---|
| = | Equal |
| <> | Not equal |
| < | Less than |
| > | Greater than |
| <= | Less than or equal |
| >= | Greater than or equal |

| | |
|---|---|
| NOT | Logical "Not" |
| AND | Logical "And" |
| OR | Logical "Or" |

Relational and logical expresions have value 1 if true, Ø is false. Relational operators can also be used to compare strings.

**NOTES:**

## SYSTEM AND UTILITY COMMANDS

| Keyword | Definition |
|---|---|
| LOAD | Loads a program from tape. |
| LOAD f | Loads a program f from disk. |
| SAVE | Saves a program on tape. |
| SAVE f | Saves a program f to disk. |
| NEW | Deletes current program. |
| RUN | Executes program starting at lowest line number. |
| RUN 477 | Executes program starting at line 477. |
| RUN f | Loads and runs a program f from disk. |
| STOP | Halts execution and tells in which line. |
| END | Halts execution with no message. |
| CTRL C | Used in immediate mode to halt program or listing. |
| (reset) | Unconditional jump to Monitor. Use ctrl C or ØG to return to APPLESOFT. If in DOS use 3DØG. |
| CONT | Continues program execution stopped by STOP, END or ctrl C. |
| TRACE | Debugging aid; lists each line number as it is executed. |
| NOTRACE | Turns off TRACE. |
| PEEK(X) | Returns contents of memory location X. |
| POKE X,13 | Changes contents of memory location X to the value 13. |
| WAIT X,Y,Z | Waits until contents of location X, when XORed with Z and ANDed with Y, gives non-zero result. |
| CALL X | Goes to machine-language subroutine beginning at memory location X. |
| USR (X) | Passes value X to a machine-language subroutine. |
| HIMEM: | Sets highest memory address available to APPLESOFT program use. |
| LOMEM: | Sets lowest memory address available to APPLESOFT program use. (Cannot be changed without destroying variables.) |

## EDITING AND FORMAT-RELATED COMMANDS

| | |
|---|---|
| LIST | Lists entire program. |
| LIST X-Y | Lists from line X to line Y. |
| DEL X,Y | Deletes from line X to line Y. |
| REM | For writing program comments; ignored by program. |
| VTAB Y | Moves cursor to line Y (1 to 24). - Vertical Position |
| HTAB X | Moves cursor to position X (1 to 4Ø). - Horizontal Position |
| TAB(X) | Used only with PRINT statement; moves cursor to position X (1 to 4Ø). |
| POS(Ø) | Returns current horizontal position of cursor (Ø to 39). |
| SPC(X) | Used only with PRINT statement; puts X spaces between last item printed and next. |
| HOME | Clears screen and puts cursor at top left corner. |
| CLEAR | Resets all variables to zero. |

| | |
|---|---|
| FRE(Ø) | Returns amount of memory still available to user. |
| FLASH | Sets computer output to flashing. |
| INVERSE | Sets computer output to black on white. (reversed) |
| NORMAL | Turns off flashing or inverse output. |
| SPEED=X | Sets character output rate (Ø to 255). |
| ESC A | Moves cursor one space right. (does not copy text) |
| ESC B | Moves cursor one space left. (does not copy text) |
| ESC C | Moves cursor one space down. (does not copy text) |
| ESC D | Moves cursor one space up. (does not copy text) |
| right-arrow | Enters character under cursor into memory, and moves cursor one space right. |
| left-arrow | Deletes one character from line being typed, and moves cursor one space left. (also CTRL H) |
| CTRL X | Cancels line currently being typed. |

## ARRAYS AND STRINGS

| | |
|---|---|
| DIM A(X,Y,Z) | Sets maximum subscripts for A; reserves memory space for X+1 * Y+1 * Z+1 real elements, starting with A(Ø,Ø,Ø). |
| DIM A$(X,Y) | Sets maximum subscripts for A$, which may contain X+1 *Y+1 strings elements, each of up to 255 characters. |
| LEN(A$) | Returns number of characters in A$. |
| STR$(X) | Returns numeric value of X, converted to a string. |
| VAL(A$) | Returns A$, up to the first non-numeric character, as a numeric value. |
| CHR$(X) | Returns ASCII character whose code is X. |
| ASC(A$) | Returns ASCII code for first character of A$. |
| LEFT$(A$,X) | Returns leftmost X characters of A$. |
| RIGHT$(A$,X) | Returns rightmost X characters of A$. |
| MID$(A$,X,Y) | Returns Y characters of A$, starting at character X. |
| + | Operator used to concatenate strings. |
| STORE A | Saves numeric array A on tape. Cannot be used to save string arrays, directly. |
| RECALL B | Loads array back from tape; array B must have been DIMensioned correctly. |

## INPUT/OUTPUT COMMANDS

**(Also see LOAD and SAVE, STORE and RECALL.)**

| | |
|---|---|
| INPUT A$ | Puts ? on screen; waits for user to type a string value for A$. (also used to input data from disk - see Disc Commands section) |
| INPUT "XYZ";A | Prints XYZ on screen; waits for user to type a real number value for A. |
| GET A$ | Waits for user to type a one-character value for A$; does not need RETURN key. (also used to input data from disk - see Disk Commands section) |

| | |
|---|---|
| DATA X,"Y",Z | Establishes list of data elements that can be used by READ statements. (X and Z represent numeric variables, Y represents literal value) |
| READ A$ | Assigns next DATA element to A$. (also used before input and get commands for disk operations) |
| RESTORE | Resets data pointer to first element of first data statement. Read statement will now begin with first data element. |
| PRINT "X= ";X | Prints string X= and value of variable X on screen. Semicolons concatenate printed items, commas separate items into three tab fields. The symbol ? also means PRINT. (there are no leading or following spaces for numbers, these must be added in programming) |
| IN#6 | Takes future input from peripheral device in slot#6, instead of from keyboard (IN#Ø). |
| PR#6 | Sends output to peripheral device in slot#6, instead of to TV screen (PR#Ø). |
| LET X=Y | LET is optional; assigns value of Y to variable X. |
| DEF FN A(X)= X+25/X | Defines a function FNA. In later use, the argument of FNA will be substituted for X in the defined expression. FNA(4) would return 9.75. |

**NOTES:**

## COMMANDS RELATING TO FLOW OF CONTROL

| | |
|---|---|
| GOTO 347 | Branches to line 347. |
| IF X=3 THEN STOP | If the assertion X=3 is true (non-zero), then execution continues. If the assertion is false (zero), then execution jumps to the next numbered line. |
| FOR X=1 TO 2Ø STEP 4: NEXT X | Executes all statements between the FOR statement and the corresponding NEXT, first with X=1, then with X=5, X=9, etc. until X>2Ø, when execution continues after NEXT. STEP size is 1 if not specified. |
| NEXT X | Defines bottom of FOR . . . NEXT loop. The X is optional. |
| GOSUB 33 | Branches to the subroutine at line 33. |
| RETURN | Marks end of subroutine; returns to statement following most recent GOSUB. (used if there is no return for a GOSUB |
| POP | Removes one address from RETURN the address stack. |
| ON X GOTO 397, 12, 458 | Branches to the Xth line number in the list. If X=2, goes to line 12. |
| ON X GOSUB 397, 12, 458 | Branches to subroutine at the Xth line number in the list. |
| ONERR GOTO 45ØØ RESUME | Subsequent errors cause branch to error-handling routine at line 45ØØ instead of message and program halt. |
| RESUME | In error-handling routine, causes return to statement where error occurred. |

## GRAPHICS AND GAME CONTROLS

### Low-Resolution Graphics

| | |
|---|---|
| GR | Sets low-resolution graphics; clears top 4Ø x 4Ø area to black; bottom 4 lines text. |
| COLOR=X | Sets color (Ø to 15) for next plotting. (see section for colors) |
| PLOT X,Y | Places colored dot at horizontal coordinate X and vertical coordinate Y. X and Y are from Ø to 39. Ø,Ø is top left. |
| LIN X1,Y2 AT Y | Draws horizontal line from the point at X1,Y to the point at X2,Y. |
| VLIN Y1,Y2 AT X | Draws vertical line from the point at X,Y1 to the point at X,Y2. |
| SCRN(X,Y) | Returns color on screen at the point X,Y. (X=horizontal Y=vertical) |

## High-Resolution Graphics

| | |
|---|---|
| HGR | Sets high-resolution graphics, page 1; clears top 28Ø x 16Ø area to black; bottom 4 lines text. |
| HGR 2 | Sets high-resolution graphics, page 2; clears entire 28Ø x 192 screen to black. |
| HCOLOR=X | Sets color (Ø to 7) for next plotting. (see section for colors) |
| HPLOT X,Y | Places colored dot at horizontal coordinate X and vertical coordinate Y. X is from Ø to 279; Y is from Ø to 159 (HGR) or to 191 (HGR2). Ø,Ø is top left corner. |
| HPLOT X1,Y1 | Draws line from the point at X1,Y1 to the point at X2,Y2. Com- |
| TO X2,Y2 | mand may be extended to additional points . . .TO XN, YN. |
| SHLOAD | Loads a shape table from tape. (loads just below HIMEM: and HIMEM: is changed to just below shape table) |
| DRAW 3 AT X,Y | Draws shape definition #3 from a previously loaded shape table, starting at the point X,Y in color set by HCOLOR. |
| XDRAW 3 AT X,Y | Draws shape definition #3 from shape table; color of each point plotted is complement of color on screen at that point. |
| ROT=X | Sets rotation of shape for DRAW or XDRAW. ROT=Ø is vertical, ROT=16 is 9Ø degrees clockwise, ROT=32 is 18Ø degrees clockwise, etc. (if SCALE=1 only 4 90° rotations are possible) |
| SCALE=X | SEts scale (1 to 255) of shape for DRAW or XDRAW. |

## Game Controls

| | |
|---|---|
| PDL (X) | Returns setting from Ø to 255 of game control X (Ø to 3). |
| PEEK(X-16287) | If >127, button on game control X (Ø to 2) is being pressed. |
| PEEK(-16336) | "Clicks" APPLE's speaker. (1 click each PEEK) |

**NOTES:**

## SOME MATH FUNCTIONS

| | |
|---|---|
| SIN(X) | Returns sine of X radians. |
| COS(X) | Returns cosine of X radians. |
| TAN(X) | Returns tangent of X radians. |
| ATN(X) | Returns arctangent, in radians, of X. |
| INT(X) | Returns largest integer less than or equal to X. |
| RND(1) | Returns random real number from Ø to Ø.999999999 each time used. |
| RND(Ø) | Returns last random number again. |
| RND(-3) | Returns 4.48217179E-Ø8. A different fixed number is returned for each different negative argument. After this, RND with positive argument will follow a fixed sequence. |
| SGN(X) | Returns -1 if X< Ø, Ø if X=Ø, and 1 if X> Ø. |
| ABS(X) | Returns absolute value of X. |
| SQR(X) | Returns positive square root of X. |
| EXP(X) | Returns e (2.718289) to the power X. |
| LOG(X) | Returns natural logarithm of X. |

**NOTES:**

# Command Subroutines in the Cassette Version of APPLESOFT II

| COMMAND WORD | ADDRESS OF ASCII STRING | ADDRESS OF POINTER TO SUBROUTINE | ADDRESS OF SUBROUTINE | COMMAND WORD | ADDRESS OF ASCII STRING | ADDRESS OF POINTER TO SUBROUTINE | ADDRESS OF SUBROUTINE |
|---|---|---|---|---|---|---|---|
| END | 8DØ | 8ØØ | 1Ø72 | LOAD | 9BB | 86C | 1ØCB |
| FOR | 8D3 | 8Ø2 | F68 | SAVE | 9BF | 86E | 1ØB2 |
| NEXT | 8D6 | 8Ø4 | 14FC | DEF | 9C3 | 87Ø | 1BØC |
| DATA | 8DA | 8Ø6 | 1197 | POKE | 9C6 | 872 | 1F72 |
| INPUT | 8DE | 8Ø8 | 13B5 | PRINT | 9CA | 874 | 12D7 |
| DEL | 8E3 | 8ØA | 2B38 | CONT | 9CF | 876 | 1Ø98 |
| DIM | 8E6 | 8ØC | 17DC | LIST | 9D3 | 878 | EA7 |
| READ | 8E9 | 8ØE | 13E5 | CLEAR | 9D7 | 87A | E6A |
| GR | 8ED | 81Ø | 2B8C | GET | 9DC | 87C | 13A3 |
| TEXT | 8EF | 812 | 2B95 | NEW | 9DF | 87E | E49 |
| PR# | 8F3 | 814 | 29EC | TAB( | 9E2 | | |
| IN# | 8F6 | 816 | 29E5 | TO | 9E6 | | |
| CALL | 8F9 | 818 | 29DC | FN | 9E8 | | |
| PLOT | 8FD | 81A | 2A2C | SPC( | 9EA | | |
| HLIN | 9Ø1 | 81C | 2A39 | THEN | 9EE | | |
| VLIN | 9Ø5 | 81E | 2A48 | AT | 9F2 | | |
| HGR2 | 9Ø9 | 82Ø | 2BD4 | NOT | 9F4 | | |
| HGR | 9ØD | 822 | 2BDE | STEP | 9F7 | | |
| HCOLOR | 91Ø | 824 | 2EE5 | + | 9FB | 8B2 | 1FB8 |
| HPLOT | 916 | 826 | 2EFA | − | 9FC | 8B5 | 1FA1 |
| DRAW | 91B | 828 | 2F62 | * | 9FD | 8B8 | 2179 |
| XDRAW | 91F | 82A | 2F68 | / | 9FE | 8BB | 226Ø |
| HTAB | 924 | 82C | 2FEØ | ^ | 9FF | 8BE | 268E |
| HOME | 928 | 82E | FC58 | AND | AØØ | 8C1 | 1758 |
| ROT | 92C | 83Ø | 2F1A | OR | AØ3 | 8C4 | 1752 |
| SCALE | 93Ø | 832 | 2F2Ø | > | AØ5 | 8C7 | 26C7 |
| SHLOAD | 936 | 834 | 2F6E | = | AØ6 | 8CA | 169B |
| TRACE | 93C | 836 | 2A74 | < | AØ7 | 8CD | 1768 |
| NOTRACE | 941 | 838 | 2A76 | SGN | AØ8 | 88Ø | 2387 |
| NORMAL | 948 | 83A | 2A7A | INT | AØB | 882 | 241A |
| INVERSE | 94E | 83C | 2A7E | ABS | AØE | 884 | 23A6 |
| FLASH | 955 | 83E | 2A87 | USR | A11 | 886 | ØØØA |
| COLOR= | 95A | 84Ø | 2A56 | FRE | A14 | 888 | 1AD7 |
| POP | 96Ø | 842 | 116D | SCRN | A17 | 88A | C12 |
| VTAB | 963 | 844 | 2A5D | PDL | A1C | 88C | 17DØ |
| HIMEM: | 967 | 846 | 2A8D | POS | A1F | 88E | 1AF8 |
| LOMEM: | 96D | 848 | 2AAD | SQR | A22 | 89Ø | 2684 |
| ONERR | 973 | 84A | 2AD2 | RND | A25 | 892 | 27A5 |
| RESUME | 978 | 84C | 2B1F | LOG | A28 | 894 | 2138 |
| RECALL | 97E | 84E | 2BB8 | EXP | A2B | 896 | 27ØØ |
| STORE | 984 | 850 | 2B9B | COS | A2E | 898 | 27E1 |
| SPEED= | 989 | 852 | 2A69 | SIN | A31 | 89A | 27E8 |
| LET | 98F | 854 | 1248 | TAN | A34 | 89C | 2831 |
| GOTO | 992 | 856 | 1140 | ATAN | A37 | 89E | 2895 |
| RUN | 996 | 858 | 1114 | PEEK | A3A | 8AØ | 1F5B |
| IF | 999 | 85A | 11CB | LEN | A3E | 8A2 | 1ECD |
| RESTORE | 99B | 85C | 104B | STR$ | A41 | 8A4 | 1BBE |
| & | 9A2 | 85E | 8ØØ-see note | VAL | A45 | 8A6 | 1EFE |
| GOSUB | 9A3 | 86Ø | 1123 | ASC | A48 | 8A8 | 1EDC |
| RETURN | 9A8 | 862 | 116D | CHR$ | A4B | 8AA | 1E3D |
| REM | 9AE | 864 | 11DE | LEFT$ | A4F | 8AC | 1E51 |
| STOP | 9B1 | 866 | 1Ø7Ø | RIGHT$ | A54 | 8AE | 1E7D |
| ON | 9B5 | 868 | 11EE | MID$ | A5A | 8BØ | 1E88 |
| WAIT | 9B7 | 86A | 1F7B | | | | |

**NOTE:** The $ command is apparently there for later expansion and if you use it now your program will bomb!