

# A.P.F.L.E.

APPLE

PUGET  
SOUND

PROGRAM

LIBRARY

EXCHANGE

## PRESENTS

### ][**WORKSHOP DISKPAK**><

*by*

● VAL J. GOLDING ●  
DARRELL ALDRICH ● RON ALDRICH

## SUMMARY OF COMMANDS (Combined with Table of Contents)

Enter	Name of Routine or Command	Inputs Required	Functions Performed	Page
<b>A</b>	<b>APPEND</b>	File Name, C/R. Carriage Return	APPENDs disk program to Workshop. APPENDs tape program to Workshop	4
<b>B</b>	<b>BYTES</b>	nil	Displays BYTES free and used; HMEM: and LOMEM:, program and variable pointers.	4
<b>C</b>	<b>CATALOG</b>	nil	Displays current disk CATALOG	5
<b>D</b>	<b>DISK</b>	[ any disk command	Executes DISK command as input	5
<b>E</b>	<b>EXAM</b>	Basic line number to be EXAMined	Returns Basic line as hex bytes in Monitor format, along with hex address	5
<b>F</b>	<b>FORMAT</b>	[ P R	Displays PRINT statement format on screen Displays REM statement format on screen	6
<b>I</b>	<b>INTEGER</b>	< any Integer Basic Keyboard operation	Displays Integer Basic prompt in inverse video and allows execution of any Integer Basic operation from within the Workshop	6
<b>K</b>	<b>KILL</b>	Carriage Return	Hides Workshop so that in effect only the APPENDed program is resident in memory	7
<b>L</b>	<b>LIST</b>	1st line number to be LISTed. Carriage Return	Returns one screen page of program LISTings. Returns additional screen pages of LISTings	7
<b>M</b>	<b>MONITOR</b>	any MONITOR command	Displays * prompt in inverse video and executes MONITOR command and returns to the Workshop	7
<b>O</b>	<b>OPTIONS</b>	nil	Displays command (OPTIONS) list (menu)	7
<b>Q</b>	<b>QUIT</b>	nil	Exits Workshop and reenters Basic	8
<b>R</b>	<b>REN</b>	1st & last lines to be RENumbered; new 1st line number and increment	RENumbers Basic program line numbers and GOTO references as specified by input parameters	8
<b>S</b>	<b>SAVE</b>	File name, C/R  Carriage Return	SAVEs named program to disk then returns to the Workshop, leaving only the Workshop resident.  SAVEs program to tape without saving the Workshop, then returns to the Workshop, leaving only the Workshop resident.	10
<b>W</b>	<b>WRITE</b>	1st and last decimal locations to be copied; starting line number and increment	WRITEs machine language automatically into program memory in Monitor string format.	9
	<b>Hex-Dec</b>	Z + hex number	Returns decimal equivalent	10
	<b>Dec-Hex</b>	Decimal number	Returns hex equivalent	10
	<b>Variable List</b>	< GOTO 32767  [ GOTO 63300	Displays in order used, all variables and their values and names of string variables  Displays in order used, all simple variables, their values and addresses, plus string variables, their contents and addresses.	10

### SUBJECT

Overview .....	11
General Information .....	12
Copying .....	12
Modifying the Workshops .....	13
Using Applesoft Workshop without ROM card .....	14
Programmer's Workshop Version 9.13 (Tape Only) .....	14
History and Credits .....	15
Applesoft II Token Chart .....	16
Integer Basic Token Chart .....	Inside Back Cover

# **][ WORKSHOP DISKPAK ><**

**REQUIRES 32K RAM, DISK & FP FIRMWARE**

**September 1, 1979**

Copyright ©1979 by Apple Pugetsound Program Library Exchange  
6708 39th Ave. S.W. Seattle, Washington 98136 (206) 932-6588

## **<PROGRAMMER'S WORKSHOP V 2.5**

*by*

**Darrell Aldrich & Val Golding**

## **[ APPLESOFT WORKSHOP V 1.5**

*by*

**Ron Aldrich & Val Golding**





## SUMMARY OF COMMANDS (Combined with Table of Contents)

Enter	Name of Routine or Command	Inputs Required	Functions Performed	Page
<b>A</b>	<b>APPEND</b>	File Name, C/R. Carriage Return	APPENDs disk program to Workshop. APPENDs tape program to Workshop	4
<b>B</b>	<b>BYTES</b>	nil	Displays BYTES free and used; HIMEM; and LOMEM; program and variable pointers.	4
<b>C</b>	<b>CATALOG</b>	nil	Displays current disk CATALOG	5
<b>D</b>	<b>DISK</b>	[ any disk command	Executes DISK command as input	5
<b>E</b>	<b>EXAM</b>	Basic line number to be EXAMined	Returns Basic line as hex bytes in Monitor format, along with hex address	5
<b>F</b>	<b>FORMAT</b>	[ P R	Displays PRINT statement format on screen Displays REM statement format on screen	6
<b>I</b>	<b>INTEGER</b>	< any Integer Basic Keyboard operation	Displays Integer Basic prompt in inverse video and allows execution of any Integer Basic operation from within the Workshop	6
<b>K</b>	<b>KILL</b>	Carriage Return	Hides Workshop so that in effect only the APPENDED program is resident in memory	7
<b>L</b>	<b>LIST</b>	1st line number to be LISTed. Carriage Return	Returns one screen page of program LISTings. Returns additional screen pages of LISTings	7
<b>M</b>	<b>MONITOR</b>	any MONITOR command	Displays * prompt in inverse video and executes MONITOR command and returns to the Workshop	7
<b>O</b>	<b>OPTIONS</b>	nil	Displays command (OPTIONS) list (menu)	7
<b>Q</b>	<b>QUIT</b>	nil	Exits Workshop and reenters Basic	8
<b>R</b>	<b>REN</b>	1st & last lines to be RENumbered; new 1st line number and increment	RENNumbers Basic program line numbers and GOTO references as specified by input parameters	8
<b>S</b>	<b>SAVE</b>	File name, C/R  Carriage Return	SAVEs named program to disk then returns to the Workshop, leaving only the Workshop resident.  SAVEs program to tape without saving the Workshop, then returns to the Workshop, leaving only the Workshop resident.	10
<b>W</b>	<b>WRITE</b>	1st and last decimal locations to be copied; starting line number and increment	WRITEs machine language automatically into program memory in Monitor string format.	9
	<b>Hex-Dec</b>	Z + hex number	Returns decimal equivalent	10
	<b>Dec-Hex</b>	Decimal number	Returns hex equivalent	10
	<b>Variable List</b>	< GOTO 32767  [ GOTO 63300	Displays in order used, all variables and their values and names of string variables  Displays in order used, all simple variables, their values and addresses, plus string variables, their contents and addresses.	10
<b>SUBJECT</b>				
<b>Overview</b> .....				11
<b>General Information</b> .....				12
<b>Copying</b> .....				12
<b>Modifying the Workshops</b> .....				13
<b>Using Applesoft Workshop without ROM card</b> .....				14
<b>Programmer's Workshop Version 9.13 (Tape Only)</b> .....				14
<b>History and Credits</b> .....				15
<b>Applesoft II Token Chart</b> .....				16
<b>Integer Basic Token Chart</b> .....				Inside Back Cover

# APPEND

This routine combines two or more programs into one. It has a twofold purpose. The first is to APPEND the Workshop to an existing program so that the various Workshop routines may be utilized on the APPENDED program.

The second is to provide the capability of joining two or more programs to form a whole, as when desired to have a number of game programs user selectable from a single menu.

Care must be taken beforehand, utilizing the Workshop RENumber routine if needed, to ascertain that duplicate line numbers do not exist prior to executing the APPEND.

Programs to be APPENDED to the Workshop may not have line numbers > 32764 in Integer Basic and 62999 in Applesoft II Basic.

When executing multiple APPENDs, programs must be APPENDED in a specific order to assure that line numbers maintain their proper sequence. The program with the *highest* line numbers should be the program to be APPENDED first, for both Integer Basic and Applesoft II. (Applesoft actually does the opposite in a simple APPEND but is compensated for in this routine.

To execute the actual APPEND, the Workshop displays an APPEND prompt in the user. To APPEND from tape, start your recorder on play when disk stops, then hit return. When the APPEND is complete you will receive your Workshop prompt back.

To APPEND a program from disk, make sure the proper diskette is in the same drive. Enter the file name of the program to be APPENDED and hit return.

In the Programmer's Workshop, should an APPEND abort midway, it *may* be possible to recover the Workshop (and any previously APPENDED programs) by entering CALL 4045.

# BYTES

This command displays the current settings of LOMEM: and HIMEM:, BYTES of RAM memory available for programming and the number of BYTES used in the current APPENDED program, along with the values of the program and variable pointers.

Note that BYTES free represent the space *currently available* for programming, while BYTES used refer to the APPENDED program only, exclusive of the Workshop.

No user input is required.

# CATALOG

This command displays the current disk CATALOG.

No user input is required.

## DISK [

This is a limited use command contained in the Applesoft Workshop only. It permits the execution of DISK commands from within the workshop. An example of its use would be to enter "DELETE FILE", thus DELEting an unneeded program from the disk CATALOG.

## EXAM

Dual purposes are served here. First, EXAM serves as a learning tool for the inexperienced programmer by aiding the user in understanding the structure and tokenizing of a Basic statement.

Secondly, as a utility, and in conjunction with the MONITOR routine, EXAM offers the user the opportunity to EXAMine and subsequently modify Basic program lines, insert illegal tokens (i.e. HIMEMS), etc.

EXAM functions by prompting the user to input a Basic line number. EXAM will then display, in a fashion similar to what might be seen by actually entering the Apple Monitor, the hexadecimal address (memory location) and each hex byte contained in the Basic statement.

At this point, if a Basic statement is to be modified, the user would enter M (for MONITOR), trace the cursor over the bytes to be modified, make the correction and hit return.

This procedure is more suitable for Integer Basic than Applesoft, since Integer checks for correctness of syntax at the time the Basic line is originally entered, while Applesoft does not check for most syntax until RUN time, which essentially limits modifications in Applesoft to illegal line numbers and other minor changes.

It is also recommended that the user read "Apple II Basic Structure" by S. Wozniak in Peeking at Call-Apple, Vol. I, 1978 and "Applesoft From Bottom to Top" by Val Golding, in Call-Apple, March 1979. These two articles cover in considerable detail the tokenizing, structure and organization of the two respective Basic languages.

## FORMAT [

Contained in Applesoft Workshop only. FORMAT expects, either of two possible inputs, P (for PRINT) or R (for REM).

FORMAT will display a screen page of blank PRINT or REM statements, with underscores indicating where characters should be typed.

If the user avoids typing in the spaces, the result will be neatly formatted PRINT or REM statements with no broken words.

A statement "GOTO 63568" for the PRINT routine or "GOTO 63560" for the REM routine will be output at the bottom of the screen. By tracing over this line and hitting return, a new screen of blank statements will be displayed.

To reenter Workshop, type "RUN 63000."

## INTEGER <

A function of the Integer Workshop only, which allows execution of any Integer Basic operation from within the Workshop. For example, while LISTing a program, an error in one statement is detected. To make a correction, the user need only type I (for INTEGER) and then using the normal edit functions, move the cursor to the program line and make the correction. After hitting return, the user will be returned to the Workshop.

While in the INTEGER mode, an Integer Basic prompt (>) in inverse video will be displayed. The only user input required is entry of any valid Integer Basic command or statement.

# KILL

User input expected: carriage return.

This command resets pointers in Integer Basic, thus effectively hiding the Workshop from RAM memory, retaining only the APPENDED program. In Applesoft the Workshop is actually deleted and is not recoverable.

In Integer Basic, Workshop may be recovered with POKE 76, H MOD 256: POKE 77, H/256 where H = HIMEM: prior to execution of KILL.

# LIST

One of the most useful utilities in the Workshop, LIST prompts the user to input a starting line number to LIST from. LIST will then display one screen page of program lines and then return to the Workshop prompt.

At this time the user may hit return to display the next screen page of LISTings or go on to another Workshop routine. However, LIST will remember where it left off, and after executing other routines, all the user need do is hit return again to resume LISTing from the same place.

# MONITOR

A Workshop command which permits the execution of any MONITOR command from within the Workshop and which returns the user to the Workshop following a carriage return. While in MONITOR, a MONITOR prompt(\*) will be displayed in inverse video.

Any valid MONITOR function may be executed while in this mode, such as modifying memory or listing. If an illegal MONITOR command is input, the Workshop may ABEND (abort/end). Recovery of the Workshop (and any APPENDED program) may usually be made by typing E88AG (for Integer Basic) or D823G (for Applesoft) and a carriage return.

# OPTIONS

Displays the Workshop command list.

No user input is required.

# QUIT

Exits user from the Workshop. May also be enabled with the escape key. The Workshop is left intact in memory. After QUITing, the Workshop may be reentered with a RUN 32765 (for Integer) or RUN 63000 (for Applesoft).

No user inputs required.

# REN

Use this routine to open up space between program lines, number by tens or reorganize number groups. REN expects four inputs from the user: First and last line numbers to be RENumbered, new starting line number and step. Numbers may not be resequenced i.e., placed in a new order.

In the Integer Workshop, REN will print a list of old and new line numbers. Execution may take several seconds, as REN must make two passes, the second to change GOTO and other references. Calculated GOTOs ( $\text{GOTO } J * 1000$ ) will be ignored.

In the Applesoft Workshop, an error message header reading "LINE NBR REF SAYS SHOULD BE" will be displayed. If any line number references are changed incorrectly (and they will be whenever an additional digit is added to a reference) a message will be displayed under this heading. In a long program the content of the error messages should be recorded on paper.

Applesoft REN is quite slow and may take up to 15 minutes to RENumber a program of up to 16 K in length. It also requires a great deal of array space and users with 32 K and disk may find their memory insufficient. This may be overcome in three different ways.

1. In line 63950 the variable N9 is set to 1000. This variable represents the maximum number of lines to be RENumbered and DIMensions array space accordingly. Array space may be deallocated by reducing N9 to equal only the exact number of lines to be renumbered, plus a 10% error margin.
2. By SAVING both the Workshop and APPENDED program on tape and then resetting HIMEM: to maximum and reLOADing tape. When RENumbering is complete, reSAVE to tape, reboot DOS, reLOAD the tape and resume normal operations. This manouver will make available the 10.5 K used by DOS.
3. Lines 63052 to 63949 may be deleted from the Workshop without affecting the operation of REN. By so doing, you will gain approximately an additional 4K of memory. When RENumbering is complete, enter DEL 63000, 63999 and SAVE your program.

If you have the RENUM/MERGE program on the DOS 3.2 Master Diskette, this is a far superior RENUMber routine and we recommend its use on longer programs.

# SAVE

This handy routine allows the user to SAVE an APPENDED program to tape or disk without also saving the Workshop.

It expects as input the name of the program to be SAVED to disk, followed by a carriage return or, if SAVING to tape, a carriage return only, once the recorder is in record mode.

In operation, it first "KILLS" the Workshop, SAVES the APPENDED program to disk or tape, DELETES the APPENDED program, and finally, recovers the Workshop.

# WRITE

An automatic utility which asks the user to input four parameters: starting and ending memory addresses in decimal, starting program line number and increment.

The function of this utility is to provide a method of incorporating a machine language program within a Basic program. Once the user has met the input requirements, Workshop will locate the assembly program in memory, WRITE it to a text file on disk and then READ it from disk in a format that will be automatically be entered into program memory, where it then becomes part of the Basic program.

The form in which it has been entered into program memory is one which is winning general acceptance as a standard for "hiding" short assembly programs in Basic.

An example where it might be used is: you have a printer driver program that resides in memory at \$300 to \$3AF (768 to 943 decimal).

From the Workshop, just enter the correct parameters 768, 943 (as in the example above) and a starting program line number and step, and WRITE will do the rest. Use the workshop's built in hex to dec converter to convert your addresses for you.

By including the newly created lines in your Basic program, the machine language portion will run and locate itself correctly in memory each time the basic program is RUN.

# HEX-DEC CONVERTER

This is one of the two routines not contained on the Workshop OPTIONS list. It was so excluded in order to increase the ease and speed of operation.

At any time the user may input a decimal number and Workshop will display the hexadecimal equivalent. By first entering a "Z" followed by a hex number, Workshop will return the decimal equivalent.

Range limitations are as shown:

hex \$0 to \$FFFF

decimal -32767 to + 32767 (Integer Basic)

decimal 0 to 65535 (Applesoft Basic)

## VARIABLE LIST

This is the second routine that does not appear on the Workshop OPTIONS (command) list, inasmuch as it is designed to be accessed from the keyboard with a GOTO, after having first completed a RUN of an APPENDED program.

In Integer Basic, VARIABLE LIST will display in order of use all variables and their values and string variables. For access, type "GOTO 32767".

In Applesoft II Basic, VARIABLE LIST displays the following, also in order of use: 1. simple string variables, their contents and location. 2. simple integer, real and DEF FN variables, their values and addresses.

For access, type "GOTO 63300". Array variables are not included in this routine.

To use VARIABLE LIST properly you must first have (APPENDED) in memory, the program whose variables you wish to examine. This program should then be run so that its' variables will be established in memory. Then exit the program with a control C or other established method, and enter the appropriate GOTO (32767 for Integer, 63300 for Applesoft) and hit return.

It should be noted that along with the variables of the Applesoft program you are testing, a null D\$ will be included and should be disregarded. Variables values from the object program will not be preserved.



# OVERVIEW

The workshops were designed with two purposes in mind: 1. To aid the programmer in learning about how the BASIC languages are structured and tokenized (through the use of the EXAM routine and 2. To assist the programmer in modifying and writing programs using the many routines available.

To put either Workshop to practical use, a program must be APPENDED to the Workshop (unless one is starting a program from scratch.) To APPEND is to join two or more programs together to form a whole.

The WORKSHOP DISKPAK is composed of three programs. WORKSHOP DISKPAK is the name of the Hello program, which runs on boot up. It offers the user a choice of running either the PROGRAMMER'S WORKSHOP (in Integer Basic) or the APPLESOFT WORKSHOP. In this documentation, the terms Programmer's Workshop and Integer Workshop are used interchangeably.

When the Programmer's Workshop is RUN and the user has 48 K of memory, then HIMEM: is automatically set to 32767, since certain routines in this Workshop do not function properly with HIMEM: > 32767.

The two Workshops have been designed to resemble one another in appearance and operation.

However, there are three routines that do not appear in both Workshops, DISK, FORMAT and INTEGER. They will be flagged with the appropriate prompt character in this documentation.

All commands appearing on the OPTIONS list are executed by typing only the first letter of the command. In many cases the Workshop will prompt the user to input additional data as may be required for the routine selected.

Upon entering the Workshop, the user will be greeted by the WORKSHOP BASIC prompt. This character is the mirror image of the Basic prompt, either a | or a < , depending on the Workshop selected.

To conserve RAM memory for programming, very little in the way of prompting or instructions are contained within the programs themselves. The user is urged to study this documentation thoroughly in order to become familiar with the requirements and operation of the Workshops.

A convenient summary of the command OPTIONS, combined with a table of contents appears preceeding this section. It is followed by a more detailed discussion of each command, some general information, and finally a brief history of the Workshops and author credits.

# GENERAL INFORMATION

In the process of initializing and in some of the routines, both Workshops will create and DELETE certain disk files. If one of these routines is interrupted midway, you may find strange files on your disk. These include TEMP and X, which may be DELETED. Also included are H and WORKSHOP, which should not be DELETED as they are essential to the proper operation of the Workshops.

Each of the included programs have been tested and made as bug-free as possible. Any errors should be reported to us in writing so that corrective action may be taken if required. A.P.P.L.E. disclaims any liability to your programs or equipment through the use of any programs supplied on this diskette or cassette.

## COPYING

Both Workshops, before RUNing, have some 3K or so of additional program which, once RUN, is self DELeting. This additional code is initially required for the proper operation of the Workshop.

To make a copy of either Workshop from RAM memory, LOAD the program from disk and do *not* RUN. Then make a copy to tape or another diskette, as appropriate, in the normal manner.

# MODIFYING THE WORKSHOPS

Both Workshops use routines that rely on a variable set to equal the exact number of bytes in the Workshop after having been RUN. These variables and the line numbers in which they appear are:

For Programmer's Workshop, BX in line 32765

For Applesoft Workshop, B in line 63010

In addition, in the Integer Workshop, the variables established in line 32765 are memory dependent and must appear in that order.

We do not recommend that modifications to either Workshop be attempted because of the complex inter-relationship of different routines. In addition, APPEND (in Applesoft Workshop), FORMAT and WRITE all use GOTO's within PRINT statements, which are not handled by RENumber routines.

To avoid line number conflicts with APPENDED programs, both Workshops have been numbered as high as possible. The Applesoft Workshop is numbered from 63000 to 63999 and may not be RENumbered by its own REN routine, which has a built in prohibition against line numbers > 62999.

Programmer's Workshop is numbered in increments of one, from 32765 to 32844, all of which, save three, are above the "legal" range. It may be RENumbered downward by using the Programmer's Aid ROM or as follows, if you use the Workshop's own REN routine:

```
POKE 2,205
POKE 3, 10
POKE 4, 1
POKE 5, 0
POKE 6,253
POKE 7, 127
POKE 8, 255
POKE 9, 255
CALL 3848
```

# USING APPLESOFT WORKSHOP WITHOUT ROM CARD

The Applesoft Workshop requires extensive use of disk and the Floating Point Firmware card. The authors have made modifications to run it without the firmware card, on an experimental basis but these changes are not considered practical. They would include changing pointer values in APPEND and RENUM, the D823 G return to Applesoft from MONITOR and WRITE, and the relocation of the HEX/DEC, VARIABLE LIST and WRITE routines, all of which use zero page.

## PROGRAMMER'S WORKSHOP V. 9.13

This documentation also includes information pertaining to Version 9.13 of the Integer PROGRAMMER'S WORKSHOP. The following routines, which require the use of disk, are not supported in Version 9.13:

CATALOG  
INTEGER  
WRITE

In addition, the user will find POKE on the OPTIONS command list. POKE performs a function similar to WRITE except that its' output is in the form of POKE statements rather than hex strings and they must manually be traced over to enter.

The assembly language APPEND/RENUM and hex converter are also located on page three rather than at \$3072 as in Version 2.5.

# HISTORY AND CREDITS

In late 1977, early 1978, printer interfaces were not available for the Apple II, hence the only way to examine a program was listing it on the screen. At the same time there was no routine available for stopping the listing, which made working on a complex program like checkbook, with its multitude of GOTOs and GOSUBs, nearly impossible.

As a consequence, one of the first programs we wrote was "list by page", a routine planned to overcome the foregoing problem. This was followed shortly by a routine, suggested by Bob Huelsdonk, to convert an assembly program to Basic POKE statements.

These two routines then, became the basis for what today is known as PROGRAMMERS WORKSHOP. The first Workshop, known then as "utility routines" was clumsily written and occupied over 9K of RAM, a far cry from today's fast, streamlined and compact Workshop of just over 4K (in Integer).

It would also be appropriate to express gratitude for the contributions to its' routines by many individuals, and particularly to Darrell and Ron Aldrich who added much to its' formatting and compactness and in ideas.

It is impossible to calculate the hundreds of man hours that have been poured into it since its inception, but we would like at this point to credit each author and his contributions.

NAME OF ROUTINE	PROGRAMMER'S WORKSHOP AUTHOR	APPLESOFT WORKSHOP AUTHOR
Append	Steve Wozniak	Val Golding
Bytes	Val Golding	Val Golding
Catalog	Darrell Aldrich	Ron Aldrich
Disk	—	Ron Aldrich
Exam	Don Williams	Val Golding
Format	—	Val Golding
Integer	Darrell Aldrich	
Kill	Darrell Aldrich	Ron Aldrich
List	Darrell Aldrich	Ron Aldrich
Monitor	S.H. Lam	S.H. Lam & Ron Aldrich
Quit	Darrell Aldrich	Ron Aldrich
Ren	Steve Wozniak	Bob Huelsdonk
Save	Val Golding	Val Golding
Write	Val Golding	Val Golding
Hex-Dec	Darrell Aldrich	Ron Aldrich
Variable List	Ron Aldrich	Val Golding
Structuring	Darrell Aldrich	Ron Aldrich

In many cases, where a single author is listed above, some of the routines were in fact the result of collaborations.

Val J. Golding

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	DECIMAL
8	END	FOR	NEXT	DATA	INPUT	DEL	DIM	READ	GR	TEXT	PR#	IN#	CALL	PLOT	HLIN	VLIN	8 128-143
9	HGR2	HGR	H COLOR=	H PLOT	DRAW	XDRAW	HTAB	HOME	ROT=	SCALE=	SHLOAD	TRACE	NOTRACE	NORMAL	INVERSE	FLASH	9 144-159
A	COLOR=	POP	VTAB	HIMEM:	LOMEM:	ONERR	RESUME	RECALL	STORE	SPEED=	LET	GOTO	RUN	IF	RESTORE	&	A 160-175
B	GOSUB	RETURN	REM	STOP	ON	WAIT	LOAD	SAVE	DEF	POKE	PRINT	CONT	LIST	CLEAR	GET	NEW	B 176-191
C	TAB	TO	FN	SPC(	THEN	AT	NOT	STEP	+	-	*	/	,	AND	OR	>	C 192-207
D	=	<	SGN	INT	ABS	USR	FRE	SCRN(	POL	POS	SQR	RND	LOG	EXP	COS	SIN	D 208-223
E	TAN	ATN	PEEK	LEN	STR\$	VAL	ASC	CHR\$	LEFT\$	RIGHT\$	MID\$			RETURN WITHOUT GOSUB	OUT OF DATA	ILLEGAL QUANTITY	E 224-239
F	OVER FLOW	OUT OF MEMORY	UNDEF'D STATE-MENT	BAD SUB-SCRIPT	REDIM'D ARRAY	DIVISION BY ZERO	ILLEGAL DIRECT	TYPE MISMATCH	STRING TOO LONG	FORMULA TOO COMPLEX	CAN'T CONTINUE	UNDEF'D FUNCTION	ERROR	(	(	(	F 240-255
0	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	

LEAST SIGNIFICANT DIGIT

## APPLESOFT ][ TOKENS Val Golding 05.28.78

NOTE: Values 00 to 7F (0 to 127 decimal) are used by the standard Ascii character set. As in Integer Basic, Apple II outputs last two rows (60-7F) as rows 20-3F.

# MOST SIGNIFICANT DIGIT

TOKENS																ASCII CHAR. & CONTROLS															
0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
HIMEM:	END OF STMT	—	:	LOAD	SAVE	CON	RUN	RUN	DEL	'	NEW	CLR	AUTO	'	MAN	NUL	SOH	STX	ETX	EOT	END	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
HIMEM:	LOMEM:	+	—	*	/	=	#	>=	>	<=	<>	<	AND	OR	MOD	DLE	DC1,	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	SS	US	
>	+	(	,	THEN	THEN	,	,	"	"	(			(	PEEK	RND	SP		"	#	\$	%	&	'	(	)	*	+	,	-	•	/
SGN	ABS	POL	RNDX	(	+	-	NOT	(	=	#	LEN (	ASC (	SCRN (	'	(	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
\$	\$	(	,	,	;	;	;	,	,	,	TEXT	GR	CALL	DIM	(	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	>	—
TAB	END	INPUT	INPUT	INPUT	FOR	=	TO	STEP	NEXT	,	RETURN	GOSUB	REM	LET	GO TO	,	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
IF	PRINT	PRINT	PRINT	POKE	,	COLOR=	PLOT	,	HLIN	,	AT	VLIN	,	AT	VTAB	=	=	)	)	LIST	,	LIST	POP	NODSP	NODSP	NO TRACE	DSP	DSP	TRACE	PR#	IN #
ASCII EQUIV.																															

## APPLE II INTEGER BASIC INTERPRETATION OF MEMORY

Val Golding & Don Williams 3.27.78

NOTE: Rows E & F will be output as rows A & B to Apple II Video Monitor

**Apple Pugetsound Program Library Exchange**  
**6708 - 39th Avenue S.W.**  
**Seattle, Washington 98136**  
**(206) 932-6588**