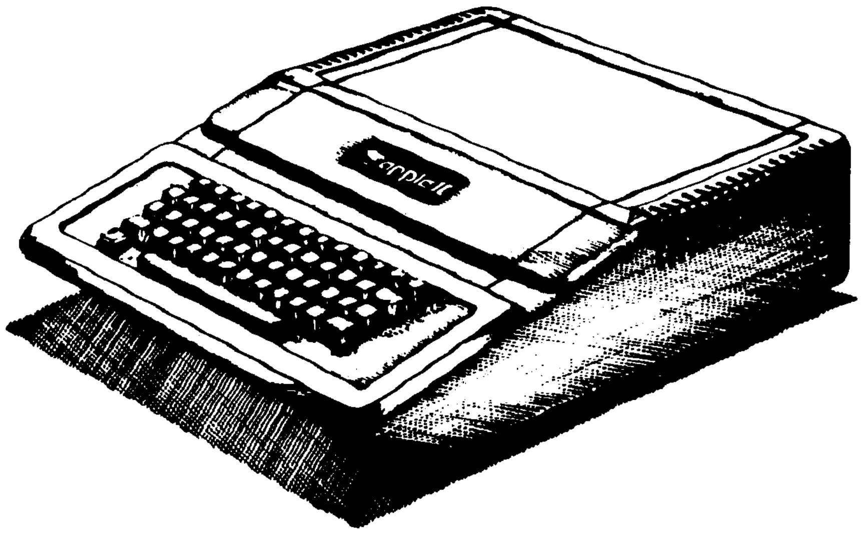
Apple 2 Computer Technical Information



Apple][Computer Family Information

AppleSoft BASIC Info:
Amplifying Apple Soft

Lingwood-

Document # 46

Ex Libris David T. Craig

AMPLIFYING APPLESOFT

APPLESOFT LOCATIONS

(in hexadecimal order)

Compiled by David A. Lingwood

The addresses and call points below are compiled from several sources. Most important is John Crossley's article in this issue. H.M. Long of Raxis International deciphered the command table and contributed his own list of addresses. The Applesoft Reference Manual also contains several of the PEEKs and POKEs mentioned below. This is not a complete PEEK/POKE reference, however.

The most valuable use of this table for me has been in deciphering what Applesoft does. Having a list sorted by RAM address makes it simpler to figure out the purpose of some obscure JSR in the routine I'm analyzing.

Applesoft's command table was a source for some of the data below. Note that this table contains the call points for each command (though the addresses are one byte low for commands END through NEW. After that the table gets flaky, and does not contain calling addresses. The functions are determined by their token alone (apparently), which causes a JSR to PARCHK and ERMEVAL to evaluate the function's argument before going to the code of the function itself.

Finally, there are no guarantees of accuracy here. The routines commonly used with ampersand, to find and evaluate variables, find line numbers, etc., all work as advertised. More esoteric calling points have yet to be evaluated for accuracy and usefulness. Who said there are no more frontiers to conquer?

The table below contains the mnemonic name of the command (invented by me in a few cases), the hex address of the value or routine, then a brief explanation. The Crossley article should be used for more complete explanation.

Name	Address	Purpose
	ZERO PAGE	
CONTINUE	00-02	Jump instruction to continue in Applesoft
STRGJMP	03-05	Jump instiuction to STROUT
USR OP	0A	USR routine jump instruction
USR L	0B	USR jump address
USR H	0C	
CHARAC	0D	Used by STRLT2, build string descriptor
ENDCHR	0E	Used by STRLT2
VALTYP	11	Flags last FAC operation 0=number, FF=string
SUBFLG	14	=$00 if subscripts allowed, $80=no subscripts
H2	2C	Used by PLOTFNS
V2	2D	Used by PLOTFNS
INVFLG	32	Inverse output mask
A1 L	3C	Cassette routine pointer
A1 H	3D	
A2 L	3E	Cassette routine pointer
A2 H	3F	
LINNUM L	50	Gen. purpose 16-bit number location
LINNUM H	51	
TEMPPT	52	Last used temp. string descriptor
LASTPT	53	
INDEX L	5E	Temp string move pointer
INDEX H	5F	
RESULT	62-66	Result of last multiply or divide
TXTTAB L	67	Start of program text
TXTTAB H	68	
VARTAB L	69	Start of variable storage
VARTAB H	6A	
ARYTAB L	6B	Start of array storage
ARYTAB H	6C	
STREND L	6D	Top of array storage
STREND H	6E	
FRETOP L	6F	Bottom of string storage
FRETOP H	70	
FRESPC L	71	Temp. string storage routine pointer
FRESPC H	72	
MEMSIZ L	73	HIMEM
MEMSIZ H	74	
CURLIN L	75	Current line # (=FF if in direct mode)
CURLIN H	76	
OLDLIN L	77	Last line executed
OLDLIN H	78	
OLDTEXT L	79	Mem locn for stmt. to be executed next
OLDTEXT H	7A	
DATLIN L	7B	Current DATA stmt. line #
DATLIN H	7C	
DATPTR L	7D	Address of next DATA byte
DATPTR H	7E	
INPTR L	7F	Ptr. to input source =INBUF if "INPUT"; or pgm. data stmt.

INPTR H	80	
VARNAM L	81	Name of last-used variable
VARNAM H	82	
VARPNT L	83	Used by PTRGET, last var. used by Applesoft
VARPNT H	84	
FORPNT L	85	General pointer, used by COPY
FORPNT H	86	
TEMP1	93-97	Temporary register 1
HIGHDS L	94	Used by BLTU
HIGHDS H	95	
HIGHTR L	96	Used by BLTU
HIGHTR H	97	
TEMP2	98-9C	Temporary register 2
LOWTR L	9B	Gen. purpose register (GETARYPT, FINDLN, BLTU)
LOWTR H	9C	
FAC	9D-A3	Main floating point accumulator
DSCTMP A	9D	Temp. string descriptor
DSCTMP L	9E	
DSCTMP H	9F	
FPGEN	A4	General use in FP math routines
ARG	A5-AA	Argument register
STRNG1 L	AB	Pointer to string, Used by MOVINS
STRNG1 H	AC	
STRNG2 L	AD	Pointer to string, Used by STRLT2
STRNG2 H	AE	
PRGEND L	AF	End of program text
PRGEND H	B0	
CHRGET	B1-C8	Get text from TXTPTR routine
CHRGOT OP	B7	TXTPTR input, no increment
TXTPTR L	B8	TXTPTR jump address
TXTPTR H	B9	
RND	C9-CD	Random number ($CD must be set to #$FF after boot)
HIRESPTRS	D0-D5	HIRES scratch pointers
ERRFLG	D8	= $80 if ONERR is active
ERRLIN L	DA	Line # where error occurred
ERRLIN H	DB	
ERRPOS L	DC	Save TXTPTR for HNDLERR
ERRPOS H	DD	
ERRNUM	DE	Error code number
ERRSTK	DF	Stack pointer value before error
HIRESXY	E0-E2	HIRES X and Y coordinates
HCOLOR	E4	HIRES color byte
GENHIRES	E5-E7	General HIRES use
HPAG	E6	HIRES page to plot on, $20=P1, $40=P2
SHAPE	E8-E9	Pointer to beginning of shape table
COLLIS	EA	HIRES collision counter
FIRST	F0	Used by PLOTFNS
SPDBYT	F1	SPEED= delay number
ORMASK	F3	Mask for flashing output
REMSTK	F8	Stack pointer saved before each stmt.
ROT VALUE	F9	Rotation value for shapes

RAM

FBUFFR	100	100-110 FOUT buffer
1BIL	1E9-1ED	Quantity one billion
BUF	200	200-2FF input buffer
& CMD	3F5	Ampersand branch to machine lang.
& L	3F6	Address for & jump
& H	3F7	

ROM

CMDTABL	D000	Command table name
TOKTABL	D0D0	Base of command token table
ERMSTB	D260	Base of Applesoft error msg. table
BLTU	D393	Make room by block transfer, move everything fwd.
REASON	D3E3	Check that Y,A < FRETOP; may garbage collect
MEMERR	D410	Out of memory error
ERROR	D412	Jump to HNDLERR if ONERR active, else print err msg.
JUMPSTART	D43C	Destination of "0G" RESTART jump in $00-02
INLIN	D52C	Input text from input device to buffer, no prompt

INLIN+2	D52E	Use X for prompt, then INLIN
GDBUFS	D539	Put 0 at end of input buffer, mask MSBs
CMD LOOP	D43C	Main command loop
INCHR	D553	Get char. from input device into A, mask MSB
RUN	D559	Run program, does not return
RUN +	D56C	Special entry into line parser
FNDLIN	D61A	Search pgm. for line# in LINNUM
NEW	D649	NEW
SCRTCH	D64B	"NEW" - clear pgm., vars., stack
CLEARC	D66C	"CLEAR" vars. & stack
STKINI	D683	Clear stack only (CALL for clearing loops, GOSUBs)
STXTPT	D697	Set TXTPTR to start of pgm.
LIST	D6A5	LIST the pgm.
FOR	D766	Start of FOR-NEXT loop
NEWSTT	D7D2	Execute new stmt., does not return
RESTOR	D849	Set data pointer, DATPTR, to start of pgm.
ISCNTC	D858	Ck. keyboard for CTRL-C & break if so
STOP	D86E	Stop the pgm.
END	D870	Terminate execution
CONT	D898	Move OLDTXT, OLDLIN to TXTPTR, CURLIN
SAVE	D8B0	Save pgm. to tape
LOAD	D8C9	Load pgm. from tape
VARTIO	D8F0	Set up A1 & A2 to save 3-byte pgm. length
PROGIO	D901	Set up A1 & A2 to save pgm. text
RUN	D912	RUN a pgm.
GOSUB	D921	GOSUB branch function
GOTO	D93E	GOTO branch function
GOTO +	D941	Special GOTO entry
RETURN	D96A	Return from GOSUB
RETURN	D96B	Return from subroutine
GET	D98F	GET an input character
DATA	D995	Move TXTPTR to end of stmt.
ADDON	D998	Add Y to TXTPTR
DATAN	D9A3	Calculate offset from TXTPTR to : or EOL, into Y
REMN	D9A6	Calculate offset from TXTPTR to next COL(0), into Y
IF	D9C9	IF test function
REM	D9DC	REMARK
GOTO	D9E3	Use LINGET and FNDLIN to update TXTPTR
ONGOTO	D9EC	ON-GOTO function
LINGET	DA0C	Read line# from TXTPTR into LINNUM
LET	DA46	Use CHRGET to find add. of var., eval. formula & store
COPY	DAB7	Free temp string in mem(Y,A), & move to mem(FORPNT)
PRINT	DAD5	Print output
CRDO	DAFB	Print C.R.
STROUT	DB3A	Print string in mem(Y,A)
STRPRT	DB3D	Print string with descriptor at mem(FACMO,FACLO)
OUTSP	DB57	Print a space
OUTQST	DB5A	Print a ?
OUTDO	DB5C	Print A. INV, FLASH, NORMAL in effect
INPUT	DBB2	INPUT routine exec
READ	DBE2	READ DATA routine
NEXT	DCF9	End of FOR-NEXT loop
FRMNUM	DD67	Evaluate formula at TXTPTR into FAC, ck. for #
CHKNUM	DD6A	Ck. FAC for numeric
CHKSTR	DD6C	Ck. FAC for string
CHKVAL	DD6D	Ck. most recent FAC for str. or #; mismatch if FAC<>C
FRMEVL	DD7B	Evaluate formula at TXTPTR into FAC, using CHRGET
STRTXT	DE81	Set Y,A = TXTPTR+C; fall into STRLIT
PARCHK	DEB2	Ck. for "(", evaluate formula, ck. for ")"; fall CHKCLS
CHKCLS	DEB8	Ck. TXTPTR for ")"
CHKOPN	DEBB	Ck. TXTPTR for "("
CHKCOM	DEBE	Ck. TXTPTR for ","
SYNCHR	DEC0	Ck. TXTPTR for char. in A
OR	DF4F	OR function
AND	DF55	AND function
PDL	DFCD	Read GAME PADDLE
DIM	DFD9	DIMension a variable
PTRGET	DFE3	Read var. name & find in memory, returns Y,A address
COLD	E000	Cold start entry
WARM	E003	Warm start entry
ISLETC	E07D	Ck. A for ASCII letter (C set if so)
-32768	E0FE	Quantity -32768
AYINT	E10C	Perform QINT if FAC <32767 and >-32767
SUB ERR	E196	Subscript error

FRE	E2DE	The FRE(X) function
GIVAYF	E2F2	Float signed integer in A,Y
POS	E2FF	Position in HIRES
SNGFLT	E301	Float unsigned integer in Y
ERRDIR	E306	Illegal diect error if pgm. not running
DEF	E313	Define function
STR$	E3C5	String function
STRINI	E3D5	Get space for string creation, create descriptor
STRSPA	E3DD	JSR GETSPA, store pointer & length in DSCTMP
STRLIT	E3E7	Store " in ENDCHR & CHARAC to stop STRLT2
STRLT2	E3ED	Build descriptor for string(Y,A); fall into PUTNEW
PUTNEW	E42A	Move string(DSCTMP) to temp descriptor
FRM ERR	E430	Formula too complex error
GETSPA	E452	Get space for string; may garbage collect
GARBAG	E484	Move strings up in memory
CAT	E597	Concatenate two strings
MOVINS	E5D4	Move string(STRNG1) to mem(FRESPA)
MOVSTR	E5E2	Move string(Y,X), len(A), to mem(FRESPA)
FRESTR	E5FD	Ck. last FAC for string, fall into FREFAC
FREFAC	E600	Free temp descriptor pointer FAC
FRETMP	E604	Free up a temp string
FRETMS	E635	Free temp string descriptor only
CHR$	E646	CHR string function
LEFT$	E65A	LEFT string function
RIGHT$	E686	RIGHT string function
MID$	E691	MID string function
LEN	E6D6	String LENGTH function
ASC	E6E5	ASC function
GTBYTC	E6F5	JSR CHRGET, gobble char., fall into GETBYT
GETBYT	E6F8	Evaluate formula at TXTPTR
CONINT	E6FB	Convert FAC into 1-byte integer in X and FACLO
VAL	E707	Value of string
GETNUM	E746	Read 2-byte# from TXTPTR to LINNUM, + 1-byte X if comma
COMBYTE	E74C	Ck. for comma and get byte into X
GETADR	E752	Convert FAC to 2-byte integer in LINNUM
PEEK	E764	PEEK memory location
POKE	E77B	POKE memory location
WAIT	E784	WAIT function
FADDH	E7A0	Add 1/2 to FAC
FSUB	E7A7	Move mem(Y,A) to ARG; fall into FSUBT
FSUBT	E7AA	Subtract FAC from ARG
FADD	E7BE	Move mem(Y,A) to ARG; fall into FADDT
FADDT	E7C1	Add FAC and ARG
OFLW ERR	E8D5	Overflow error
1	E913	Quantity 1
SQR(.5)	E92D	Quantity SQR(.5)
SQR(2)	E932	Quantity SQR(2)
-1/2	E937	Quantity -1/2
LN(2)	E93C	Quantity LOGN(2)
LOG	E941	LOGe of FAC
FMULT	E97F	Move mem(Y,A) to ARG; fall into FMULTT
FMULTT	E982	Multiply FAC and ARG
CONUPK	E9E3	Load ARG from mem(Y,A)
MUL10	EA39	Multiply FAC by 10 (both + and - numbers)
10	EA50	Quantity 10
DIV10	EA55	Divide FAC by 10 (positive #s only)
FDIV	EA66	Move mem(Y,A) into ARG; fall into FIDVT
FIDVT	EA69	Divide ARG by FAC
MOVFM	EAF9	Move mem(Y,A) into FAC
MOV2F	EB1E	Pack FAC into temp register 2, uses MOVMF
MOV1F	EB21	Pack FAC into temp register 1, uses MOVMF
MOVML	EB23	Pack FAC into zero page(X)
MOVMF	EB2B	Pack FAC into mem(Y,A)
MOVFA	EB53	Move ARG into FAC
MOVAF	EB63	Move FAC into ARG
RNDB	EB72	Round last locn. of FAC
SIGN	EB82	Set A according to value of FAC
SGN	EB90	Call SIGN and float results in FAC
FLOAT	EB93	Float signed integer in A
ABS	EBAF	Absolute of FAC
FCOMP	EBB2	Compare FAC and Packed # in mem(Y/A)
QINT	EBF2	Quick greatest integer function
INT	EC23	Greatest integer value of FAC, uses QINT
FIN	EC4A	Input floating # into FAC from CHRGET

1BIL	ED14	Quantity 1000000000
INPRT	ED19	Print "IN" and line # from CURLIN
LINPRT	ED24	Print 2-byte # in X,A
PRNTFAC	ED2E	Print current FAC
FOUT	ED34	Create string in FBUFFR = to value of FAC
1/2	EE64	Quantity 1/2
SQR	EE8D	Square root of FAC
FPWRT	EE97	Exponentiate ARG to the FAC power
NEG	EECF	NEGATE function
NEGOP	EED0	FAC = -FAC
LOGE(2)	EEDB	Quantity LOGe(2)
EXP	EF09	e to the FAC power
RND	EFAE	Form random # in FAC
COS	EFEA	COS(FAC)
SIN	EFF1	SIN(FAC)
TAN	F03A	TAN(FAC)
PI/2	F063	Quantity PI/2
PI*2	F06B	Quantity PI*2
1/4	F07C	Quantity 1/4
ATN	F09E	ARCTAN(FAC)
COLDST	F128	Cold start point
CALL	F1D5	CALL machine language location
IN#	F1DE	Set input port
PR#	F1E5	Set output port
PLOTFNS	F1EC	Get LORES coordinates from TXTPTR
PLOT	F225	Plot a LORES point
HLIN	F232	Draw horizontal LORES line
VLIN	F241	Draw vertical LORES line
COLOR	F24F	Set LORES color
VTAB	F256	Vertical TAB
SETTRACE	F26D	Turn on TRACE
TRACEOFF	F26F	Turn off TRACE
SETNORM	F273	Set NORMAL text
INVERSE	F277	Set INVERSE text
FLASH	F280	Set FLASHING text
HIMEMSET	F286	Set HIMEM pointer
LOMEMSET	F2A6	Set LOMEM pointer
ONERR	F2CB	Set ONERR flag
HANDLERR	F2E9	Save CURLIN, TXTPTR, X (in ERRNUM) and REMSTK
RESUME	F318	Restore CURLIN, TXTPTR and STACK
DELETE	F32D	Delete a line
SETGR	F38C	Set graphics routine
GR	F390	Set mixed graphics
SETTXT	F395	Set text mode
STORE	F39B	Save variables/array to tape
RECALL	F3B8	Recall vars/array from tape
HGR2	F3D4	Init & clear HIRES p.2
HGR	F3DE	Init & clear HIRES p.1
HCLR	F3EE	Clear HIRES screen to black
BKGND	F3F2	Clear HIRES screen to last-plotted color
HPOSN	F40D	Positon HIRES cursor without plotting
HPLOT	F453	HPOSN, then plot a dot at cursor
HLIN	F530	Draw a line
HFIND	F5CB	Convert HIRES cursor to coordinates (used after shape)
DRAW	F601	Draw shape pointed to by Y,X, ROT=A
XDRAW	F65D	Draw shape pointed to by Y,X, A=ROT
HFNS	F6B9	Get HIRES coordinates from TXTPTR
SETHCOL	F6EC	Set HIRES color to X
LINE	F6FA	Draw HIRES line
SETROT	F71D	Set rotation for shape
SETSCALE	F727	Set scale for shape
SHLOAD	F775	Load shape table from tape to memory
GETARYPT	F7D9	Read var. name from CHRGET & find in memory
HTAB	F7E7	Horizontal tab X # of spaces