Bob Kahn
Director of Special Projects
Atari, Inc.
Box 427
1196 Borregas
Sunnyvale, CA, 94086

Dear Bob:

Enclosed are originals and backup for

- Coin-Op FORTH, Swarthmore Disk 1, Disk 2, and QE Disk, containing bootable versions of FORTH and appropriate source screens (see Section I.1 of the document, "What this is")

- 5 AtariWriter disks of documentation marked "AtariWriter ①" to "AtariWriter ⑤".

- Instructions for printing out document (on the following page).

- One copy of printout.

I hope you find it satisfactory. We think it's pretty darn good.

— Regards
Gene Klotz

Instructions For Printing Document

This document is on 5 AtariWriter disks, marked "AtariWriter ①" to "AtariWriter ⑤". Two copies of each are enclosed.

For a complete printout you only need to start printing the files indicated.

Disk Number	Beginning of Chain Files	Sections of Documents
①	TITLE.PG	I
②	FIG.MAN	II.1
	EXCL.FIG	II.2
③	ATARI.EXT	II.3 - II.5
	WTOBJOLD.MAN	III
④	IV.1	IV.1 - IV.6
⑤	IV.17	IV.17
	V	V

Gen Klotz 6/1/83

F.I.G. FORTH EDITOR

(7/8/80 - SRC)

n EDIT	- Enter editor vocabulary on screen n.
EDITOR	- VOCABULARY name - used to get to the Editor words.
SCR	- VARIABLE containing the current edit screen #.
L	- List current screen.
N	- List next screen.
LL	- List lower 8 lines for screen editor.
UL	- List upper 8 lines for screen editor.
DOIT	- Take the top 16 lines on the TV display and put into the upper or lower half of the edit SCR. (whichever was last displayed). Use the colleen cursor and edit keys to change the screen. Move the cursor to the DOIT line and hit RETURN. DO NOT HIT RETURN while editing the LL or UL screens.
SRC DEST COPY	- copy block SRC to block #DEST.
FIRST LAST SHOW	- list all blocks inclusive.
n LIST	- set SCR to n and list the block.

Line Edit Functions:

n TL	- Type line n and save it in PAD.
n HL	- Hold line n in PAD.
n DL	- Delete line n - save it in PAD.
n IL	- Insert PAD after line n.
n RL	- Replace line n with PAD.
n SL	- Spread to create blank line n by pushing every thing down and losing line F.
n BL	- Blank line n - not saved in PAD.
n SCR CL	- Move line n of Screen #SCR to PAD.
n $	- Put text following $ up to RETURN onto line n.
n %	- Insert text following % up to RETURN after line n.
FLUSH	- RETURNS to FORTH Vocabulary. Writes out changed blocks.

Currently 7/8/80 8 Load loads the assembler, Ed Logg's Colleen I/O and Graphics words and this Editor.

This is normally typed as - (dash) > (greater-than)

ATARI FIG ADDITIONS
(From DECUS)

1+!	(addr →)	add one to word at addr
1-	(n → n-1)	subtract one from TOS
OSET	(addr →)	set word at address to 0
2*	(n → n)	mult. TOS by 2 } faster than
2/	(n → n)	div. TOS by 2 } 2*, 2/ spaces F-stash
CHH	(b →)	type value of LSB of TOS as 2-digit HEX
HH	(u →)	type value of TOS in unsigned HEX
CH?	(addr →)	type value of byte at addr in HEX
H?	(addr →)	type value of word at addr in unsigned HEX
J	(→ n)	return Index for next outer most DO-LOOP
TBL		;defines ROM table (like DECUS, CALTECH IARRAY)
ALLOC	(n →)	allocate n words in RAM
ARRAY	(n →)	CREATE named ARRAY, n words long, returns addr of first element when exe
S?	(→)	TYPE contents of param stack, without removing them.
RP	(→ n)	point to TOP of RETURN (control) stack
R?	(→)	TYPE contents of return stack (non-destructive)
BDUMP	(addr ,addr →)	Display contents of addr to addr (inclusive) always in increments of 8 bytes.

TERMINAL INPUT-OUTPUT

.	(n →)	Print number.
.R	(n fieldwidth →)	Print number, right-justified in field.
D.	(d →)	Print double-precision number.
D.R	(d fieldwidth →)	Print double-precision number, right-justified in field.
CR	(→)	Do a carriage return.
SPACE	(→)	Type one space.
SPACES	(n →)	Type n spaces.
."	(→)	Print message (terminated by ").
DUMP	(addr u →)	Dump u words starting at address.
TYPE	(addr u →)	Type string of u characters starting at address.
COUNT	(addr → addr+1 u)	Change length-byte string to TYPE form.
?TERMINAL	(→ f)	True if terminal break request present.
KEY	(→ c)	Read key, put ascii value on stack.
EMIT	(c →)	Type ascii value from stack.
EXPECT	(addr n →)	Read n characters (or until carriage return) from input to address.
WORD	(c →)	Read one word from input stream, using given character (usually blank) as delimiter.

INPUT-OUTPUT FORMATTING

NUMBER	(addr → d)	Convert string at address to double-precision number.
<#	(→)	Start output string.
#	(d → d)	Convert next digit of double-precision number and add character to output string.
#S	(d → 0 0)	Convert all significant digits of double-precision number to output string.
SIGN	(n d → d)	Insert sign of n into output string.
#>	(d → addr u)	Terminate output string (ready for TYPE).
HOLD	(c →)	Insert ascii character into output string.

DISK HANDLING

LIST	(screen →)	List a disk screen.
LOAD	(screen →)	Load disk screen (compile or execute).
BLOCK	(block → addr)	Read disk block to memory address.
B/BUF	(→ n)	System constant giving disk block size in bytes.
BLK	(→ addr)	System variable containing current block number.
SCR	(→ addr)	System variable containing current screen number.
UPDATE	(→)	Mark last buffer accessed as updated.
FLUSH	(→)	Write all updated buffers to disk.
EMPTY-BUFFERS	(→)	Erase all buffers.

DEFINING WORDS

: xxx	(→)	Begin colon definition of xxx.
;	(→)	End colon definition.
VARIABLE xxx	(n →) xxx: (→ addr)	Create a variable named xxx with initial value n; returns address when executed.
CONSTANT xxx	(n →) xxx: (→ n)	Create a constant named xxx with value n; returns value when executed.
CODE xxx	(→)	Begin definition of assembly-language primitive operation named xxx.
;CODE	(→)	Used to create a new defining word, with execution-time "code routine" for this data type in assembly.
<BUILDS . . . DOES>	does: (→ addr)	Used to create a new defining word, with execution-time routine for this data type in higher-level Forth.

VOCABULARIES

CONTEXT	(→ addr)	Returns address of pointer to context vocabulary (searched first).
CURRENT	(→ addr)	Returns address of pointer to current vocabulary (where new definitions are put).
FORTH	(→)	Main Forth vocabulary (execution of FORTH sets CONTEXT vocabulary).
EDITOR	(→)	Editor vocabulary; sets CONTEXT.
ASSEMBLER	(→)	Assembler vocabulary; sets CONTEXT.
DEFINITIONS	(→)	Sets CURRENT vocabulary to CONTEXT.
VOCABULARY xxx	(→)	Create new vocabulary named xxx.
VLIST	(→)	Print names of all words in CONTEXT vocabulary.

MISCELLANEOUS AND SYSTEM

(	(→)	Begin comment, terminated by right paren on same line; space after (.
FORGET xxx	(→)	Forget all definitions back to and including xxx.
ABORT	(→)	Error termination of operation.
' xxx	(→ addr)	Find the address of xxx in the dictionary; if used in definition, compile address.
HERE	(→ addr)	Returns address of next unused byte in the dictionary.
PAD	(→ addr)	Returns address of scratch area (usually 68 bytes beyond HERE).
IN	(→ addr)	System variable containing offset into input buffer; used, *e.g.*, by WORD.
SP@	(→ addr)	Returns address of top stack item.
ALLOT	(n →)	Leave a gap of n bytes in the dictionary.
,	(n →)	Compile a number into the dictionary.

FORTH HANDY REFERENCE

Stack inputs and outputs are shown; top of stack on right.
This card follows usage of the Forth Interest Group (S.F. Bay Area); usage aligned with the *Forth 78* International Standard.
For more info: Forth Interest Group
P.O. Box 1105
San Carlos, CA 94070.

Operand key:

n, n1, . . .	16-bit signed numbers
d, d1, . . .	32-bit signed numbers
u	16-bit unsigned number
addr	address
b	8-bit byte
c	7-bit ascii character value
f	boolean flag

STACK MANIPULATION

DUP	(n → n n)	Duplicate top of stack.
DROP	(n →)	Throw away top of stack.
SWAP	(n1 n2 → n2 n1)	Reverse top two stack items.
OVER	(n1 n2 → n1 n2 n1)	Make copy of second item on top.
ROT	(n1 n2 n3 → n2 n3 n1)	Rotate third item to top.
-DUP	(n → n ?)	Duplicate only if non-zero.
>R	(n →)	Move top item to "return stack" for temporary storage (use caution).
R>	(→ n)	Retrieve item from return stack.
R	(→ n)	Copy top of return stack onto stack.

NUMBER BASES

DECIMAL	(→)	Set decimal base.
HEX	(→)	Set hexadecimal base.
BASE	(→ addr)	System variable containing number base.

ARITHMETIC AND LOGICAL

+	(n1 n2 → sum)	Add.
D+	(d1 d2 → sum)	Add double-precision numbers.
-	(n1 n2 → diff)	Subtract (n1-n2).
*	(n1 n2 → prod)	Multiply.
/	(n1 n2 → quot)	Divide (n1/n2).
MOD	(n1 n2 → rem)	Modulo (*i.e.* remainder from division).
/MOD	(n1 n2 → rem quot)	Divide, giving remainder and quotient.
*/MOD	(n1 n2 n3 → rem quot)	Multiply, then divide (n1*n2/n3), with double-precision intermediate.
*/	(n1 n2 n3 → quot)	Like */MOD, but give quotient only.
MAX	(n1 n2 → max)	Maximum.
MIN	(n1 n2 → min)	Minimum.
ABS	(n → absolute)	Absolute value.
DABS	(d → absolute)	Absolute value of double-precision number.
MINUS	(n → -n)	Change sign.
DMINUS	(d → -d)	Change sign of double-precision number.
AND	(n1 n2 → and)	Logical AND (bitwise).
OR	(n1 n2 → or)	Logical OR (bitwise).
XOR	(n1 n2 → xor)	Logical exclusive OR (bitwise).

COMPARISON

<	(n1 n2 → f)	True if n1 less than n2.
>	(n1 n2 → f)	True if n1 greater than n2.
=	(n1 n2 → f)	True if top two numbers are equal.
0<	(n → f)	True if top number negative.
0=	(n → f)	True if top number zero (*i.e.*, reverses truth value).

MEMORY

@	(addr → n)	Replace word address by contents.
!	(n addr →)	Store second word at address on top.
C@	(addr → b)	Fetch one byte only.
C!	(b addr →)	Store one byte only.
?	(addr →)	Print contents of address.
+!	(n addr →)	Add second number on stack to contents of address on top.
CMOVE	(from to u →)	Move u bytes in memory.
FILL	(addr u b →)	Fill u bytes in memory with b, beginning at address.
ERASE	(addr u →)	Fill u bytes in memory with zeroes, beginning at address.
BLANKS	(addr u →)	Fill u bytes in memory with blanks, beginning at address.

CONTROL STRUCTURES

DO . . . LOOP	do: (end+1 start →)	Set up loop, given index range.
I	(→ index)	Place current index value on stack.
LEAVE	(→)	Terminate loop at next LOOP or +LOOP.
DO . . . +LOOP	do: (end+1 start →) +loop: (n →)	Like DO . . . LOOP, but adds stack value (instead of always '1') to index.
IF . . . (true) . . . ENDIF	if: (f →)	If top of stack true (non-zero), execute. [Note: *Forth 78* uses IF . . . THEN.]
IF . . . (true) . . . ELSE . . . (false) . . . ENDIF	if: (f →)	Same, but if false, execute ELSE clause. [Note: *Forth 78* uses IF . . . ELSE . . . THEN.]
BEGIN . . . UNTIL	until: (f →)	Loop back to BEGIN until true at UNTIL. [Note: *Forth 78* uses BEGIN . . . END.]
BEGIN . . . WHILE . . . REPEAT	while: (f →)	Loop while true at WHILE; REPEAT loops unconditionally to BEGIN. [Note: *Forth 78* uses BEGIN . . . IF . . . AGAIN.]

~~TED:~~ BOB:

Here is the documentation on Fig-Forth 1.4 (Calfee). The address for "Going Forth" is:

Creative Solutions
14625 Tynewick Terrace
Silver Springs MD. 20906

This stuff doesn't seem to cover the editor, so I'm sending a Xerox of the paper stuff. The editor is pretty simple, just whatever you do don't hit RETURN. Ed Rotberg has an improved (less lethal) version. You might try him at 745-1090 (Videa)

Sorry, My brain malfunctions when overheated by over work

good luck, call again Monday if problems arise.

GRAPHICS and CONTROLLER words

<n>	PADDLE	1-1	n=0-7 Reads paddle <n>. Value is 0-FF
<n>	PTRIG	1-1	n=0-7 Reads paddle trigger button <n> 0=pressed, 1=not pressed.
<n>	STICK	1-1	n=0-3 Reads joystick <n>. Lower 4 bits represent 4 directions. (0=ON 1=OFF) D3-right, D2-left, D1-down, D0-up
<n>	STRIG	1-1	n=0-3 Reads joystick <n> trigger button. 0=pressed, 1=not pressed
<n>	GRAPHICS GR.	1-0	Open screen in graphics mode <n> (BASIC mode #, NOT ANTIC). Add 16 (dec.) to eliminate split screen. Add 32 (dec.) to prevent clearing screen
<n>	COLOR C.	1-0	Set "color" (pixel data) for subsequent PLOT or DRAWTO.
<x> <y>	PLOT PL.	2-0	Plot a point at (<x>,<y>) on screen. (0,0) is upper left corner. See BASIC reference manual for limits in particular modes.
<x> <y>	DRAWTO DR.	2-0	Draw a line from current position to (<x>,<y>).
<x> <y>	POSITION POS.	2-0	Set current position to (<x>,<y>). Does not plot, used with PUT, GET.
<n>	PUT	1-0	Put the value <n> at current position. Current <x> is then incremented.
	GET	0-1	Get the value of pixel data at current position. Current <x> is then incremented.
<x> <y>	LOCATE	2-1	Same as <x> <y> POS. GET
<n> <c> <l>	SE. SETCOLOR	3-0	Set color register <n> to color <c>, luminance <l>.
<n> <p> <d> <v>	SOUND SO.	4-0	Set sound channel <n> to pitch <p>, "distortion" <d>, and volume <v>. Consult BASIC reference manual for further info. In general, storing to F1AUD, C1AUD, etc. will be easier.

.c

PLAYER/MISSILE Graphics

<n>	PLAYER	1-0	Set up player-missile graphics with <n> (n=1 or 2) vertical line resolution.
	PBASE	constant	Returns base address of player/missile DMA area. This is set by GR., and changes with different modes.
<x> <n>	HPOS!	2-0	n=0-7 Set PLAYER/MISSILE <n> horizontal position to <x>. MISSILE 0 = PLAYER 4 etc.
<x> <n>	SIZE!	2-0	n=0-4 Set size of PLAYER/MISSILE

<x> <n> SIZE! 2-0 n=0-4 Set size of PLAYER <n> (all MISSILEs if n=4) to <x>. x=1 -> double size. x=3 -> quad size.

<x> <n> COLPM! 2-0 n=0-3, x=0-254 step 2
Set color of PLAYER/MISSILE <n> to <x>. Note that MISSILE <n> is same color as PLAYER <n> unless "fifth player enable" is selected. (see PRIOR in hardware manual)

PRIOR byte variable You should store the desired priority value here. Consult the hardware manual for details.

VDELAY byte variable You should store the "vertical delay" bits here. This is only needed if single line positioning is needed with double line graphics resolution. Consult the hardware manual.

.c
CIO words

<n> IOCB 1-0 n=0-7 Set "current" IOCB to <n>. IOCB 0 is normally open to E: and should not be changed. IOCB 1 is used by GRAPHICS commands and should be restored before using them. (ed. note.- I know that this should have been IOCB 6, and that words shouldn't make such assumptions. I didn't write this stuff, I'm just documenting what exists.)

CIO 0-0 Calls CIO. Assumes that " <n> IOCB " has been done, and that IOCB n is correctly set up. You should carefully read the O.S. manual. The following "constants" will refer to the "current" IOCB:

ICDNO Device # in HATABS. (byte)
ICCOM Command (byte)
ICSTA Status (byte)
ICBAL Buffer Address (word)
ICBLL Buffer Length (word)
I1CAX AUX1 (byte)
I2CAX AUX2 (byte) (note: these are odd to allow use with fixed-length-name FORTHs)
ICPTL "put-byte" address (word) (used by BASIC, shun like plague)

<n> ACIO 1-0 Call CIO with acc=n. Normally used with ICBLL=0

CIOA 0-1 Call CIO and put returned acc on stack. Normally used with ICBLL=0

<2> <1> <d> OPEN 3-0 Open current IOCB with AUX1=<1>, AUX2=<2> <d> pointing to dvc:file.ext. <d> should be just a pointer (no count) and string should be terminated with non-alphanumeric.

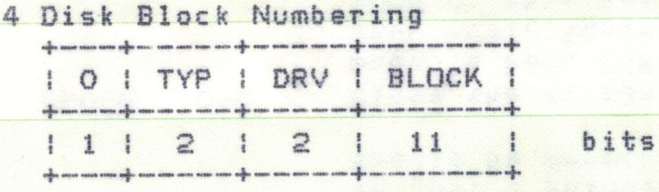
CLOSE 0-0 Close the current IOCB. One of the quickest ways to hang FORTH is to type CLOSE without previously typing <n> IOCB. This closes IOCB 0, then returns to the outer interpreter which attempts to read from IOCB 0. The resulting error message also tries to get out via IOCB 0.

<n> PUT 1-0 Do a "put character" with char=<n>

GET 0-1 Get one character from current IOCB, returning character on stack.

.c

Fig 1.4 Disk Block Numbering



Where:

TYP = 0 (add 0000) -> 810 type 5 1/4" single dens.
1 (" 2000) -> 815 or 8" double dens.
2 (" 4000) -> 8" single dens. DEC interleave
3 (" 6000) -> 8" " " non-interleaved

DRV = 0 (add 0000) -> Drive 1
1 (" 0800) -> " 2
2 (" 1000) -> " 3
3 (" 1800) -> " 4

.c

Dictionary Entries

Fig 1.3 and earlier

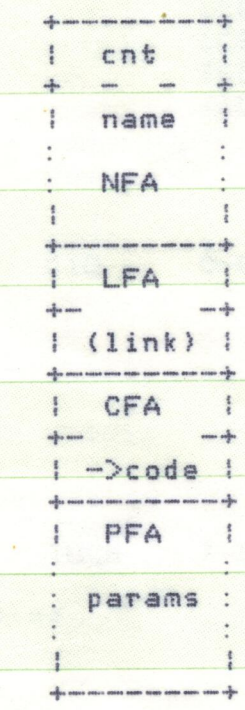


Fig 1.4