DISK GUIDE™

ATARI® 400/800™

David A. Wilson, Series Editor

Osborne/McGraw-Hill

ATARI® 400/800™ DiskGuide™

John Taylor

This compact computer-side reference contains vital commands and functions for using the ATARI® 400/800™ computers.

Other books in the DiskGuide series include:

IBM® PC DiskGuide™, by David A. Wilson (94-2)
VisiCalc® DiskGuide™, by David A. Wilson (98-5)
CP/M® DiskGuide™, by Curtis A. Ingraham (97-7)
Apple® II DiskGuide™, by Zelda Gifford (96-9)

Osborne/McGraw-Hill
Berkeley, California

Published by Osborne/McGraw-Hill
2600 Tenth Street
Berkeley, California 94710

ATARI® 400/800™ DISKGUIDE™

1234567890 GBGB 89876543

ISBN 0-931988-95-0

Series Editor: David A. Wilson
Cover Design: Yashi Okita
Text design: KLT van Genderen

Contents

ATARI 400/800 Keyboard

BREAK	Halts current operation; returns computer from Programmed to Immediate mode; in Immediate mode, cancels current line
CAPS/LOWR	Shifts to lowercase or control-character mode
ESC	Press and release before next key; *see* Escape Sequences, p. 15
SYSTEM RESET	Halts current operation and initializes system; shifts computer to Immediate mode; dangerous when the disk drive is active
⅄	Toggles normal/inverse video mode
CTRL, ESC, CAPS/LOWR, SHIFT, ⅄	ATASCII characters; *see* ATASCII Codes, Characters, & Keystrokes, p. 51.

SHIFT Key Functions	
Keystroke	**Character or Action**
SHIFT-TAB	Set tab stop
SHIFT-<	Clear display screen
SHIFT->	Insert blank line
SHIFT-BACK S	Delete current line
SHIFT-CAPS/LOWR	Switch keyboard to upper-case mode

CTRL Key Combinations	
Keystroke	**Character or Action**
CTRL-TAB	Clear tab stop
CTRL--	Move cursor up one line
CTRL-=	Move cursor down one line
CTRL-+	Move cursor left one space
CTRL-*	Move cursor right one space
CTRL-1	Freeze/restart screen display
CTRL-3	Usually results in an error
CTRL-<	Clear display screen
CTRL->	Insert a space
CTRL-BACK S	Delete next character
CTRL-CAPS LOWR	Switch keyboard to graphics mode

Cassette Use

To write-protect Side A, hold up cassette with label toward you; remove tab from notch on top left.

EOF Stops Tape

To SAVE and LOAD *program*:

- Position tape; if start position not known, load programs until you reach it.
- Type appropriate saving/loading command (*see* p. 11-12)

- Wait for console beep—once for SAVE, twice for LOAD
- Press PLAY (and RECORD for SAVE) on recorder; press RETURN on console
- Wait for 20-second data file leader tone from TV (SAVE only)
- Wait for soundburst(s) from TV speaker to signal SAVE/LOAD in progress

Diskette Use

Diskettes can contain 64 files, 90K characters. File names begin with capital letter and contain 1-8 capital letters or numbers plus optional extension. File name extensions contain any combinations of 1-3 capital letters or numbers.

Examples:

DISKFILE.BAS B1234567.01A JUNK

*.TXT	Names all files with .TXT extension
FILE.*	Names all files named FILE, any extension
FILE*	Names any file whose name begins with FILE
FILE.*T	Names any file whose extension ends with T

DOS

There are two versions of DOS, 1.0 and its replacement 2.0S. DOS 2.0S can always read diskettes prepared by DOS 1.0; the reverse is not always true.

DOS Start-up Menu

The DOS command activates the utilities menu, which includes choices:

***a.* DISK DIRECTORY**

Lists files and free sectors

Examples:

RETURN

Lists all files on drive #1 (default)

D2:

Lists all files on drive #2 disk; D optional

1:,P:
Sends list of all files on drive #*1* to printer

2:*.TXT
Lists all .TXT files on drive #2

*DISKFILE BAS 038
Signifies a locked (*) file with *38* 128-byte blocks (possible display)

***b.* RUN CARTRIDGE**

Transfers control from operating system to ROM cartridge; substitute SYSTEM RESET with DOS 2.0S if MEM.SAVE file active

***c.* COPY FILE**

Copies files on same diskette with one drive, on different diskettes with two drives (use menu item *o* to copy on different diskette on same drive)

Examples:

FILE1.BAS,FILE1.BAK
Copies *FILE1.BAS* on diskette on drive #1 to *FILE1.BAK* on same diskette. Will overwrite *FILE1.BAK* if it already exists

NAMES.TXT,NUMBERS.TXT/A
Appends *NAMES.TXT* to end of *NUMBERS.TXT* (2.0S only); overwrites *NUMBERS.TXT* if /A not given

PROG1.BAS,PROG2.BAS/A
Appends correctly only if BASIC program was stored with LIST not with SAVE

*.BAS,D2:
Copies all BASIC files on drive #1 to drive *#2*

***d.* DELETE FILE(S)**

Examples:

D2:FILE
Deletes FILE on drive #*2*. (Wild cards okay)

D2:FILE/N
Deletes FILE on drive #*2*. (/N disables confirmation dialog)

e. **RENAME FILE**

Renames a file; requires no drive number with new name; old name can be ambiguous (don't rename DOS.SYS or DUP.SYS files)

1:.DAT,*.TXT*

Renames all *.DAT* files on drive *#1* to *.TXT* files (drive number optional with old name, invalid with new); Danger: don't create duplicate filenames

f. **LOCK FILE**

Prevents changes to file. (Wild cards okay); lock DOS.SYS and DUP.SYS for protection

g. **UNLOCK FILE**

Permits changes to file. (Wild cards okay)

h. **WRITE DOS FILES**

Copies DOS.SYS and DUP.SYS files from memory to disk (don't use menu item C)

i. **FORMAT DISK**

Prepares new disk; erases all information on disk

j. **DUPLICATE DISK**

Copies entire disk; erases program in memory if used with DOS 1.0 (enter 1,1

if using one drive; enter 1,2 or 2,1 with two drives)

k. **BINARY SAVE**

Saves a memory area onto disk as an object file

Examples:

PROGM1.OBJ/A,*3E00,4AFF*

Saves memory area, beginning at hex address *3E00* and ending at hex address *41AFF*

If DOS 1.0: POKE program starting address into memory locations 736 and 737 before selecting K; /A saves address. If DOS 2.0S: INIT,RUN option is displayed, RUN is program starting address, INIT starting address of initialization routine; INIT,RUN in hex.

PROGRM4.OBJ,*3E10,517F,,4800*
Omits INIT

AUTORUN.SYS,*[START],[END]*
Saves program to file containing built-in initialization routine (program must end with RTS instruction); executes automatically when DOS 2.0S booted

l. **BINARY LOAD**
Loads a binary file created by DOS option K or one created by the Assembler/Editor cartridge; PROGM1.OBJ/N /N to block execution

m. **RUN AT ADDRESS**
In hex; with RTS instruction, control returns

n. **CREATE MEM.SAV**
Saves and stores contents of memory area used by DOS menu program (exit menu to restore MEM.SAV contents to memory) (DOS 2.0 only)

o. **DUPLICATE FILE**
Copies files between diskettes using one drive. (Wild cards okay) (DOS 1.0 erases program in memory)

Program Development

Line numbers range from 0 to 32767. Logical line contains 114 characters. Colons separate multiple statements:

LIST	Displays first line through last line (BREAK halts listing)
LIST *100*	Displays line *100*
LIST *100,860*	Displays program lines *100-860* stored in memory
NEW	Clears memory for new program (deletes old program and variables)
RUN	Executes program (SYSTEM RESET and BREAK halt program execution, return computer to Immediate mode)
STOP	Halts program execution, returns computer to Immediate mode
END	Halts like STOP but voices off, I/O channels 1-7 closed; not needed at end of program

CONT — Restarts program on next line after mid-line BREAK, STOP, END; restarts but doesn't execute statement that caused error

PRINT FRE(*0*) — Displays number of bytes available in RAM for program; 0 can be any number

Program Files

Command	Loads	Saved By	Format	Effect	Comments
CLOAD	All	CSAVE	Tokens	NEW	Cassette only; faster than LOAD
ENTER	All/part	LIST	ATASCII	Merges program lines	Invoke NEW before ENTER to erase program and variables already in memory
LOAD/RUN	All	SAVE	Tokens	NEW	

To Transfer Program to Cassette:

CSAVE — Saves program and its Variable Name Table (VNT) in token format (read with CLOAD)

SAVE "C:" — Saves programs like CSAVE; read with LOAD or RUN

LIST "C:",*90,900* — Saves lines *90* to *900* in ATASCII code (read with ENTER)

BREAK — Halts transfer to cassette

To Transfer Program from Cassette to Memory:

CLOAD — Loads program saved with CSAVE or SAVE "C:"; invokes NEW

LOAD "C:" — Loads program and VNT saved with SAVE; invokes NEW

RUN "C:" — Loads program and VNT saved with SAVE; invokes NEW and executes after loading

300 RUN "C:" — Runs next program on tape (chaining); pressing RETURN at beep deletes from memory VNT and program containing line 300

ENTER "C:" — Loads lines saved with LIST; adds lines to program and VNT in memory (halt with BREAK)

To Transfer Program to Disk:

SAVE "D:*PROG1*" Saves PROG1 and its VNT to drive #1 in token format (read with LOAD or RUN)

LIST "D:*PROG1.LST*",*90*,*900* Saves lines *90* to *900* in ATASCII code (read with ENTER)

BREAK Halts transfer to diskette

To Transfer Program from Disk to Memory:

LOAD "D:*PROG1*" Loads *PROG1* and VNT saved with SAVE; invokes NEW

RUN "D:*PROG1*" Loads programs like LOAD, but executes after loading

300 RUN "D:*PROG2*" Loads/runs *PROG2* (chaining); deletes VNT and program containing line *300* from memory

ENTER "D:*PROG.LST*" Loads program lines saved with LIST; merges lines with programs and VNT in memory; (halt with BREAK; memory not erased)

To Save and Load Subroutines and Subroutine Libraries:

LIST "C:" Saves subroutine

ENTER "C:" Merges saved subroutine with program in memory

LIST "D:*SUBR1*" Stores subroutine *SUBR1* in library

ENTER "D:*SUBR1*" Merges *SUBR1* with program in memory

ATARI BASIC Language

Initializing Statements

Variable Name Table (VNT):

Room for 128 variable/array names. NEW clears VNT (CLR doesn't). Invoked by CLOAD, LOAD, RUN. ENTER adds variables to existing VNT; CSAVE and SAVE save VNT; LIST doesn't (to clear VNT but not program: LIST program to cassette or diskette, invoke NEW, then ENTER program).

Variable Names:

First character must be a capital letter; capitals or digits for rest of name. Up to 114 characters; string variables end with $. Dimension numeric array variables and string variables before use:

CLR	Undimensions variables; resets pointer to start of DATA list (use before redimensioning)
DIM A$(24)	Assigns 25-character string (0 to 24) to A$
DIM X(*10*)	Assigns up to 11 array elements indexed 0<10, to *X*; COM is the same as DIM
DIM Y(*3,2*)	*Y* is two-dimensional array indexed 0-3, 0-2; dimensions *12* array elements (3 groups of 4)

DATA and READ can assign values to multiple variables:

DATA *"May '84", 2+2, 7*	Contains three values: *"May '84"*, *2+2*, and *7*.
READ DATE$, SUM$, NUM	Reads DATA above; causes error if DATA statement does not contain at least three elements
DATA *1.4E-2,,2/3,*	Contains 4 values (2nd and 4th values are null string constants); scientific notation okay for numeric variables
DATA *"May, 1984"*	Contains the two values *"May* and *1984"*
DATA *5,10:READ X,Y*	Contains three values: *5*, *10:READ X*, and *Y* (statement following DATA on same line treated as data)
READ A: READ B	If READ A reads item 4 in a DATA statement list, READ B reads item 5
RESTORE	Resets pointer to first value listed in first DATA statement in program
RESTORE *140*	Resets pointer to DATA statement on line *140*

Numbers

Numeric Constants:

Floating (decimal) point numbers (no commas); can have nine significant digits plus two-digit exponent; range from −9.99999999E+97 to 9.99999999E+97 (numbers nearer 0 than +/−9.99999999E−98 are set to 0). Use scientific notation if 10 digits before decimal, if nearer 0 than +/0.01: (2E+02=2000.001=1E−03 −12345E+08=−123450000−1.23E−08=−0.0000000123).

Math Functions and Statements

Y=ABS(X)	Absolute value of X
Y=ATN(X)	Arctangent of X; −PI/2 to PI/2
Y=CLOG(X)	Common logarithm (base 10) of X
Y=COS(X)	Cosine of X radians; X degrees with DEG
Y=EXP(X)	*e* (2.71828179) to the X power
Y=INT(X)	Largest integer <= X
Y=LOG(X)	Natural logarithm (base *e*) of X
Y=RND(X)	Random number (0 to 1)
Y=SGN(X)	Sign of X
Y=SIN(X)	Sine of X radians; X degrees with DEG
Y=SQR(X)	Square root of X
DEG	Arguments of trigonometric functions interpreted in degrees
RAD	Arguments interpreted as radians (default condition)

Expressions Valid	Invalid
A+B, A=B, A AND B	A$=B, A$+B$, A$ AND B$
CHR$(10)<A$	CHR$(10)<CHR$(14)
STR$(2 3<>A$	STR$(2 3)<>STR$(3 4)

Numeric Operators			
	Precedence	**Operator**	**Meaning**
	High 9	( )	Parentheses denote order of evaluation
Arithmetic Operators	8	^	Exponentiation
	7	-	Unary Minus
	6	*	Multiplication
	6	/	Division
	5	+	Addition
	5	-	Subtraction
Relational Operators	4	=	Equal
	4	< >	Not equal
	4	<	Less than
	4	>	Greater than
	4	< =	Less than or Equal
	4	> =	Greater than or Equal
Boolean Operators	3	NOT	Logical complement
	2	AND	Logical AND
	Low 1	OR	Logical OR

Boolean Truth Table
The AND operation results in a 1 only if both values are 1.
1 AND 1 = 1 1 AND 0 = 0
0 AND 1 = 0 0 AND 0 = 0
The OR operation results in a 1 if either value is 1.
1 OR 1 = 1 1 OR 0 = 1
0 OR 1 = 1 0 OR 0 = 0
The NOT operation logically complements each value.
NOT 1 = 0 NOT 0 = 1

Strings

Strings can be up to 114 characters, can use any ATASCII characters, are enclosed by quotation marks (*e.g.*, "SUBTOTAL" "ACCOUNT 4019-181"):

CHR$(34) Assigns quotation mark; *e.g.*, PRINT "CHR$(34); "SPEED KILLS"

ESC/CTRL← Assigns "cursor left" to a string and echoes ← (subsequent PRINT statement moves cursor left)

ESC/ESC Prints echoed character when entered before escape-sequence character

PRINT CHR$(30) Displays ←

Escape Sequences			
Keystroke	**Echoed Character**	**ATASCII Code**	**String Character**
ESC\ESC		27	Escape code
ESC\BACK S		126	Cursor left, replace with blank space
ESC\TAB		127	Cursor right to next tab stop
ESC\CTRL--		28	Cursor up
ESC\CTRL-=		29	Cursor down
ESC\CTRL-*		30	Cursor right
ESC\CTRL-+		31	Cursor left
ESC\CTRL-BACK S		254	Delete character
ESC\CTRL->		255	Insert character
ESC\CTRL-<		125	Clear screen
ESC\CTRL-TAB		158	Clear tab stop
ESC\CTRL-2		253	Sound built-in speaker
ESC\SHIFT-BACK S		156	Delete line
ESC\SHIFT->		157	Insert line
ESC\SHIFT-<		125	Clear screen
ESC\SHIFT-TAB		159	Set tab stop

Substrings and Concatenation:

A$ is the five-character string ABCDE:

A$(2,4)	Specifies substring BCD; 2nd through 4th characters
A$(2)	Specifies same substring as A$(2,5); 2nd through last character
A$(2>1,X)	Specifies substring ABC if X=3 (expressions and variables okay)

To concatenate A$ and B$, use LEN to make B$ a substring of A$ (*e.g.*, following program); concatenated string can be any length:

```
10 DIM A$(15), B$(9), C$(5)
20 A$="ATARI ": B$="DISK": C$="GUIDE"
30 B$ (LEN(B$)+1)=C$
40 A$ (LEN(A$)+1)=B$
50 PRINT A$
```

The preceding program prints ATARI DISKGUIDE.

String Functions:

Conversion to and from strings:

ASC(*"ABC"*)	Returns *65*, ATASCII code of first character in string
CHR$(*65*)	Returns "*A*", string value of ATASCII code
STR$(*14*)	Returns "*14*", string equivalent of numeric value
STR$(*2/3*)	Returns "*0.6666666666*"
VAL(*"14"*)	Returns numeric value *14*, equivalent of numeric string; VAL's argument must be in acceptable numeric format
ADR(*string*)	Returns address of first character in *string*
A=USR(ADR(*SUBR$*))	USR executes machine code in *SUBR$*
LEN(*"A BC"*)	Returns *4*, the number of characters in the string

BASIC Programming Statements

Branch Statements:

GOTO *100*	Branches to line *100*
GOTO *3*A*	Calculates line number from expression

ON*A+1* GOTO*50,80,110* Computes *A+1*; must be numeric expression between 0 and 255

A	A+1	Go To Line
0	1	50
1	2	80
2	3	110

Create loops with FOR/NEXT; terminate innermost loop first; avoid branching out of loop (wastes memory).

FOR N=0 TO 99 STEP 2 N will have values of 0,2,4, ... 98 (default=STEP 1)

FOR N=99 TO 0 STEP-1 N will have values of 99, 97...0

FOR N=A*3 TO 70 STEP A>4 Uses expressions for index values and step sizes

NEXT Paired with FOR statement (FOR invalid without it)

70 GOSUB A**1000* Calls subroutine at line 2000 if A=7 (following program); 70 ON A GOSUB 1000,2000 can replace line 70 in program; A can be 0-255; calls subroutine at line 2000 if A=2

RETURN Branches from subroutine to line 80 (line following subroutine call)

POP Erases last RETURN address; used when exiting subroutine with GOTO instead of RETURN; prevents accumulation of unused addresses

```
70 GOSUB A*1000
 :
80 PRINT "RETURNED"
 :
2000 REM Subroutine
 :
2050 RETURN
```

Nesting:

A subroutine can call another subroutine. Recursion is invalid:

TRAP *20000* Branches to error handling subroutine at *20000*; *see* p. 42

PRINT USR(*1536*) Branches to machine-language program at memory location 1536; *see* p. 42

IF *A=B+5* THEN PRINT *MSG$* If the expression *A*=B+5 is true, *MSG$* is printed; executes next line if false

IF *Q<14* THEN *IN=0*: GOSUB *2000* If expression *Q<14* is true, *IN=0* sets IN to 0 and branches to subroutine on line *2000*

IF *MM$=DD$* THEN *100* If expression *MM$*=DD$ is true, begin execution at line 100; GOTO implied

IF *A$="X"* THEN IF *B=2* THEN *50* If both expressions are true, begin execution at line *50*; equivalent Boolean expression can be used instead: IF A$="X" AND B=2 THEN 50

General Programming Statements

BYE Switches from BASIC to Memo Pad mode; SYSTEM RESET restores program and VNT

CONT Resumes program on next line if halt in middle of line

DATA Creates list of values (READ assigns them to variables); *see* p. 12

END Halts program

X=0 Sets value of X (LET optional)

LET EXP=*2* Allows BASIC keyword as variable name

NEW Deletes current program and variables

READ Assigns values from DATA statements to variables; *see* p. 12

REM Treats text to right of REM as remarks; not executed

RESTORE *140* Restores data pointer to 1st item on line *140*

RESTORE Restores data pointer to 1st DATA statement in program

STOP Stops program execution; prints line number

SYSTEM RESET Key initializes system (data may be lost)

TRAP *20000* Branches to error handling subroutine at line *20000* if any error is encountered after statement

To Halt Program Execution	BREAK	END	STOP	SYSTEM RESET
Immediate mode restored	x	x	x	x
Resume execution with CONT	x	x	x	
I/O channels 1-7 cleared, sound turned off		x		x

Keyboard Input

Input data can be numeric or string values; must end with RETURN; Screen echoes entry:

INPUT *A,B* — Accepts two numeric values separated by comma on one line or on two lines (RETURN after each)

INPUT *A$,B$* — Accepts two string values on two lines only

80 PRINT *"HOW MUCH IS ";N*9;* (semicolon at end of line suppresses carriage return)

90 INPUT ANS — Waits for response with cursor on same line as prompt message

GET*#1,A* — Assigns *5* to *A* if *5* entered; specifies input from channel *1* (channel must be opened with OPEN first); entry doesn't echo or require RETURN; entry can be one ATASCII character only (see following program)

```
10 OPEN #1,4,0,"K:"
20 PRINT "Are you there?"
30 PRINT "Press RETURN if so."
40 GET #1,R
50 IF R<>155 THEN 40
60 PRINT "OK, let's get on with it."
```

Graphics

Graphics Modes Summary

Display Type	Graphics Mode Number	Colors Available	Screen Size (columns × rows)	Foreground Color Register(s)	Background Color Register	Border Color Register	Register Selection With COLOR	Memory Used (Bytes)
Normal text	0	1 color, 2 luminances	40×24	1 (Color is not selectable)	2	4	*	992
Double-width text	1	5	20×20 (split) 20×24 (full)	0, 1, 2, 3	4	4		674 (split) 672 (full)
Double-width, double-height text	2	5	10×20 (split) 12×20 (full)	0, 1, 2, 3	4	4		424 (split) 420 (full)
Four-color graphics	3	4	40×20 (split) 40×24 (full)	0, 1, 2	4	4	COLOR 1: register 0 COLOR 2: register 1 COLOR 3: register 2 COLOR 0: register 4	Mode 3: 434 (split) 432 (full) Mode 5: 1174 (split) 1176 (full) Mode 7: 4190 (split) 4200 (full)
	5	4	80×40 (split) 80×48 (full)	0, 1, 2	4	4		
	7	4	160×80 (split) 160×96 (full)	0, 1, 2	4	4		
Two-color graphics	4	2	80×40 (split) 80×48 (full)	0	4	4	COLOR 1: register 0 COLOR 0: register 4	Mode 4: 694 (split) 696 (full) Mode 6: 2174 (split) 2184 (full)
	6	2	160×80 (split) 160×96 (full)	0	4	4		
High-resolution graphics	8	1 color, 2 luminances	320×160 (split) 320×192 (full)	1 (color is not selectable)	2	4	COLOR 1: register 1 COLOR 0: register 2	8112 (split) 8138 (full)

* In Mode 0, COLOR will accept an ATASCII character to plot with. For example, COLOR ASC("!") in Mode 0, followed by PLOT or DRAWTO statements, will place ! characters on the screen.

Color Register Assignments, Graphics Modes 1 and 2			
Number	**Color**	**Number**	**Color**
0	Grey	8	Light Blue
1	Gold	9	Blue-Green
2	Orange	10	Aqua
3	Red	11	Green-Blue
4	Pink	12	Green
5	Violet	13	Yellow-Green
6	Blue-Purple	14	Orange-Green
7	Blue	15	Orange

Color Numbers Used with SETCOLOR			
Register Number	**Color Number**	**Luminance Value**	**Actual Screen Color**
0	2	8	Orange
1	12	10	Aqua
2	9	4	Blue
3	4	6	Light Red
4	0	0	Black

Default (Preassigned) Color Register Setting		
Characters	**ATASCII Values**	**Color Register Assigned**
Upper-case alphabet (A-Z), numbers, special characters (! $ + -)	32-90 160-218	Normal: 0 Inverse*: 2
Lower-case* alphabet	61-122 225-250	Normal: 1 Inverse*: 3
*Lower-case and inverse characters display as normal, upper-case text. They are assigned to different registers, however.		

GRAPHICS 0	Selects normal text mode; opens channel #6 for output to screen
GRAPHICS *n*	Selects graphics mode *n* (*n*=1-8) with four-line text window; clears screen
GRAPHICS *n*+16	Provides full-screen graphics, suppresses window
GRAPHICS *n*+32	Suppresses screen clearing
GRAPHICS *n*+48	Suppresses window and screen clearing
CLOSE *#6*	Closes channel *#6* after GRAPHICS *n* opens it

Character Display:

LOCATE *3,12,X*	Assigns X the ATASCII code of character X at column *3*, row *12* (modes 0-2); assigns X the color register (modes 3-8)
POSITION *10,3*	Begins subsequent PRINT statement at column *10*, row *3*; also subsequent GET, PUT, INPUT, LOCATE
PRINT *"text"*	Uses window in modes 1-8, displaying "*text*" in graphics mode 0
INPUT *"text"*	Uses window

Graphics Modes 1 and 2:

PRINT #6, *"text"*	Displays double-width characters in graphics window; bits 7 and 8 determine character color; bits 1 through 6 define character

```
 5 REM GRAPHICS MODE 1 DEMO
10 GRAPHICS 1    ;Selects graphics mode 1, standard character set
20 POKE 756,226  ;Selects alternate character set
30 PRINT #6
40 PRINT #6      ;"you've got to have"
50 PRINT " ART:  ;GRAPHICS DEMO"
60 COLOR 0       ;Produces " " in color register 1; see Color, page 23
70 PLOT 1,2      ;Displays " " at column 1 row 2
80 DRAWTO 18,2   ;Uses " "'s to draw line from 1,2 to 18,2
```

COLOR and Graphics Modes 1 and 2
(Characters and color registers selected by values of COLOR *numexpr*)

Value for Color Register				Character*		Value for Color Register				Character*	
0	1	2	3	Std.	Alt.	0	1	2	3	Std.	Alt.
32**	0	160	128			51	19	179	147		
33	1	161	129			52	20	180	148		
34	2	162	130			53	21	181	149		
35	3	163	131			54	22	182	150		
36	4	164	132			55	23	183	151		
37	5	165	133			56	24	184	152		
38	6	166	134			57	25	185	153		
39	7	167	135			58	26	186	154		
40	8	168	136			59	27	187	None†		
41	9	169	137			60	28	188	156		
42	10	170	138			61	29	189	157		
43	11	171	139			62	30	190	158		
44	12	172	140			63	31	191	159		
45	13	173	141			64	96	192	224		
46	14	174	142			65	97	193	225		
47	15	175	143			66	98	194	226		
48	16	176	144			67	99	195	227		
49	17	177	145			68	100	196	228		
50	18	178	146			69	101	197	229		

Note:

* For standard characters, POKE 756,224. For alternate characters, POKE 756,226.

** 155 selects the same character and color register as value 32.

† No value selects this color register character combination.

COLOR and Graphics Modes 1 and 2
(Characters and color registers selected by values of COLOR *numexpr*)

Value for Color Register				Character*		Value for Color Register				Character*	
0	1	2	3	Std.	Alt.	0	1	2	3	Std.	Alt.
70	102	198	230	F	f	83	115	211	243	S	s
71	103	199	231	G	g	84	116	212	244	T	t
72	104	200	232	H	h	85	117	213	245	U	u
73	105	201	233	I	i	86	118	214	246	V	v
74	106	202	234	J	j	87	119	215	247	W	w
75	107	203	235	K	k	88	120	216	248	X	x
76	108	204	236	L	l	89	121	217	249	Y	y
77	109	205	237	M	m	90	122	218	250	Z	z
78	110	206	238	N	n	91	123	219	251	[	♠
79	111	207	239	O	o	92	124	220	252	\	|
80	112	208	240	P	p	93	None†	221	253	]	↖
81	113	209	241	Q	q	94	126	222	254	^	◀
82	114	210	242	R	r	95	127	223	255	_	▶

Note:

* For standard characters, POKE 756,224. For alternate characters, POKE 756,226.

** 155 selects the same character and color register as value 32.

† No value selects this color register/character combination.

Display of Points, Lines, and Solid Areas in Modes 3-8:

PLOT *5,15* Displays dot of color at column *5*, row *15*; gets register from last COLOR statement

DRAWTO *5,15* Draws a line from the point displayed by the last PLOT or DRAWTO statement to column *5*, row *15*; staircases on diagonal

LOCATE *5,15,X* Assigns to X the color register number of the point at column *5*, row *15*

POSITION *5,15* Moves cursor to column *5*, row *15* at next GET, PRINT, PUT, INPUT, LOCATE

XIO *18,#6,0,0,"S:"* Fills the screen or shape with solid color

```
 5 REM DRAWS SHAPE AND FILLS IT
10 GRAPHICS 21              ;Graphics mode 5, no window
20 COLOR 3:POKE 765,2       ;Border color 3, fill color 2
30 PLOT 70,40               ;Start shape
40 DRAWTO 50,10
50 DRAWTO 30,5
60 POSITION 10,40           ;Finish shape
70 XIO 18,#6,0,0,"S:"       ;Fills shape with horizontal lines from upper left
```

Color:

SETCOLOR Changes color-register default hue and luminance values

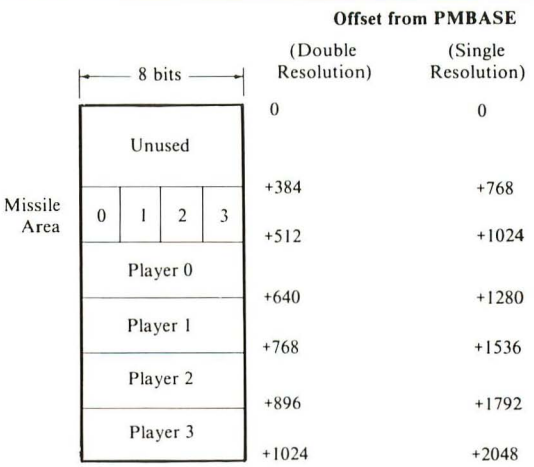
COLOR Specifies color register in graphics modes 3-8, register and character in graphics modes 0-2

SETCOLOR *0,4,14* Sets hue of color registers *0* to *4* (pink); sets luminance to *14* (14 is brightest; 0 is darkest; only even-numbered settings are meaningful)

COLOR *2* Determines that next PLOT or DRAWTO will use:

- Color register 1 in graphics modes 3, 5, or 7
- CHR$(2) in graphics mode 0
- CHR$(2) (standard character)
- CHR$(34) (alternate character using color register 1 in graphics modes 1 and 2

Player-Missile Graphics Table Layout



Note: All locations shown are offsets from the start of the player-missile graphics table base address (PMBASE).

Player-Missile DMA Control Register Values	
Value to POKE	**Setting Which Results**
4	Enable Missile DMA only
8	Enable Player DMA only
12	Enable Player-Missile DMA
Add 16	Single-line resolution (double-line resolution = default)

Playfield and Player-Missile Color Register Values					
Color Value			Color Value		
Color	Decimal	Hex	Color	Decimal	Hex
Grey	0	0	Blue	128	80
Gold	16	10	Light Blue	144	90
Orange	32	20	Turquoise	160	A0
Red	48	30	Green-Blue	176	B0
Pink	64	40	Green	192	C0
Violet	80	50	Yellow-Green	208	D0
Blue-Purple	96	60	Orange-Green	224	E0
Blue	112	70	Light Orange	240	F0
Add an even number, 2 to 14, to set luminance; 0=no luminance, 14=maximum luminescence.					

Sound Generation

SOUND Statement Controls TV Speaker:

SOUND *voice, pitch, distortion, volume*

voice	0-3 (four voices)
pitch	0 (highest note) to 255 (lowest note)
distortion	0-15; 10 and 14 produce pure tones; other even numbers add noise to pure tones; odd numbers turn off sound
volume	0 (no sound) to 15 (full volume)

All four voices turned off by pressing SYSTEM RESET or entering:

CLOAD, CSAVE, DOS, END, ENTER, LIST (except to display screen), LOAD, NEW, RUN, SAVE.

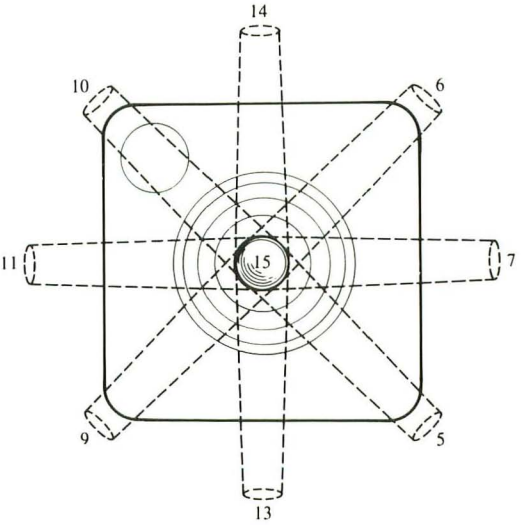
Device I/O

Joysticks

STICK(*3*) Returns integer identifying position of stick *#3*

STRIG(*3*) Returns 0 if *#3* stick trigger being pressed, 1 if released

STICK Function Values



PADDLE(*3*) Returns 1-228 identifying position of paddle *#3*; full clockwise is 1, full counterclockwise is 228

PTRIG(*3*) Returns 0 if paddle *#3* trigger is pressed; 1 if released

Printer Port

The ATARI 820/822 Printer attaches directly to the serial bus, and the ATARI 825 Printer is connected to the parallel port of ATARI 850 Interface Module. ATARI 820/822 has automatic EOL at character 40 (at 39 on screen but printer determines wraparound point); ATARI 825 has automatic EOL at character 80. ATARI 825 uses some ATASCII codes as control characters:

LIST "P:",*10,100*	Sends lines *10-100* to printer
LPRINT	Sends EOL character to printer (default)
LPRINT *"hits" "runs,"*	Prints "*hits*" at column stop 1, "*runs*" at 11, outputs next PRINT statement at 21 (delete comma or semicolon at end of line with ATARI 825)
PRINT #6,*"hits", "runs"*	Prints like LPRINT statement; requires OPEN #6,8,0,"P:"
PUT #6,*7*	Prints *7* (single numeric value, modulo 256); requires OPEN #6,8,0,"P:"

Standard Printer Character Set Summary

Decimal Code	Character	
	ATARI 820/822 Printer	ATARI 825 Printer
0-31	Space	Control characters
32-95	Same as display screen	Same as display screen
96		
97-122	Same as display screen	Same as display screen
123	¦	¦
124	{	{
125	}	}
126	~	~
127	Space	Non-printing

To print sideways characters on the 820 printer with PRINT# or PUT# statements, open the printer channel with: OPEN #4,8k83,"P:", then use PRINT #4 or PUT#4 to direct the output. The character set is slightly different. Decimal 48-95 correspond to the normal character set, but decimal 96-127 repeat decimal 64-95. There are 29 characters per line.

Printer Control Characters (ATARI 825 Printer)			
Decimal Code(s)	**Graphics Character(s)**	**Keystroke(s)**	**ATARI 825 Printer function**
10		CTRL-J	Line feed
27 & 10		ESC\ESC & CTRL-J	Reverse line feed
27 & 28		ESC\ESC & ESC\CTRL - –	Half-line feed
27 & 30		ESC\ESC & ESC\CTRL - +	Reverse half-line feed
13		CTRL-M	Carriage return with automatic line feed
15		CTRL-O	Start underlining
14		CTRL-N	Stop underlining
27 & 14		ESC\ESC & CTRL-N	Start double-wide printing
27 & 15		ESC\ESC & CTRL-O	Stop double-wide printing
27 & 19		ESC\ESC & CTRL-S	Select standard (10 cpi) characters
27 & 20		ESC\ESC & CTRL-T	Select condensed (16.7) characters
27 & 17		ESC\ESC & CTRL-Q	Select proportionally sized characters
27 & 1		ESC\ESC & CTRL-A	One dot blank space
27 & 2		ESC\ESC & CTRL-B	Two dot blank spaces
27 & 3		ESC\ESC & CTRL-C	Three dot blank spaces
27 & 4		ESC\ESC & CTRL-D	Four dot blank spaces
27 & 5		ESC\ESC & CTRL-E	Five dot blank spaces
27 & 6		ESC\ESC & CTRL-F	Six dot blank spaces
8 & *nn**		CTRL-H & *keystroke**	Backspace *nn** dots

* The character that follows the backspace control character (ASCII code 8) is interpreted as the number of dots to backspace. Use Appendix D to select the *keystroke* which produces the ATASCII character whose code number equals *nn*, the number of dots to backspace.

General Device I/O

OFEN #*chan, task, aux, dev* Use before accessing an external device for input or output; see following tables for parameter values; be sure channel to be OPENed is already closed (use CLOSE statement)

CLOSE #*6* Closes Channel *6*

OPEN Parameter Number 2 (*auxexpr*)		
Device	**Function Description**	**Value**
Program recorder (C:)	Normal inter-record gaps	0
	Short inter-record gaps	128
Disk drive (D[*n*]:*filename* [.*ext*])	Ignored	0
Screen editor (E:)	Ignored	0
Keyboard (K:)	Ignored	0
Printer (P:)	Normal characters	0
	Sideways characters (ATARI 820)	83
RS-232 serial port (R[*n*]:)	Ignored	0
Screen display (S:)	BASIC graphics mode 0	0
	BASIC graphics mode 1	1
	BASIC graphics mode 2	2
	BASIC graphics mode 3	3
	BASIC graphics mode 4	4
	BASIC graphics mode 5	5
	BASIC graphics mode 6	6
	BASIC graphics mode 7	7
	BASIC graphics mode 8	8

OPEN Parameter Number 1 (*taskexpr*)

Device	Task Number	Task Description			
Program recorder (C:)	4	Read			
	8	Write			
Disk file (D[n]:*filename* [.*ext*])	4	Read			
	6	Read disk directory			
	8	Write — new file			
	9	Write — append			
	12	Read and write — update			
Screen editor (E:)	8	Screen output			
	12	Keyboard input & screen output			
	13	Screen input & output			
Keyboard (K:)	4	Read			
Printer (P:)	8	Write			
RS-232 serial port (R[*n*]:)	5	Concurrent read			
	8	Block write			
	9	Concurrent write			
	13	Concurrent read and write			
		Clear Screen	Text Window†	Read	Write
Screen display (S:)	8	Yes	No	No	Yes
	12	Yes	No	Yes	Yes
	24	Yes	Yes	No	Yes
	28	Yes	Yes	Yes	Yes
	40	No*	No	No	Yes
	44	No*	No	Yes	Yes
	56	No*	Yes	No	Yes
	60	No*	Yes	Yes	Yes

* Screen always cleared in graphics mode 0. † No separate text window in graphics mode 0.

Channel I/O (*Chan*)

0 Screen editor (E:), command input from keyboard (K:), simple INPUT/PRINT statements, LIST and PRINT statements to screen display (S:)

1-5 BASIC program

6 ATARI BASIC screen graphics statements,

7 Some printer operations, cassette save/load, programs that don't save/load or use LPRINT, LIST statements except to S:

PRINT #6;*A*$;*A*	Outputs *A*$ concatenated to *A* and EOL character to device #6
PRINT #6	Outputs EOL character only (default)
PUT #6,X	Outputs X (0-255 or modulo 256) to device *#6*
LIST "*C:*",*100*,*200*	Outputs lines *100* to *200* to Cassette (*C:*) without OPEN statement
INPUT #6,*A*$	Accepts entry from device *#6* (entry must end with RETURN [from keyboard] or EOL [CHR$(155)])
GET #6,*X*	Accepts entry to variable *X* from device linked to *#6*; if S: or E:, value returned is code of character or color register at cursor
ENTER "C:"	Inputs from cassette (C:) without prior OPEN statement

Serial Port (850 Interface)

For serial port use, turn on the disk drive through the 850 Interface Module, then the console. The disk drive must boot the RS-232 serial device handler into memory (AUTORUN.SYS). Don't turn the module off during the session unless only the ATARI 825 printer is attached.

XIO *cmd, #chan, num1, num2, dev* Conditions serial port after RS-232 device handler is in memory; see following table for summary of first three parameters

Summary of XIO Commands			
Action	**cmd**	**num1**	**num2**
Output partial block	32	0	0
Control DTR, RTS, XMT	34	see X10 34 table	0
Baud rate, word size, stop bits, & ready monitoring	36	see X10 36 table	see X10 36 table
Translation mode	38	see X10 38 table	ATASCII code
Concurrent mode	40	0	0

XIO 34 (Serial) Parameter *numexpr*$_1$							
Add one number from each column to get value of *numexpr*$_1$			**Selected values of *numexpr*$_1$**				
	DTR	**RTS**	**XMT**	**DTR**	**RTS**	**XMT**	**Value**
No change	0	0	0	Off	Off	0	162
Turn off (XMT to 0)	128	32	2	Off	Off	1	163
				Off	On	0	178
Turn on (XMT to 1)	192	48	3	Off	On	1	179
				On	Off	0	226
				On	Off	1	227
				On	On	0	242
				On	On	1	243

XIO 36 (Serial) Parameters *numexpr*$_1$ and *numexpr*$_2$											
numexpr$_1$ (Add one value from each column)							*numexpr*$_2$			
Stop Bits	**Value**	**Word Size**	**Value**	**Baud Rate**	**Value**		**DSR**	**CTS**	**CRX**	**Value**
1	0	8 bits	0	300	0		No	No	No	0
2	128	7 bits	16	45.5	1		No	No	Yes	1
		6 bits	32	50	2		No	Yes	No	2
		5 bits	48	56.875	3		No	Yes	Yes	3
				75	4		Yes	No	No	4
				110	5		Yes	No	Yes	5
				134.5	6		Yes	Yes	No	6
				150	7		Yes	Yes	Yes	7
				300	8					
				600	9					
				1200	10					
				1800	11					
				2400	12					
				4800	13					
				9600	14					
				9600	15					

XIO 38 (Serial) Parameter *numexpr*$_1$							
Add one value from each column							
Line Feed Append	**Value**	**Translation ATASCII — ASCII Mode**	**Value**	**Input Parity Mode**	**Value**	**Output Parity Mode**	**Value**
No	0	Light	0	Ignore	0	No change	0
Yes*	64	Heavy	16	Odd†	4	Odd	1
		None	32	Even†	8	Even	2
				Ignore†	12	Bit on	3

* Line feed character appended after carriage return (ATASCII EOL).
† Check parity as indicated, then clear parity bit.

Data Files

Sequential Access to Data Files:

Buffer holds 128 characters; channels #1-5 available.

Cassette Data Files

To Create the File on Tape:

OPEN *#1,8,0,"C:"* Opens channel *#1* for output *8* to file (always *0*) on cassette *"C"*; cues tape; writes 20-second leader

PRINT #1;*10*;CHR$(155);*100*;CHR$(155);*"ETC."* Writes *10* and *100* (numeric values) and string value to file in ATASCII; outputs automatic EOL; uses CHR$(155) to signal mid-line EOL (prevents concatenation of values); uses semicolons (not commas) to compress data

PUT #1,*9* Writes a single numeric value *9*: range 0-255 (values can be interpreted as ATASCII codes); does not output EOL

CLOSE *#1* Outputs data in partially full buffer to #1, then closes *#1*

To Access the File on Tape (Read Values to Variables):

OPEN *#1,4,0,"C:"* Opens channel *#1* for input (4) to memory (always *0*) from cassette ("C:"); cues tape; reads 20-second leader

INPUT #1, A,B$ Reads values written by PRINT#1 statement above

GET #1,D Reads numeric value written by PUT #1 statement above

CLOSE *#1* Closes channel #1

Disk Data Files

To Create or Add to the Disk File:

OPEN *#1,8,0,"D:FILENAME.EXT"* Opens channel *#1* for output (*8*) to file (always *0*) on *D1* (default); deletes *FILENAME.EXT* if it exists

OPEN *#1,9,0,*"D:FILENAME.EXT" Opens channel #1 to existing FILENAME.EXT for appending data

PRINT #1;10;CHR$(155);100;CHR$(155);"ETC." Writes *10* and *100* (numeric values) and string value to file in ATASCII; outputs automatic EOL; uses CHR$(155) to signal midline EOL characters (prevents concatenation of values); uses semicolons (not commas) to compress data

PUT #1,9 Writes a single numeric value *9*: range 0-255 (values can be interpreted as ATASCII codes); does not output EOL

CLOSE #1 Forces output of data in partially full buffer to *#1*, then closes *#1*

To Access the File (Read Values to Variables):

OPEN #1,4, 0, "D:FILENAME.EXT" Opens *#1* for input (*4*)

INPUT #1,A,B,C$ Reads values written by *PRINT #1* statement above

GET #1,D Reads numeric value written by *PUT #1* statement above

```
10 OPEN #1,12,0,"D:FILE"    ;Opens channel #1 for update (12)
20 GET #1,3                 ;Reads first character in FILE
30 PRINT #1;"ATARI"         ;Replaces characters 2-6 in FILE
```

Random Access to Disk File (DOS 2.0S Only)

NOTE #1,S,C Determines current location of pointer in OPENed file on channel *#1*; assigns sector number (1-719) to numeric variable *S*, character number within sector (0-125) to *C*)

POINT #1,S,C Moves pointer to sector *S* character *C*; S and C must be variables

Error Messages

2 Out of Memory
3 Bad Value
4 Too Many Variables
5 String Length Exceeded
6 DATA List Exhausted
7 Number Greater Than 32767
8 INPUT Statement Type Mismatch
9 Array or String Dimension Error
10 Expression Too Complex
11 Numeric Overflow/Underflow
12 Line Not Found
13 NEXT Without FOR
14 Line Too Long
15 GOSUB or FOR Line Deleted
16 RETURN Without GOSUB
17 Undecipherable Statement Encountered
18 Invalid String Character
19 Program Too Large
20 Bad Channel Number
21 Not LOAD Format
128 Break Abort
129 Channel Already Open
130 Unknown Device
131 Output Only
132 XIO Syntax Error
133 Channel Not Open
134 Unknown Channel Number
135 Input Only
136 End of File
137 Record Truncated
138 Device Does Not Respond
139 Device Malfunctions or Refuses Command
140 Framing Error on Serial Bus
141 Cursor Out of Range
142 Data Frame Overrun
143 Data Frame Checksum
144 Disk Error
145 Read-After-Write Compare Error, or Bad Screen Mode
146 Function Not Implemented
147 Insufficient RAM for Graphics Mode
150 Serial Port Open
151 Concurrent Mode Error
152 Concurrent Mode Buffer Error
153 Concurrent Mode Active
154 Concurrent Mode Inactive
160 Drive Number Unknown
161 Too Many Files Open
162 Disk Full
163 Unrecoverable System Error
164 File Number Mismatch
165 Bad File Name
166 POINT Data Length Error
167 File Locked
168 Unknown XIO Command
169 Directory Full
170 File Not Found
171 POINT Invalid

Error Handling

PEEK(195) Retrieves error code
PEEK(187)*256+PEEK(186) Returns line number with error

TRAP *20000* Disables auto error handling; when executed before error it sets flag causing branch to error handling subroutine at line *20000* when error occurs; the error clears flag

TRAP *40000* Turns off current TRAP

Musical Notes for Pitch Values
(Values of *pitchexpr* in SOUND statement)

Value	Note	Value	Note
29	C	91	F
31	B	96	E
33	$A^{\#}/B^{b}$	102	$D^{\#}/E^{b}$
35	A	108	D
37	$G^{\#}/A^{b}$	114	$C^{\#}/D^{b}$
40	G	121	C
42	$F^{\#}/G^{b}$	128	B
45	F	136	$A^{\#}/B^{b}$
47	E	144	A
50	$D^{\#}/E^{b}$	153	$G^{\#}/A^{b}$
53	D	162	G
57	$C^{\#}/D^{b}$	173	$F^{\#}/B^{b}$
60	C	182	F
64	B	193	E
68	$A^{\#}/B^{b}$	204	$D^{\#}/E^{b}$
72	A	217	D
76	$G^{\#}/A^{b}$	230	$C^{\#}/D^{b}$
81	G	243	C
85	$F^{\#}/G^{b}$		

Machine Language

Memory locations addressed by number, 0-65535. Each location has a numeric value of 0-255. Two art locations (such as SAVMSC, 88-89) are read with:

```
SAVMSC=PEEK(88)+256*PEEK(90)
```

PEEK(*82*) Returns decimal value stored at memory location *82*

POKE *82,226* Writes decimal *226* to memory location *82*

Useful PEEK and POKE Locations

Memory Configuration:

14,15	Highest memory used by BASIC program
88,89	Screen Memory Address (SAVMSC)
106	Top of RAM Address (Most Significant Byte) (RAMTOP)
741,742	Free Memory High Address (MEMTOP)
743,744	Free Memory Low Address

Display Screen:

77	Attract Mode (ATRACT)
82	Left Margin of Text Area (LMARGN)
83	Right Margin of Text Area (RMARGN)
84	Current Cursor Row (ROWCRS)
85,86	Current Cursor column (COLCRS)
87	Display Mode (DINDEX)
90	Starting Graphics Cursor Row (OLDROW)
91,92	Starting Graphics Cursor Column (OLDCOL)
93	Cursor Character Save/Restore (OLDCHR)
94,95	Cursor Memory Address (OLDADR)
96	Ending Graphics Cursor Row (NEWROW)
297,98	Ending Graphics Cursor Column (NEWCOL)
201	Display Screen Tab Interval (PTABW)
656	Text Cursor Row Position (TXTROW)
657,658	Text Cursor Column Position (TXTCOL)
675-680	Display Screen Tab Stop Map (TABMAP)
752	Cursor Inhibit (CRSINH)
755	Character and Cursor Control (CHACT)
756	Character Address Base (CHBAS)
765	Fill Data (FILDAT)
766	Display Control Characters (DSPFLG)
659	Split-Screen Text Windown Screen Mode (TINDEX)
660,661	Split-Screen Memory Address (TXTMSC)
665-667	Split-Screen Cursor Data
763	Last ATASCI Character or Plot Point (ATACHR)
54273	Character Control Register (CHACTL), same as CHACT

Display Lists:

512,513	Display List Interrupt Vector (VDSLST)
559	DMA Control Register (SDMCTL)
560,561	Display List Address (SDLST)
54286	Non-maskable Interrupt Enable (NMIEN)

Player-Missile Graphics:

623	Player/Playfield Priorities (GPRIOR)
704-707	Player-Missile Color Registers (COLPM0-COLPM3)
53248-53251	Player Horizontal Position Registers (HPOSP0-HPOSP3)
53256-53259	Player Width Registers (SIZEP0-SIZEP3)
53260	Missile Width Register (SIZEM)
53277	Graphics Control Register (GRACTL)
54279,54280	Player-Missile Base Register (PMBASE)

Cassette Buffer:

61	Cassette Buffer Pointer (BPTR)
63	Cassette End-of-File Flag (FEOF)
64	Beep Count (FREQ)
649	Cassette Read/Write Mode Flag (WMODE)
650	Cassette Buffer Size (BLIM)
1021-151	Cassette Buffer (CASEBUF)

Keyboard:

17	BREAK Key Flag (BRKKEY)
694	Inverse Video Keystrokes (INVFLG)
702	Shift/Control Lock Flag (SHFLOK)
764	Keyboard Character (CH)
767	Start/Stop Display Screen (SSFLAG)
53279	CONSOLE Switch Port (CONSOL)

Sound Control:

65	Input/Output Noise Control (SOUNDR)

Printer:

29	Printer Buffer Pointer (PBPNT)
30	Printer Buffer Size (PBUFSZ)
960-999	Printer Buffer (PRNBUF)

Free Memory Area:

1536-1663	Conditionally Available
1664-1791	Unconditionally Available

BASIC Program Control:

186,187	Stop Line Number (STOPLN)
195	Error Number (ERRSAV)
212,213	USR Function Value (FR0)
251	Radians or Degrees (RADFLG or DEGFLG)
564	Light Pen Horizontal Position (LPENH)
565	Light Pen Vertical Position (LPENV)

Other:

18-20	Real Time Clock (RTC)
832-960	IOC13's 0-7 (IOCB0-IOCB7)

Interrupt Control:

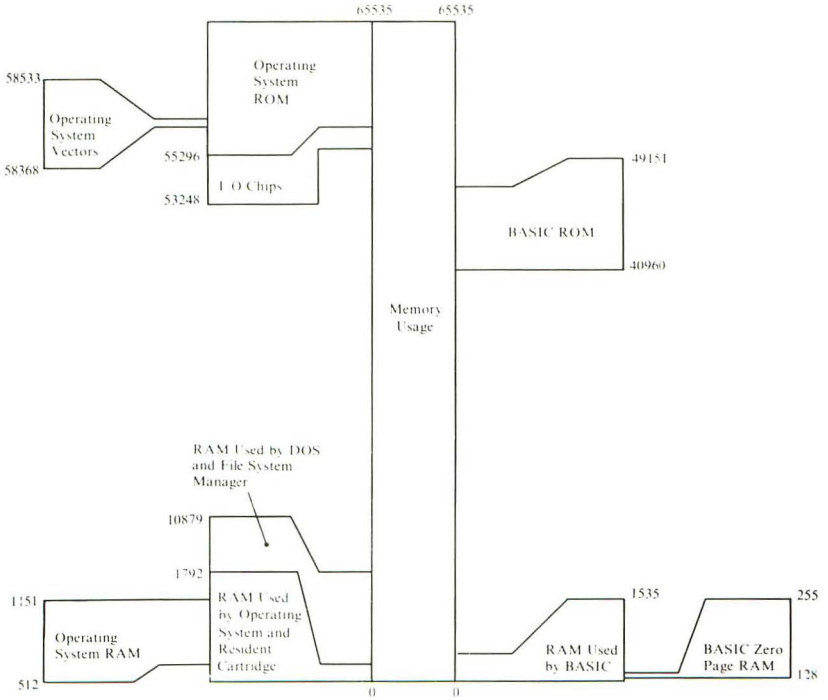
53744	IRQ Interrupt Status/Enable (IRQST/IRQEN)

Branching to Machine Language Subroutines from BASIC:

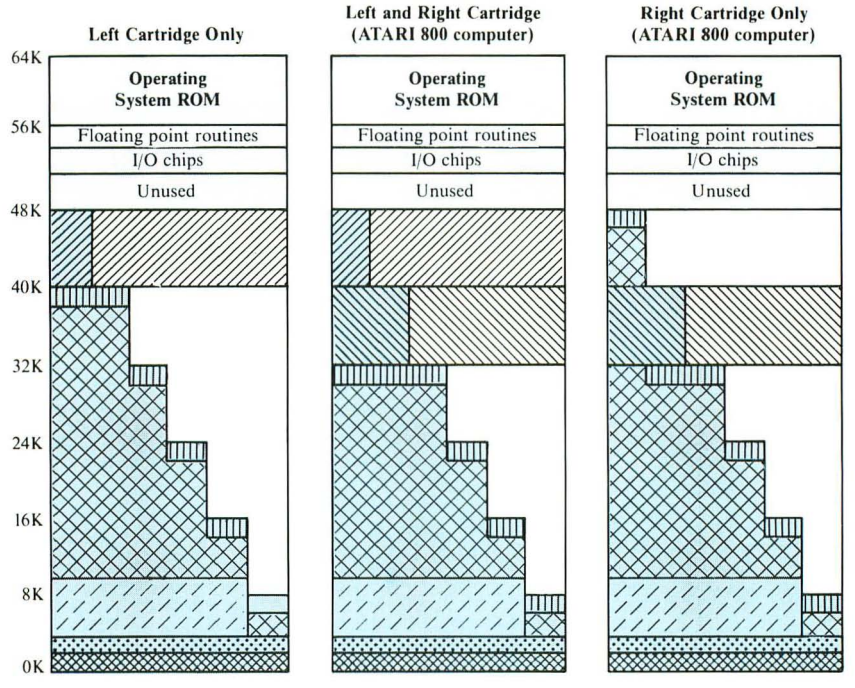
PRINT USR(*1536*)	Branches to machine program starting at *1536* (decimal)
PRINT USR(ADR(*SUBR$*))	Branches to machine language program stored in SUBR$; uses ADR to specify location of program
PRINT USR(*1536,3,,100*)	Branches, passes values of parameters *2-4*

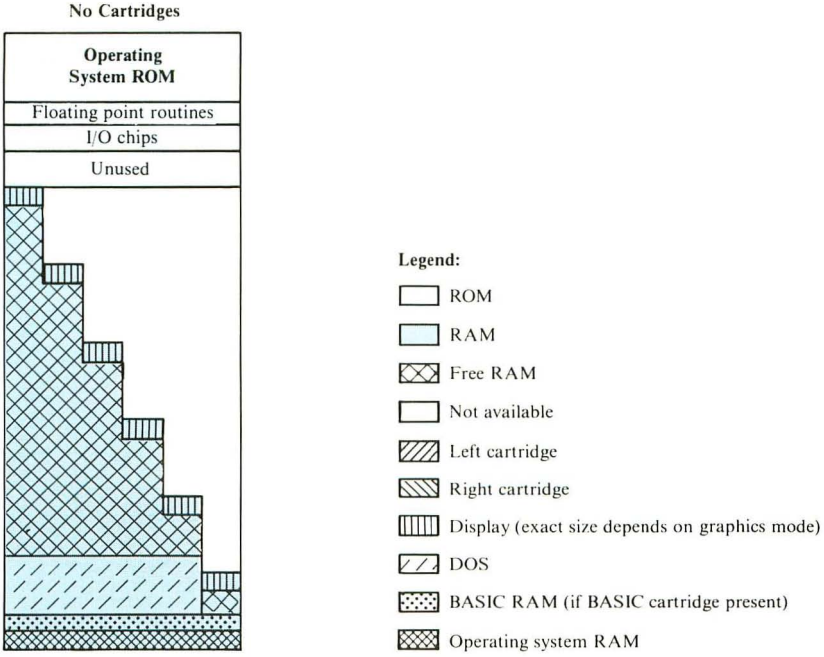
1. Parameters converted to 2-byte hexadecimal integers (0-65535)
2. Location of USR function pushed onto bottom of hardware stack
3. Parameters 2-4 PLACED on stack, last (100) first; low bytes before high
4. 1-byte count of 2-byte values (not including address) pushed on top
5. Returns to BASIC only if program at *1536* removes all entries

Memory Usage

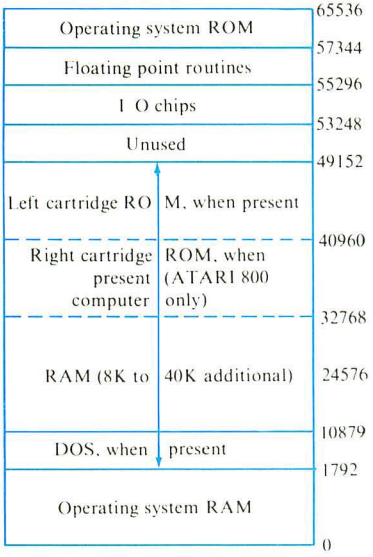


ATARI 400/800 Computer Memory Map

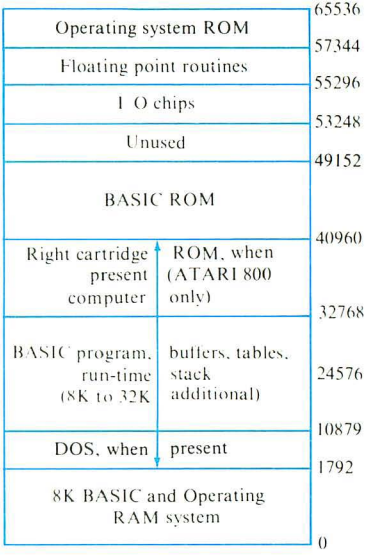

Left Cartridge Only
Left and Right Cartridge (ATARI 800 computer)
Right Cartridge Only (ATARI 800 computer)
64K
56K
48K
40K
32K
24K
16K
8K
0K
Operating System ROM
Floating point routines
I/O chips
Unused


No Cartridges
Operating System ROM
Floating point routines
I/O chips
Unused
Legend:
ROM
RAM
Free RAM
Not available
Left cartridge
Right cartridge
Display (exact size depends on graphics mode)
DOS
BASIC RAM (if BASIC cartridge present)
Operating system RAM

Memory Locations Without BASIC Resident



Memory Locations with Standard ATARI BASIC Resident



Operating System ROM (Memory Locations 55296-65535)	
Location	**Usage**
55296-57393	Floating point routines
57344-58367	Character set
58368-58533	Vectors
58534-59092	CIO
59093-59715	Interrupt handler
59716-60905	SIO
60906-61047	Disk handler
61048-61248	Printer handler
61249-61666	Cassette handler
61667-62435	Monitor
62436-65535	Display and keyboard handler

Operating System Vectors (Memory Locations 58368-58533)		
Location	**Type of Memory**	**Usage**
58368-58383	ROM	Editor
58384-58399	ROM	Screen
58400-58415	ROM	Keyboard
58416-58431	ROM	Printer
58432-58447	ROM	Cassette
58448-58495	ROM	Jump vectors
58496-58533	ROM	Initial RAM vectors

I/O Chips (Memory Locations 532-55295)		
Location	**Type of Memory**	**Usage**
53248-53503	I O	CTIA or GTIA
53504-53759	I O	Unused
54760-54015	I O	POKEY
54016-54271	I O	PIA
54272-54783	I O	ANTIC
54784-55295	I O	Unused

Operating System RAM (Memory Locations 512-1151)	
Location	**Usage**
512-553	Interrupt vectors
554-623	Miscellaneous
624-647	Game controllers
648-655	Miscellaneous
656-703	Screen RAM (depends on graphics mode)
704-711	Colors
712-735	Spare
736-767	Miscellaneous
768-779	DCB
780-793	Miscellaneous
794-831	Handler address tables
832-847	I/O Channel 0 (IOCB0)
848-863	I/O Channel 1 (IOCB1)
864-879	I/O Channel 2 (IOCB2)
880-895	I/O Channel 3 (IOCB3)
896-911	I/O Channel 4 (IOCB4)
912-927	I/O Channel 5 (IOCB5)
928-943	I/O Channel 6 (IOCB6)
944-959	I/O Channel 7 (IOCB7)
960-999	Printer buffer
1000-1020	Spare
1021-1151	Cassette buffer

RAM Used by Operating System, Resident Cartridge, or Free RAM (Memory Locations 0-2047)	
Location	**Usage**
0-127	Operating system zero page RAM
128-255	User zero page RAM
256-511	Stack
512-1151	Operating System RAM
1152-1791	User RAM
1792-2047	User Boot Area

BASIC* ROM (Memory Locations 40960-49151			
Location	**Usage**	**Location**	**Usage**
40960-41036	Cold start	43744-44094	Execute expression
41037-41055	Warm start	44095-44163	Operator precedence
41056-42081	Syntax	44164-45001	Execute operator
42082-42158	Search	45002-45320	Execute function
42159-42508	Statement name table	45321-47127	Execute statement
42509-43134	Syntax tables	47128-47381	CONT subroutines
43135-43358	Memory manager	47382-47542	Errors
43359-43519	Execute CONT	47543-47732	Graphics
43520-43631	Statement table	47733-48548	I/O routines
43632-43743	Operator table	48549-49151	Floating point

* Applies to standard ATARI BASIC only.

RAM Used by BASIC* (Memory Locations 0-255F)	
Usage	**Location**
0-127	Operating system zero page RAM
128-255	BASIC zero page RAM
256-511	Stack
512-1151	Operating system RAM
1152-1405	Syntax stack
1406-1535	Input line buffer
1536-1791	Free RAM
1792-End of free RAM	BASIC program: Syntax buffer or argument stack** Name table** Value table** Tokenized program** Array-strings area** Run-time stack**

* Applies to standard ATARI BASIC only.
** The actual memory locations depend on program and variable usage.

BASIC Zero Page RAM (Memory Locations 128-255)	
Location	**Usage**
128-145	Program pointers
146-202	Misc. BASIC RAM
203-209	Unused
210-255	Floating point work area

RAM Used by DOS Version 1.0 and File Management System (FMS)	
Location	**Usage**
1792-4863	File management system RAM
4864-9855	Disk operating system (DOS) RAM
9856-10879	Disk I/O buffers

RAM Used by DOS Version 2.0S and File Management System (FMS)	
Location	**Usage**
1792-4863	File management system RAM
4864-9855	Disk operating system (DOS) RAM
9856-10879	Drive 1-4 buffers and sector buffers 1-2
10880-LOMEM	Disk operating system (DOS) Utility programs (Sector buffers 3-7)

Codes, Characters, and Keystrokes

Codes, Characters, and Keystrokes							
Decimal Code	ATASCII Character	ASCII Character (If Any)	Keystrokes to Produce Character	Decimal Code	ATASCII Character	ASCII Character (If Any)	Keystrokes to Produce Character
0		NULL	CTRL-,	13		CR	CTRL-M
1		SOH	CTRL-A	14		SO	CTRL-N
2		STX	CTRL-B	15		SI	CTRL-O
3		ETX	CTRL-C	16		DLE	CTRL-P
4		EOT	CTRL-D	17		DC1	CTRL-Q
5		ENQ	CTRL-E	18		DC2	CTRL-R
6		ACK	CTRL-F	19		DC3	CTRL-S
7		BEL	CTRL-G	20		DC4	CTRL-T
8		BS	CTRL-H	21		NAK	CTRL-U
9		HT	CTRL-I	22		SYN	CTRL-V
10		LF	CTRL-J	23		ETB	CTRL-W
11		VT	CTRL-K	24		CAN	CTRL-X
12		FF	CTRL-L	25		EM	CTRL-Y

Codes, Characters, and Keystrokes (Continued)

Decimal Code	ATASCII Character	ASCII Character (If Any)	Keystrokes to Produce Character	Decimal Code	ATASCII Character	ASCII Character (If Any)	Keystrokes to Produce Character
26		SUB	CTRL-Z	40		(	SHIFT-9
27	 [1]	ESC	ESC\ESC	41		)	SHIFT-0
28	 [2]	FS	ESC\CTRL--	42		*	*
29	 [3]	GS	ESC\CTRL-=	43		+	+
30	 [4]	RS	ESC\CTRL-+	44		,	,
31	 [5]	US	ESC\CTRL-*	45		-	-
32		Space	SPACE BAR	46		.	.
33		!	SHIFT-1	47		/	/
34		"	SHIFT-2	48		0	0
35		#	SHIFT-3	49		1	1
36		$	SHIFT-4	50		2	2
37		%	SHIFT-5	51		3	3
38		&	SHIFT-6	52		4	4
39		'	SHIFT-7	53		5	5

Codes, Characters, and Keystrokes (Continued)							
Decimal Code	ATASCII Character	ASCII Character (If Any)	Keystrokes to Produce Character	Decimal Code	ATASCII Character	ASCII Character (If Any)	Keystrokes to Produce Character
54	6	6	6	68	D	D	D
55	7	7	7	69	E	E	E
56	8	8	8	70	F	F	F
57	9	9	9	71	G	G	G
58	:	:	SHIFT-;	72	H	H	H
59	;	;	;	73	I	I	I
60	<	<	<	74	J	J	J
61	=	=	=	75	K	K	K
62	>	>	>	76	L	L	L
63	?	?	SHIFT-	77	M	M	M
64	@	@	SHIFT-8	78	N	N	N
65	A	A	A	79	O	O	O
66	B	B	B	80	P	P	P
67	C	C	C	81	Q	Q	Q

Codes, Characters, and Keystrokes (Continued)							
Decimal Code	ATASCII Character	ASCII Character (If Any)	Keystrokes to Produce Character	Decimal Code	ATASCII Character	ASCII Character (If Any)	Keystrokes to Produce Character
82	R	R	R	96	♦	`	CTRL-,
83	S	S	S	97	a	a	(LOWR) A
84	T	T	T	98	b	b	(LOWR) B
85	U	U	U	99	c	c	(LOWR) C
86	V	V	V	100	d	d	(LOWR) D
87	W	W	W	101	e	e	(LOWR) E
88	X	X	X	102	f	f	(LOWR) F
89	Y	Y	Y	103	g	g	(LOWR) G
90	Z	Z	Z	104	h	h	(LOWR) H
91	[	[	SHIFT- ;	105	i	i	(LOWR) I
92	\	\	SHIFT- ,	106	j	j	(LOWR) J
93	]	]	SHIFT- +	107	k	k	(LOWR) K
94	^	↑	SHIFT- *	108	l	l	(LOWR) L
95	_	←	SHIFT--	109	m	m	(LOWR) M

Codes, Characters, and Keystrokes (Continued)

Decimal Code	ATASCII Character	ASCII Character (If Any)	Keystrokes to Produce Character	Decimal Code	ATASCII Character	ASCII Character (If Any)	Keystrokes to Produce Character
110		n	(LOWR) N	124		|	SHIFT-=
111		o	(LOWR) O	125	[6]	}	ESC\CTRL-< or ESC\SHIFT-<
112		p	(LOWR) P	126	[7]	~	ESC\BACK S
113		q	(LOWR) Q	127	[8]	DEL	ESC\TAB
114		r	(LOWR) R	128			(ATARI) CTRL-,
115		s	(LOWR) S	129			(ATARI) CTRL-A
116		t	(LOWR) T	130			(ATARI) CTRL-B
117		u	(LOWR) U	131			(ATARI) CTRL-C
118		v	(LOWR) V	132			(ATARI) CTRL-D
119		w	(LOWR) W	133			(ATARI) CTRL-E
120		x	(LOWR) X	134			(ATARI) CTRL-F
121		y	(LOWR) Y	135			(ATARI) CTRL-G
122		z	(LOWR) Z	136			(ATARI) CTRL-H
123		{	CTRL-;	137			(ATARI) CTRL-I

Codes, Characters, and Keystrokes (Continued)

Decimal Code	ATASCII Character	ASCII Character (If Any)	Keystrokes to Produce Character
138			(ATARI) CTRL-J
139			(ATARI) CTRL-K
140			(ATARI) CTRL-L
141			(ATARI) CTRL-M
142			(ATARI) CTRL-N
143			(ATARI) CTRL-O
144			(ATARI) CTRL-P
145			(ATARI) CTRL-Q
146			(ATARI) CTRL-R
147			(ATARI) CTRL-S
148			(ATARI) CTRL-T
149			(ATARI) CTRL-U
150			(ATARI) CTRL-V
151			(ATARI) CTRL-W
152			(ATARI) CTRL-X
153			(ATARI) CTRL-Y
154			(ATARI) CTRL-Z
155	EOL[9]		(ATARI) RETURN
156	[10]		ESC\SHIFT-BACK S
157	[11]		ESC\SHIFT->
158	[12]		ESC\CTRL-TAB
159	[13]		ESC\SHIFT-TAB
160			(ATARI) SPACE BAR
161			(ATARI) SHIFT-1
162			(ATARI) SHIFT-2
163			(ATARI) SHIFT-3
164			(ATARI) SHIFT-4
165			(ATARI) SHIFT-5

Codes, Characters, and Keystrokes (Continued)

Decimal Code	ATASCII Character	ASCII Character (If Any)	Keystrokes to Produce Character	Decimal Code	ATASCII Character	ASCII Character (If Any)	Keystrokes to Produce Character
166	&		(ʌ) SHIFT-6	180	4		(ʌ) 4
167	'		(ʌ) SHIFT-7	181	5		(ʌ) 5
168	(		(ʌ) SHIFT-9	182	6		(ʌ) 6
169	)		(ʌ) SHIFT-0	183	7		(ʌ) 7
170	*		(ʌ) *	184	8		(ʌ) 8
171	+		(ʌ) +	185	9		(ʌ) 9
172	,		(ʌ) ,	186	:		(ʌ) SHIFT-;
173	-		(ʌ) -	187	;		(ʌ) ;
174	.		(ʌ) .	188	<		(ʌ) <
175	/		(ʌ) /	189	=		(ʌ) =
176	0		(ʌ) 0	190	>		(ʌ) >
177	1		(ʌ) 1	191	?		(ʌ) SHIFT-/
178	2		(ʌ) 2	192	@		(ʌ) SHIFT-8
179	3		(ʌ) 3	193	A		(ʌ) A

Codes, Characters, and Keystrokes (Continued)

Decimal Code	ATASCII Character	ASCII Character (If Any)	Keystrokes to Produce Character	Decimal Code	ATASCII Character	ASCII Character (If Any)	Keystrokes to Produce Character
194			(^) B	208			(^) P
195			(^) C	209			(^) Q
196			(^) D	210			(^) R
197			(^) E	211			(^) S
198			(^) F	212			(^) T
199			(^) G	213			(^) U
200			(^) H	214			(^) V
201			(^) I	215			(^) W
202			(^) J	216			(^) X
203			(^) K	217			(^) Y
204			(^) L	218			(^) Z
205			(^) M	219			(^) SHIFT-,
206			(^) N	220			(^) SHIFT-+
207			(^) O	221			(^) SHIFT-.

Codes, Characters, and Keystrokes (Continued)

Decimal Code	ATASCII Character	ASCII Character (If Any)	Keystrokes to Produce Character	Decimal Code	ATASCII Character	ASCII Character (If Any)	Keystrokes to Produce Character
222	^		(ʌ) SHIFT-*	236	l		(ʌ) (LOWR) L
223	_		(ʌ) SHIFT--	237	m		(ʌ) (LOWR) M
224	◆		(ʌ) CTRL-.	238	n		(ʌ) (LOWR) N
225	a		(ʌ) (LOWR) A	239	o		(ʌ) (LOWR) O
226	b		(ʌ) (LOWR) B	240	p		(ʌ) (LOWR) P
227	c		(ʌ) (LOWR) C	241	q		(ʌ) (LOWR) Q
228	d		(ʌ) (LOWR) D	242	r		(ʌ) (LOWR) R
229	e		(ʌ) (LOWR) E	243	s		(ʌ) (LOWR) S
230	f		(ʌ) (LOWR) F	244	t		(ʌ) (LOWR) T
231	g		(ʌ) (LOWR) G	245	u		(ʌ) (LOWR) U
232	h		(ʌ) (LOWR) H	246	v		(ʌ) (LOWR) V
233	i		(ʌ) (LOWR) I	247	w		(ʌ) (LOWR) W
234	j		(ʌ) (LOWR) J	248	x		(ʌ) (LOWR) X
235	k		(ʌ) (LOWR) K	249	y		(ʌ) (LOWR) Y

Codes, Characters, and Keystrokes (Continued)

Decimal Code	ATASCII Character	ASCII Character (If Any)	Keystrokes to Produce Character	Decimal Code	ATASCII Character	ASCII Character (If Any)	Keystrokes to Produce Character
250	 [14]		ESC\CTRL-2	253			(Λ) (LOWR) Z
251	 [15]		(Λ) ESC\CTRL-BACK S	254			(Λ) CTRL-;
252	 [16]		(Λ) ESC\CTRL->	255			(Λ) SHIFT-=

Notes:

[1]The character represents a control character. In most cases, this control character does nothing; CHR$(27) is generally a nondisplaying character. However, if the next character displayed is a control character (ATASCII codes 27, 28, 29, 30, 31, 125, 126, 127, 156, 157, 158, 159, 253, 254, or 255), the control process does not take place. Instead, the representative character itself appears.

[2]The character  represents the control character which moves the cursor up one row. If the character displayed just before this was ATASCII code 27, the character displays; the cursor does not move.

[3]The character  represents the control character which moves the cursor down one row. If the character displayed just before this was ATASCII code 27, the character displays; the cursor does not move.

[4]The character represents the control character which moves the cursor one column left. If the character displayed just before this was ATASCII code 27, the character displays; the cursor does not move.

[5]The character  represents the control character which moves the cursor one column right. If the character displayed just before this was ATASCII code 27, the character displays; the cursor does not move.

[6]The character represents the control character which clears the screen and moves the cursor to the home position. If the character displayed just before this was ATASCII code 27, the character displays; the screen is not cleared.

Notes (continued):

[7]The character ◄ represents the control character which moves the cursor one column left and replaces the character there with a blank space. If the character displayed just before this was ATASCII code 27, the character ◄ displays; the cursor does not move.

[8]The character ► represents the control character which advances the cursor to the next tab stop. If the character displayed just before this was ATASCII code 27, the character ► displays; the cursor does not move.

[9]The ATASCII end-of-line character.

[10]The character ▲ represents the control character which deletes the line on which the cursor is located. If the character displayed just before this was ATASCII code 27, the character ▲ displays; the deletion does not occur.

[11]The character ▼ represents the control character which inserts a line above the one on which the cursor is located. If the character displayed just before this was ATASCII code 27, the character ▼ displays; the insertion does not occur.

[12]The character ◄ represents the control character which clears the tab stop (if any) at the current cursor position. If the character displayed just before this was ATASCII code 27, the character ◄ displays; no tab stop is affected.

[13]The character ► represents the control character which sets a tab stop at the current cursor position. If the character displayed just before this was ATASCII code 27, the character ► displays; no tab stop is set.

[14]The character ◣ represents the control character which beeps the built-in speaker; nothing is displayed. If the character displayed just before this was ATASCII code 27, the character ◣ displays; the speaker remains silent.

[15]The character ◄ represents the control character which deletes the character to the right of the cursor, shifting the remainder of the logical line one space to the left. If the character displayed just before this was ATASCII code 27, the character ◄ displays; no deletion occurs.

[16]The character ► represents the control character which inserts a blank space to the right of the cursor, shifting the remainder of the logical line one space to the right. If the character displayed just before this was ATASCII code 27, the character ► displays; no insertion occurs.

ATARI BASIC Keywords and Abbreviations

Keyword	Abbrev.	Keyword	Abbrev.	Keyword	Abbrev.
ABS		GOTO	G.	PUT	PU.
ADR		GRAPHICS	GR.	RAD	
AND		IF		READ	REA.
ASC		INPUT	I.	REM	R. or
ATN		INT		RESTORE	RES.
BYE	B.	LEN		RETURN	RET.
CLOAD	CLOA.	LET	LE.	RND	
CHR$		LIST	L.	RUN	RU.
CLOG		LOAD	LO.	SAVE	S.
CLOSE	CL.	LOCATE	LOC.	SETCOLOR	SE.
CLR		LOG		SGN	
COLOR	C.	LPRINT	LP.	SIN	
COM		NEW		SOUND	SO.
CONT	CON.	NEXT	N.	SQR	
COS		NOT		STATUS	ST.
CSAVE	CS.	NOTE	NO.	STEP	
DATA	D.	ON		STICK	
DEG	DE.	OPEN	O.	STRIG	
DIM	DI.	OR		STOP	STO.
DOS	DO.	PADDLE		STR$	
DRAWTO	DR.	PEEK		THEN	
END		PLOT	PL.	TO	
ENTER	E.	POINT	P.	TRAP	T.
EXP		POKE	POK.	USR	
FOR	F.	POP		VAL	
FRE		POSITION	POS.	XIO	X.
GET	GE.	PRINT	PR. or ?		
GOSUB	GOS.	PTRIG			

Index

ATARI® 400/800™ DiskGuide™

ATARI® 400/800™ users will find quick-reference tips on DOS, ATARI® BASIC, numeric functions, machine functions, and more in this DiskGuide.™ Tables, charts, and diagrams are also included in this easy-to-use, computer-side reference.

Look for other Osborne/McGraw-Hill titles including:

- **Your ATARI® Computer: A Guide to the ATARI® 400/800™ Personal Computers**

ISBN 0-931988-95-0

McGraw Hill