



D:CHECK in Action!

by Steven Yates

Because of the nature of the Action! system, typing checkers like D:CHECK (issue 16) cannot be implemented. A lack of line numbers leaves no way to communicate to the user where typos are. Moreover, the flexibility in the source program's form makes finding errors difficult—without requiring the reader to type the program exactly as it appears.

D:CHECK in Action! gives you a program (D:CHECK.ACT) which works with the Action! system to provide interactive checking and correcting of typing errors. Instead of printing a list of numbers which you compare to a similar list in the magazine, this Action! version finds the errors and puts you back into the editor at their approximate locations in your source.

This extra power takes some more typing on your part. If you look at Listing 1, you'll notice the first few lines contain a set of numbers headed "CHECKSUM DATA." All other Action! programs printed in **ANALOG Computing** from now on will have similar lines at the start of their listings.

These lines give D:CHECK in Action! its power. The program generates numbers from what you've typed and compares them to the numbers here. If there are any discrepancies, it will then help you to find the problems, so they can be corrected.

When typing in the listing, these lines must be typed in exactly as printed.

There are square brackets at the beginning and end of the checksum list, and one space between each value. All values are two-digit numbers and must be typed as such.

After these lines are typed, the remainder of the program can be entered in any way you wish.

You may type it as listed, or you may decide not to include the blank lines inserted for easier reading. You may add more blank lines, if you like. You may combine short lines to form one line with a space or more between the originals; or you could break long lines into two or more lines, so everything fits on the screen.

The program ignores spaces, so you don't need to indent lines, and you can add spaces if it's more readable for you. D:CHECK in Action! also ignores comments, so you can leave them out and save some typing, or add some of your own.

This flexibility of program form is basic to the Action! system which D:CHECK in Action! preserves. Remember, any modifications to the form of the program may make it more difficult to compare to the original if there's a problem—so use your judgement. Anything between quotes must be typed exactly as shown; including spaces, and upper and lower case letters.

The latter present another problem. Action! allows you to type in whatever case you want, but also offers the option of being case sensitive. This lets you have two or more variables of the same name, with different letters in upper case to distinguish them.

D:CHECK in Action! offers the same option. If the article doesn't specify otherwise, you may use either case. If the program must be compiled with the case-sensitive option, the words *Case Sensitive* will appear with the checksum data. You must type all letters in the case in which they're listed. Because it must be able to handle other case-sensitive programs, D:CHECK in Action! must also be able

D:CHECK in Action! *continued*

to compile under this option, so remember to type each letter properly.

Using it.

First, type in Listing 1 and save it. Because the program can't check itself until it compiles successfully, you must be careful keying it in. Don't forget that the program is case sensitive.

Try compiling. If there are any compile errors, fix them as you normally would. Once **D:CHECK in Action!** compiles, run it. If the program says there are no problems, you're ready to use it for other programs printed in **ANALOG Computing**.

If it tells you there are problems, follow the directions below. The program cannot guarantee the locations it gives for errors in its own source file, because typing errors may be causing problems with its error-locating routines. Be sure to save a final copy after you've fixed all the errors, as **D:CHECK.ACT** (cassette users may save it as **C:**).

To check another program containing the checksum numbers, type it in and save a copy. With the newly typed program still in memory, go to the monitor. If the article says the program is case sensitive, use the monitor option command to set it for this. Otherwise set this option to "no."



Tired of always searching for the right recipe?
Are the pages of your recipe books covered with your recipe ingredients?
Fed up at guessing amounts when a recipe serves five but you want it for two?
If so then you need **The Computer Gourmet**.

With **The Computer Gourmet** you can:

- Easily save your favorite recipes (even give them a rating!)
- Find any recipe you need within seconds
- Adjust for a different serving size *automatically*
- Print the whole recipe or just the list of ingredients

Best of all, **The Computer Gourmet** comes with a disk full of recipes!
(With everything from main courses to desserts).

Available on disk for Atari[®] computers (requires 48K). To order, send \$29.95 plus \$2.00 for postage (Texas residents please add 5 1/8% sales tax) to:

New Horizons Software • P.O. Box 43167 • Austin, Texas 78745
Or call (512) 280-0319

New Horizons Expanding Your Life

MasterCard and Visa accepted.

Please write to us for information on all of our products for Atari computers.
Dealer inquiries invited. Atari is a trademark of Atari, Corp.

CIRCLE #111 ON READER SERVICE CARD

When this is done, type **R"D:CHECK.ACT** (cassette users, type **R"C:**). The program will load from disk, compile, then run. It must be uncompiled when stored, because it needs to relocate itself for each program checked.

If the program being checked isn't too large (about 100 disk sectors), time can be saved by loading **D:CHECK.ACT** into the second editing window. It can then be compiled and run without accessing the disk each time.

To do this, type in the program to be checked and save it. Enter window 2 by pressing **CTRL-SHIFT-2**, read **D:CHECK.ACT**, then enter the monitor and use **C** and **R** to compile and run.

To rerun **D:CHECK.ACT** after a problem's corrected, just enter window 2 (**CTRL-SHIFT-2**) before returning to the monitor, then compile and run as before. If you don't move the cursor to window 2 before entering the monitor, **Action!** will attempt to compile the program in window 1, and **D:CHECK.ACT** won't be executed.

If you get an out-of-memory error when loading or compiling **D:CHECK.ACT**, make sure your source program is saved and use the monitor boot command to reset **Action!** Then reload your program and run **D:CHECK.ACT** from disk or tape as explained previously.


If the **D:CHECK.ACT** program says there are no problems, you should be able to compile the program. If it does find a problem, it will display the checksum lines on-screen, with one number highlighted. It will ask if that sum was typed correctly.

Check the highlighted value against the magazine listing. If they don't match, press **N** in response to its question. Return to the editor. The cursor will be on the first digit of the incorrect sum. Retype the sum, return to the monitor, and repeat the command to run **D:CHECK.ACT**.

If the highlighted sum matches that printed in the magazine, press **Y**. The program will tell you to return to the editor and check the line containing the cursor, plus a certain number of lines following it. The number of lines to check depends mostly on line length, and blank lines aren't counted.

Find any mistakes you can on these lines, correct them, return to the monitor, and rerun **D:CHECK.ACT**. Once you've found all errors and are given a clean check, save a final copy, then compile and run it, according to the directions in the related article.

It's possible that a program which checks out all right won't compile. If this happens, return to the editor and make sure you didn't insert or delete a space, which would confuse the compiler. Check the word the cursor's on and change it to look exactly as it does in the magazine. This should be the only thing to cause a problem in a properly checked program.

Now you should be able to enjoy printed **Action!** programs without being frustrated by cryptic compiler errors, or having a program compile and not perform as expected. If you remember the difference **D:CHECK** made in the time it took to get a **BASIC** program running, you'll type in **D:CHECK in Action!** as soon as possible. 

(Listing starts on next page)

Steven Yates has owned his Atari 400 (with 48K and an external keyboard) for three and one-half years. He has an associate's degree in data processing and is currently pursuing a bachelor's in electrical engineering.

Listing 1.
Action! listing.

```

; CHECK.ACT
; Steven Yates
; 12/02/85

; CHECKSUM DATA
; C9E 43 B1 2D 74 DD 67 7C
; 13 30 E5 8F 7A CA C9 77
; AD AE 96 44 B0 F8 99 39
; EB J

BYTE StartChar, CurChar, Count, X,
Flag=[0], Character, Sum, ISum,
Product, Key, Case=[0], Column=1152,
Sensitive=[1], Lines, SumLine,
WrongLine, Segment=[0], String=[1],
SubString=[2], Space=[32]
CARD StartLine, Line=1160,
FirstLine=1156
BYTE ARRAY Sums(256)
CARD ARRAY SumLines(32)
BYTE POINTER Length
CARD POINTER CurLine, NextLine,
WrongSum

DEFINE is="=", not="(>)", Done="Flag=1"

PROC End_Of_Line()
DO
CurLine=NextLine^
CurChar=7
Length=CurLine+6
NextLine=CurLine+4
IF CurLine=0 THEN
EXIT
FI
UNTIL Length^>0
OD

RETURN

PROC Quotes()
IF Segment is String THEN
Segment=0
ELSE
Segment=String
FI

RETURN

PROC Ignore()
X=-1
Count=-1
Character=0

RETURN

PROC Check_Line()
Character=Peek(CurLine+CurChar)
IF Segment is SubString THEN
Segment=0
ELSEIF Character="" THEN
Quotes()
ELSEIF Segment is String THEN

```

```

ELSEIF Character="" THEN
Segment is SubString
ELSEIF Character is Space THEN
Ignore()
ELSEIF Character=''; THEN
Ignore()
CurChar=Length^+6
ELSEIF Case not Sensitive AND
Character>96 AND
Character<123 THEN
Character=-32
FI
Product=(X*Character) RSH Case
Sum+=Product
CurChar+=1
X+=1
IF X=4 THEN
X=1
FI

RETURN

PROC Find_Sums()
DO
DO
Character=Peek(CurLine+CurChar)
IF Character not ' THEN
End_Of_Line()
ELSEIF Length^>1 THEN
EXIT

```

ATARI Pierstorff
800.800XL.65XE.
130XE

MORE! GRAPHICS!
for The Print Shop™

Over 120 Graphics!
10 Screen Magic Scenes!

THE PIERSTORFF CO. \$19⁹⁵ CLASSIC SUPPLY
131 W. MAIN ST. WOODLAND, CA 95695 (916) 666-3530 ORDERS

California residents add 8% tax add \$3. shipping
CHECK C.O.D. VISA MASTERCARD
THE PRINTSHOP IS A TRADEMARK OF BRODERBUND SOFTWARE DEALER INQUIRIES WELCOME
NOT AFFILIATED WITH THE PIERSTORFF CO.

CIRCLE #112 ON READER SERVICE CARD

```

PROC D_Check()
  Initialize()
  Get_Sums()
  ISUM=0
  DO
    IF Done THEN
      EXIT
    FI
    SUM=0
    Lines=0
    FOR Count=0 TO 127 DO
      IF Count=0 THEN
        StartLine=CurLine
        StartChar=CurChar-6
      FI
      Check_Line()
      IF CurChar=Length^+7 THEN
        End_Of_Line()
        IF CurLine=0 THEN
          Check_SUM()
          IF Done THEN
            EXIT
          FI
          No_Problems()
          EXIT
        FI
        Lines==+1
      FI
    OD
    IF Done THEN
      EXIT
    FI
    Check_SUM()
  OD
  Close(1)
RETURN

```

WANT TO SUBSCRIBE?

It's worth it.

**CALL TOLL FREE
1-800-345-8112
In Pennsylvania
1-800-662-2444**

MICROTYPE

A DIVISION OF MICRO PERIPHERALS, INC.



P.O. BOX 388
KETTERING, OHIO 45409



ATARI		SOFTWARE and BOOKS							
520 ST's C'mon Now, Do It! CALL	ST SOFTWARE TOO MUCH TO LIST CALL	ALL titles from: Haba, VIP, Broderbund,							
SF 314 Double Sided Drive CALL	Mark of the Unicorn, Hippo, Unison World,	Migraph, Oss, Infocom, Atari, Michtron,							
SHD 204 20 Megabyte Hard Disk CALL	SST Systems, Mirage Concepts, etc.	We will have everything WORTH having!							
SC 1224 RGB Color Monitor CALL	"THE C PROGRAMMING LANGUAGE" by B.W.	Kernighan and D.M. Ritchie 19							
130 XE (8-bit Wonder of the World!) 129	8 BIT SOFTWARE FOR THE LATEST. CALL	PAPERCLIP 39							
65 XE 89	PRINTSHOP 29	GRAPHICS LIBRARY #1, #2, or #3 (each) 16							
1050 Disk Drive 135	O.S.S. BASIC XE 46	O.S.S. BASIC XL 36							
1020 Color Printer/Plotter 25									
Power Supply 400/800/810 1050/850 15									
Power Supply 600/800 XL, 130 XE 26									
INDUS GT 219									
Power Supply for Indus GT 19									
PANASONIC		MONITORS							
KX-P1080 5 NLO MODES! NEW 219	TEKNIKA MJ-10 Composite Color 189	TEKNIKA MJ-22 RGB and Composite 279							
KX-P1091 Rated the No. 1 Printer! 249	THOMPSON Green W/Audio 85	THOMPSON Amber W/Audio 90							
KX-P1092 80 col, True 180 cps 339	THOMPSON Composite Color 159	THOMPSON RGB/Composite 319							
KX-P1592 136 col, True 180 cps 549									
KX-P3131 L.Q. Daisy, 80 col 279									
KX-P3151 L.Q. Daisy, 136 col 429									
KX-P110 Ribbon, Blk 9									
COLOR RIBBONS 11									
CITIZEN		ACCESSORIES							
MSP-10 289	ST- COVERS, Heavy Grade Vinyl 8	ST- MOUSE MAT, Matching ST Color 10							
MSP-15 CALL	ST- 6" Printer Cable 19	ST- Modem Cable (to Hayes, etc.) 17							
EPSON		ST- Monitor Stand, Swivel & Tilt 15							
LX-80 (80 col) 239	Disk File for 3.5" disks (holds 40) 9	Flop N File DATA CASE (holds 50) 8							
FX-85 (80 col) 379	Disk File, with Lock (holds 100!) 13	Rotary Disk File (holds 72) 15							
FX-286 200 cps (135 col) 539	Power Strip, 6 outlet, (15 amp Surge) 15	Printer Stand, Heavy Duty, Sloping 13							
STAR MICRONICS		ATARI "Standard" Joystick 6							
NX-10 (80 col) NEW MODEL CALL	6" Atari Serial I/O Cable 7	Compuserve Starter Kit 21							
SG-10 (80 col) 229	U.S. DOUBLER (Dbl. Density for 1050) 49	"Duplicator" 129							
SG-15 (135 col) 429									
STAR SG-10 Ribbons 4									
MODEMS		PRINTER SUPPLIES							
ATARI 1030 45	MAILING LABELS, White, 500 pack 3	per 1000 4							
XM-301 Direct Connect 38	Blu, Pnk, Gn, Yel, 800 pack (200 ea) 9	per 500, any 1 color 5							
QMI 1200 ST (for 520 ST Complete!) 179	per 1000, any 1 color 7	Big Labels, 1-7/16x4", White, per 500 5							
HAYES 1200 Smartmodem 399	PRINTER PAPER, Micro-Fine perfs, 20 lb.	500 sheets, Pure White Bond 8							
US ROBOTICS COURIER 2400-100% Hayes! 429	1000 sheets, same as above 14	Carton (2600 sheets), as above 29							
PRENTIS P212ST-1200 bps, 100% Hayes! 239	PRINTSHOP "Rainbow" Color Paper Packs	Pastels (5 colors), 50 sheets of ea 12							
SUPRA 1200 AT 179	Matching Envelopes, 20 of each 6	Brights (8 colors), 50 sheets of ea 29							
SUPRA ST MODEM, 1200 bps 179	Matching Envelopes, 20 of each 10	ALL 13 colors, 50 sheets of each 39							
VOLKMODEM 1200 189	Matching Envelopes, 20 of each 14	(Deduct 10% for 100/color paper packs)							
AVATEX Smart 1200 bps Special 119									
INTERFACES/BUFFERS		Prices Are Per Box of 10 DISKETTES Minimum Order of 2 Boxes							
ATARI 850 In Stock! 119	No. of	GENERIC	BONUS	WABASH	3.5" MICRO-FLOPPIES				
P:R: CONNECTION (100% 850 compatible) 66	Boxes	SS/DD	DS/DD	SS/DD	SONY	MAXELL	VERBATIM		
CABLES - We've Got 'Em CALL	2-5	8.50	10.50	10.50	13.50	10.50	23.50	21.50	18.50
U CALL (For Hayes, etc.) 39	6-10	7.50	9.50	9.50	12.50	9.50	21.50	19.50	17.50
U PRINT A 59									
U PRINT A-64 with 64K Buffer 99									
APE FACE XLP 59									
SUPRA/MPP MICROPRINT 39									
SUPRA/MPP MICROSTUFFER (64K Buffer) 79									
SUPRA/MPP 1150 54									

Prices Are Per Box of 10 DISKETTES Minimum Order of 2 Boxes

No. of Boxes	GENERIC		BONUS		WABASH	3.5" MICRO-FLOPPIES		
	SS/DD	DS/DD	SS/DD	DS/DD		SONY	MAXELL	VERBATIM
2-5	8.50	10.50	10.50	13.50	10.50	23.50	21.50	18.50
6-10	7.50	9.50	9.50	12.50	9.50	21.50	19.50	17.50

Rainbow Colored Centech Disks (2 ea of 10 colors per pkg) 17
"Silver" Centech Disks (20 Pack) 17

**TO ORDER, CALL TOLL FREE
1-800-255-5835**

M-F 9 am-9 pm • SAT 10 am-4 pm EST

Ohio Residents, Order Status or
Tech. Info Call (513) 294-6236



TERMS AND CONDITIONS
24 HR shipping on in stock items • NO EXTRA CHARGES FOR CREDIT CARDS! • Minimum order \$20 • C.O.D. to continental U.S. only, add \$3 • Ohio residents add 6% sales tax • Please allow 3 weeks for personal or company checks to clear • Shipping/Handling: Hardware, \$4 minimum; Software and most accessories, \$3 minimum • Over-night shipment available at extra charge • We ship to Alaska, Hawaii, Puerto Rico (UPS Blue Label Only), APO, and FPO • Canadian orders, actual shipping plus 5%, minimum \$5 • All defective products require a return authorization number to be accepted for repair or replacement • No free trials or credit • Due to changing market conditions, call toll free for latest price and availability of product.

CIRCLE #113 ON READER SERVICE CARD

```

ELSE
  End_Of_Line()
FI
IF CurLine=0 THEN
  Done
  EXIT
FI
OD
CurChar=8
Character=Peek(CurLine+CurChar)
IF Character is '[' THEN
  EXIT
FI
IF Done THEN
  PrintE("Listing does not")
  PrintE("contain checksums.")
  Pute()
  PrintE("Cannot CHECK!")
  EXIT
FI
End_Of_Line()
OD
RETURN
PROC Get_Sums()
BYTE I
BYTE ARRAY Hex(1)
Find_Sums()
SumLine=0
DO
  IF Done THEN
    EXIT
  FI
  SumLines(SumLine)=CurLine
  CurChar=9
  FOR ISUM=0 TO 7 DO
    FOR I=0 TO 1 DO
      Hex(I)=Peek(CurLine+CurChar+I)
      IF Hex(I)='A' THEN
        Hex(I)=-('A'-'9'-1)
      FI
      Hex(I)=-'0
    OD
    SUM=(Hex(0) LSH 4)+Hex(1)
    Sums(SumLine*8+ISUM)=SUM
    IF Peek(CurLine+CurChar+3) is '['
      THEN
        Done
        EXIT
      FI
      CurChar==+3
    OD
    End_Of_Line()
    IF Done THEN
      Flag=0
      EXIT
    FI
    SumLine==+1
  OD
RETURN
PROC Mistyped()
Line=StartLine
Column=StartChar
PrintE("Return to editor and check")
Print("the ")
PrintB(Lines)
PrintE(" lines following the line")
Print("the cursor is on for a ")
PrintE("typo.")
RETURN

```

```

PROC Bad_Sum()
BYTE I
IF Case is Sensitive THEN
  Print("If article does not ")
  PrintE("specify")
  Print("case sensitive, use ")
  PrintE("option")
  Print("command to set this to ")
  PrintE("no.")
  Pute()
  Pute()
FI
WrongLine=ISUM/8
SUM=ISUM&7
WrongSUM=SumLines(WrongLine)+9+SUM*3
WrongSUM^==%$8080
FOR I=0 TO SumLine DO
  PrintE(SumLines(I)+6)
OD
WrongSUM^==!$8080
Pute()
Print("Is highlighted sum correct?")
Key=GetD(1)
Key==%32
Pute()
Pute()
IF Key='y' THEN
  Mistyped()
ELSE
  Print("Return to editor and ")
  PrintE("correct")
  PrintE("mistyped sum.")
  Line=SumLines(WrongLine)
  Column=(SUM+1)*3
FI
RETURN
PROC No_Problems()
PrintE("Program CHECKs out fine.")
PrintE("Save program and use")
PrintE("according to directions")
PrintE("in the article.")
Done
RETURN
PROC Check_Sum()
IF SUM<>Sums(ISUM) THEN
  Bad_Sum()
  Done
FI
ISUM==+1
RETURN
PROC Initialize()
IF Peek(1226)=255 THEN
  Case is Sensitive
  FI
  X=1
  CurLine=FirstLine
  Length=CurLine+6
  NextLine=CurLine+4
  IF Length^=0 THEN
    End_Of_Line()
  FI
  CurChar=7
  Close(1)
  Open(1,"K:",4,0)
RETURN

```