BY KEVIN SHERRATT

ACTION! Toolbox

Lightning-fast command finder

Two powerful and widely useful routines for the ACTION! programming language. These programs work on all 8-bit Atari computers of any memory size, with disk or cassette. The ACTION! cartridge from ICD/OSS is required.

Whether you're using ACTION! to build "The Wizards of Zondar" or "The Ultimate Chef's Companion," your programming toolbox will be incomplete without a procedure that removes individual words from a string you've entered—and a procedure that compares those words with a list of known words in hopes of a match.

For efficiency and versatility, the following two procedures fill the gap nicely and can easily be customized by experienced ACTION! programmers.

1: WORDFIND

This procedure strips each Word, one at a time, from String—which is a global **BYTE ARRAY** similar to a BASIC string variable. In the process it discards the spaces between Words, no matter how many times you pressed the [SPACEBAR].

In its first loop, **Wordfind( )** searches String for a non-space character, incrementing the Index into the array as it goes. Upon finding one, it stores the Index value in Start. The next loop searches for a space—and the end of the Word—while continuing to increment Index.

When another space or the end of the array is found, the procedure writes the characters between Start and Index into the global Word. Since Index, too, is a global variable, calling **Wordfind( )** again will result in the next consecutive Word. Therefore, Index must be set to 1 before each new string is examined.

2: MATCHUP

In most applications, after you isolate a single Word you'll want to check it against the commands with which your program is prepared to deal. **Matchup( )** can help you here.

This procedure requires that each global List of commands contains only elements of the same length. For example:

```
Comlist1 = 'EAST WEST
   NORTH SOUTH".
Comlist2 = 'EAWENOSO"
```

In Comlist1 the Increment is five—meaning that a new command begins every five characters. Comlist2 has shortened those same commands to two characters. In either case, **Matchup( )** must be called using three parameters: the potential Command to be compared, the List of known commands and the Increment of the list.

Matchup( ) then jumps through the list by Increments, searching the first character of each command for a match. Upon finding one, it compares the remaining characters. If all the characters match, it alters the global variable Match to show where in the list the command was found. For example, after calling **Matchup(Word,Comlist1,5)** you find that Match=6. You then know that "Word" matched the command beginning at character 6—in this case, WEST.

Matchup( ) will not search past either the given Increment or a space. Thus, if you call it to examine the word WESTERLY against Comlist1, Match would still equal 6. If no match is found, Match will equal 0. As a global, Match can be used in any number of procedures, but it is always reset by the next call to **Matchup( )**.

CALLER EXAMPLE

Carefully type in Listing One, TOOLBOX.ACT, and store a copy to disk before you compile and run it.

The sample **Caller( )** procedure shows you how to use **Wordfind( )** and **Matchup( )**. In this example, Comlist, the command list, is "DOG CAT COW MULE". When run, the program asks you to type one of the four Words in the command list. Then the program finds the Word in the command list and prints the word and its position in the string. ▲

Kevin Sherratt is a full-time science fiction writer and part-time programmer from London, Ontario. He is currently working on an 800XL text adventure game. This is his first appearance in Antic.

Listing on page 77

```
   L(RESPONSE$(1,PTR-1)):RATE1=ENTRY
HG 4030 ON VECTOR-27 GOTO 3010,5010,4510,
   4510
QE 4040 GOTO 7010
WH 4501 REM RATE OF RETURN INVESTMENT 2
DR 4510 POSITION 29,7:? "■■";:GOSUB 100
NS 4520 IF PTR<>1 THEN TRAP 4530:ENTRY=VA
   L(RESPONSE$(1,PTR-1)):RATE2=ENTRY
DU 4530 ON VECTOR-27 GOTO 3510,5510,4010,
   4010
QO 4540 GOTO 7010
LY 5001 REM REINVEST DIVIDENDS?  1
ZV 5010 POSITION 12,9:? "■■";:GOSUB 100
SQ 5020 IF PTR<>1 THEN REIN1$=RESPONSE$
MD 5030 ON VECTOR-27 GOTO 4010,6010,5510,
   5510
QF 5040 GOTO 7010
NN 5501 REM REINVEST DIVIDENDS?  2
BJ 5510 POSITION 32,9:? "■■";:GOSUB 100
TX 5520 IF PTR<>1 THEN REIN2$=RESPONSE$
IR 5530 ON VECTOR-27 GOTO 4510,6510,5010,
   5010
QP 5540 GOTO 7010
KD 6001 REM LOAD ON REINVESTED DIV? 1
RN 6010 POSITION 16,11:? "■■";:GOSUB 100
OR 6020 IF PTR<>1 THEN LDREIN1$=RESPONSE$
RF 6030 ON VECTOR-27 GOTO 5010,2510,6510,
   6510
QG 6040 GOTO 7010
LV 6501 REM LOAD ON REINVESTED DIV? 2
TB 6510 POSITION 36,11:? "■■";:GOSUB 100
QC 6520 IF PTR<>1 THEN LDREIN2$=RESPONSE$
HZ 6530 ON VECTOR-27 GOTO 5510,2510,6010,
   6010
PE 7001 REM SECOND SCREEN
DD 7010 GRAPHICS 0:POKE 710,4:? INV1$:POS
   ITION 22,0:? INV2$:V1=1-(LD1/100):V2=1
   -(LD2/100)
NP 7020 FOR YEAR=1 TO 20:? YEAR:POSITION
   5,YEAR
MU 7025 IF REIN1$<>"Y" AND REIN1$<>"y" TH
   EN V1=V1+RATE1/100:GOTO 7035
UY 7030 V1=V1+V1*(RATE1/100)*(1-LD1*(LDRE
   IN1$="Y" OR LDREIN1$="y")/100)
MG 7035 IF REIN2$<>"Y" AND REIN2$<>"y" TH
   EN V2=V2+RATE2/100:GOTO 7050
BZ 7040 V2=V2+V2*(RATE2/100)*(1-LD2*(LDRE
   IN2$="Y" OR LDREIN2$="y")/100)
SM 7050 ? (V1-1)*100:POSITION 22,YEAR:? (
   V2-1)*100
QU 7060 NEXT YEAR
HE 7070 ? :? "   <esc> to return to data
   entry":POKE 752,1
PI 7080 ? "   (*) to end program";
BC 7090 GET #1,X:IF X=27 THEN POKE 752,0:
   GOTO 1000
HM 7100 IF X=42 THEN GRAPHICS 0:END
UA 7110 GOTO 7090
KD 10000 DIM INV1$(15),INV2$(15),RESPONSE
   $(15),REIN1$(1),REIN2$(1),LDREIN1$(1),
   LDREIN2$(1)
AG 10010 CLOSE #1:OPEN #1,12,0,"K:"
ZA 10020 REM DEFAULT VALUES
MV 10030 INV1$=" ":INV2$=" "
ZA 10040 LD1=0:LD2=0
NC 10050 RATE1=7:RATE2=7
MU 10060 REIN1$="Y":REIN2$="Y"
GB 10070 LDREIN1$="N":LDREIN2$="N"
VR 10999 GOTO 1000:REM START MAIN BODY
```

lighting-fast command finder

ACTION! TOOLBOX

Article on page 52

LISTING 1

```
; ACTION! TOOLBOX
; BY KEVIN SHERRATT
; (c)1988, ANTIC PUBLISHING

MODULE
 BYTE Index,
      Match
 BYTE ARRAY String,
            Word,
            Comlist

PROC Wordfind()
 BYTE Start,
      Counter
 FOR Counter=Index TO String(0)
 DO
  IF String(Index)<>32 THEN
   EXIT
  FI
  Index==+1

 OD
 Start=Index
 FOR Counter=Index TO String(0)
 DO
  IF String(Index)=32 THEN
   EXIT
  FI
  Index==+1
 OD
 SCopyS(Word,String,Start,Index)
RETURN

PROC Matchup(BYTE ARRAY Command, List BY
TE Increment)
 BYTE Counter1,
      Counter2
 Match=0
 FOR Counter1=1 TO List(0) STEP Increment
 DO
  IF Command(1)=List(Counter1) THEN
   Match=1
   FOR Counter2=2 TO Increment
   DO
    IF List(Counter1+Counter2-1)=32 THEN
     EXIT
    ELSEIF Command(Counter2)<>List(Counter1+Co
unter2-1) THEN
     Match=0:EXIT
    FI
   OD
  FI
  IF Match=1 THEN
   EXIT
  FI
 OD
 IF Match=1 THEN
  Match=Counter1
 FI
RETURN

PROC Caller()
 Comlist="DOG CAT COW MULE"
 Print("TYPE ONE OF THE FOLLOWING: "
 PrintE(Comlist)
 InputS(String)
 Index=1
 Wordfind()
 Matchup(Word, Comlist, 4)
 PrintE(Word)
 PrintBE(Match)
RETURN
```