General Information#

Author: Kevin Sharratt
Language: ACTION!
Compiler/Interpreter: ACTION!
Published: ANTIC Vol. 7, #6 (10/ 88)

- Link

ACTION! Toolbox#

ATR-Image#

- Toolbox.atr

ACTION! file (ACT)#

- TOOLBOX.ACT

PDF file#

- actiontoolbx.PDF ; PDF-file of the ACTION! Toolbox

Lightning-fast command finder#

Two powerful and widlely 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 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 BYTE 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+Counter2-1) 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
```