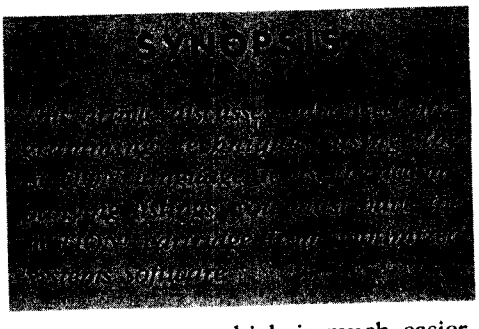
LINDA TAPSCOTT

An "interrupt" takes place in a computer system whenever one process takes precedence over a process that is being executed. It interrupts the lower-priority process so that it (the interrupt) can be executed first. Several interrupts are available on the Atari computers, including the display-list interrupt (DLI), the vertical-blank interrupt (VBI), and the system-timer interrupt, which was discussed in the January issue (**Antic**, "Page Flipping," p. 94).

Before **ACTION!** came along, you had to be able to program in machine language to use these interrupts. Clinton Parker, ACTION!'s author, may have envisioned that ACTION! programmers would continue to use machine code to write interrupt routines, installing blocks of machine-language codes generated by their assemblers into their ACTION! programs. But ACTION! is so fast that you can actually write a VBI or a system-timer interrupt in this high-level language, which is much easier than writing it in machine language.



Unfortunately, when an ACTION! VBI interrupts an ACTION! program, the two use the same space in memory to hold temporary math variables for calculation. Because of this, the interrupt routine can alter the variables from the interrupted routine. As a result, results can be quite unpredictable.

Mike Fitch of Optimized Systems Software (OSS), ACTION!'s publisher, has solved this dilemma with two short machine-language routines that save the contents of the temporary math registers to the stack at the beginning of the interrupt, and then restore them just before the interrupt ends. Mike calls these routines SAVETEMPS and GETTEMPS.

You use the DEFINE command to assign machine-language code blocks to SAVETEMPS and GETTEMPS. The accompanying ACTION! program demonstrates the use of GETTEMPS and SAVETEMPS in a VBI. It also produces an interesting effect on the screen. These routines are just what you need if you want to use interrupts written in ACTION!

David Plotkin is a chemical engineer with Standard Oil Company of California and an avid game programmer.

continued on next page

```
MODULE; VBI DEMO FOR ANTIC
DEFINE RTI="$4Ø",
       PHA="$48",
       PLA="$68",
       TXA="$8A",
       TAX="$AA",
       TYA="$98",
       TAY="$A8",
       JMP="$4C",
       XITVBV="$E462",
SAVETEMPS="[$A2 7 $B5 $A8 $48
                    $CA $1Ø $FA]",
GETTEMPS="[$A2 Ø $68 $95 $A8 $E8
                    $EØ 8 $DØ $F8]"
CARD SDLST=56Ø,VDSLST=512,
                VVBLKD=$224
BYTE NMIEN=$D4ØE,COLBK=$DØ1A,
     WSYNC=$D4ØA,COUNT=[Ø]
BYTE ARRAY DLIST
BYTE ARRAY CLRS(Ø)=[64 66 68 7Ø 72 74
     72 7Ø 68 66  64  66  68  7Ø  72
     74  72  7Ø  68  66  64  66  68
     7Ø 72 74 76 ]

PROC DLINT(); a DLI written in ACTION!
BYTE DUM
[PHA TXA PHA TYA PHA]
IF COUNT=26 THEN DUM=Ø
ELSE DUM=CLRS(COUNT) FI
WSYNC=1
COLBK=DUM
COUNT=COUNT+1
IF COUNT=27
THEN COUNT=Ø
FI
[PLA TAY PLA TAX PLA RTI]

PROC INIT7()
GRAPHICS(7)
SETCOLOR(Ø,2,1Ø) SETCOLOR(1,5,12)
SETCOLOR(2,Ø,Ø)
RETURN

PROC DLSETUP(); custom Display List
BYTE I
INIT7()
NMIEN=$4Ø
DLIST=SDLST
VDSLST=DLINT
FOR I=3Ø TO 4Ø
DO DLIST(I)=141 OD
FOR I=42 TO 54 STEP 2
DO DLIST(I)=141 OD
FOR I=57 TO 72 STEP 3
DO DLIST(I)=141 OD
FOR I=76 TO 84 STEP 4
DO DLIST(I)=141 OD
NMIEN=$CØ
RETURN

PROC ROTATE(); the VBI routine
BYTE HOLD,CTR,CNTR
SAVETEMPS; save the temp registers
HOLD=CLRS(26); save the last element
FOR CTR=Ø TO 25; the loop
DO CNTR=25-CTR; to count backwards,
                ; ACTION has no STEP-1
                ; statement
CLRS(CNTR+1)=CLRS(CNTR) OD; rotate
CLRS(Ø)=HOLD; put the last element
            ; into the first
GETTEMPS    ; get the temp registers
[JMP XITVBV]; exit the VBI

PROC VBINST(); install the VBI
NMIEN=Ø; turn off the interrupts
VVBLKD=ROTATE; vector to PROC ROTATE
NMIEN=$4Ø; turn the interrupts back on
RETURN

PROC DJD(); the driver routine, named
BYTE CRSINH=752; for a famous
                ; computer genius
VBINST() ; install the VBI
DLSETUP(); set up the Display List
CRSINH=1
PRINTE()
PRINTE("Antic Interrupts In ACTION!")
PRINT("          by DAVID PLOTKIN")
DO OD; an endless loop...
RETURN
```