Subject: At last! A free Action! Run-Time Library (LONG)
Posted by Anonymous on Sat, 22 Mar 1986 05:14:10 GMT
View Forum Message <> Reply to Message

Originally posted by: wa60&#64sdcc12.UUCP (paul van de graaf{})

Article-I.D.: sdcc12.558
Posted: Sat Mar 22 00:14:10 1986
Date-Received: Tue, 25-Mar-86 02:59:56 EST
Organization: U.C. San Diego, Academic Computer Center
Lines: 388

I posted this a week ago, but apparently the UUCP at a local site was broken.
If you've seen it before, my apologies. I'm still unsure if this posting will
get out; so I'd appreciate a reply by mail if you get it running on your Atari.

Copyright (c) 1986, Paul van de Graaf
Permission is granted for Usenet & Arpanet users to copy this article for their
own use. It is NOT to be distributed beyond Usenet & Arpanet sites (or normal
tributaries thereof); sold, or traded without permission from the author.

[The reason for the above notice is that I am considering getting this article
and programs published in either Analog or Antic. I recently had my printer
and modem destroyed by a power-surge, and could use some cash to replace them.
(I accidently plugged them into my un-surge-protected power strip. BLAST!)
Consider this a Beta-Test of the article & programs. They will either be
published in another form, or released into the public domain. I would
especially like to keep them off of Compuserve et al., for the time being.]

Here's the program you Action! programmers have all been waiting for!
A free Run-Time Library of sorts for the Action! language.

But first a few disclaimers:
1.) This run-time library requires at least 48K of memory. It will not run
when cartridges are installed, as it uses memory from $A000 to $B5FF.

2.) The library has only been tested for version 3.6 of the Action! cartridge.
To find out which version you have, check location $B000. It contains
the version number in hex. If it's not $36, then the library might not
work. I'd try it anyway, since it might work, or there probably is an
easy patch. See the text below for more details.

3.) Since this library requires that you make a copy of part of the Action!
cartridge, there may be some issue of piracy. I consider this a legal
back-up copy of the cartridge as stated in the licensing agreement, but

OSS's attorneys might think otherwise. Proceed at your own risk!
I take no legal responsibility for your actions.

4.) Because the routines PrintF() and Error() have some bank-switching in them, they do not work normally. PrintF() works fine, except when asked to print CARDs in hex, it prints them in decimal. [ie. PrintF("%H",foo)] FIXACT.ACT changes the PrintF routine to make it act this way, otherwise it would lock up, since it tries to call bank-switched code. I personally never cared for this routine anyway, because it always prints a leading "$", and you can't get it to suppress leading zeros. I'm sure you could write a better PrintF() yourself. The Error() routine works about the same as before, but instead of returning to the monitor after an error, it returns to DOS.

Introduction:

I wrote this library, because I can't understand why OSS decided not to include a run-time library with the Action! system. The run-time library routines are only about 2K of the cartridge. They could have easily made some facility to link the routines into the code. Perhaps they were short on space in the cartridge, or maybe this is some marketing ploy to sell run-time library disks. I haven't seen or heard any details on the run-time library disk, other than you can get one for $30.00. I'm sure their library is a much more elegant, as I went the quick and dirty route. Instead of writing a full-blown linker and relocator to link in the run-time library, I opted for a simpler approach.

How it works:

My home-brew run-time library consists of 4 programs:

RTL.ACT	Sets up memory for the run-time library.
ACTION.COM	A copy of part of the Action! cartridge saved to a binary file.
FIXACT.ACT	Fixes the self-modifying and bank-switching code in ACTION.COM
CLEANUP.ACT	Cleans up after the run-time library.

The library works like this: RTL.ACT is run first. It is a one-shot program which makes your 48K or greater machine look like a 40K computer by closing the editor, bumping down RAMTOP, and opening the editor again. It also sets up the error handler, and opens the keyboard as channel 7. Channel 7 is opened because the Action! cartridge keeps channel 7 open all the time, and some programs take advantage of this channel. After RTL.ACT is run, ACTION.COM is loaded, and the run-time library is in place. It's as if the cartridge is inserted, except this cartridge can be written to and can't do bank-switching. After ACTION.COM is loaded, a compiled Action! program can be run, and it will call the library routines that were normally on the cartridge, but now have been copied and loaded into RAM. After the compiled Action! program is run, you have the option of running CLEANUP.ACT to reset the machine so it looks like it has 48K of memory again.

There's just one problem, though. The Action! cartridge has some self-modifying code in it that can alter the routines in the run-time library as it runs. It works fine when it's in a ROM cartridge, but it gets messed up when in RAM. This code was put in to make life difficult for pirates who might be considering ripping-off the Action! cartridge. I took a disassembler to ACTION.COM and I'm fairly certain I've located all the self-modifying code. While disassembling I also located the bank-switching code mentioned above. FIXACT.ACT replaces the tainted code with NOPs, thereby de-fusing the logic bombs.

Contrary to what you might expect, you CAN write programs in Action! that will run without the run-time library. You can't call any library routines, do a multiply, divide, MOD, RSH, or LSH; use the trace or error facilities, or have a PROC/FUNC that has more than 3 bytes of parameters. These constraints make programming difficult, but not impossible. RTL.ACT is written in this manner. It uses CIO calls to do opens, closes and puts to IO channels, rather than calling the run-time library routines.

Dim the lights, it's time for the sermon:

I believe it's important to mention the distinction between what I'm doing here, and piracy. I am only making a backup of the cartridge, and this backup is taking the place of the cartridge when I want to run a program without the cartridge inserted. If the Action! system were better designed, I needn't bother with this. I don't think this constitutes piracy unless I give/sell a copy of the cartridge to somebody, or use it on a different computer. I only give the patches to fix code in the run-time library in FIXACT.COM-- NOT for the editor, monitor, or compiler sections of the cartridge. In fact after looking at most of the code in the Action! cartridge, I would guess any effort to pirate the cartridge, other than building a similar bank-switching arrangement, would greatly reduce the functionality of Action!. Besides-- you can get it for $29.00 at some mail-order shops. Despite their mistake with the run-time library for Action!, OSS is a quality software vendor, and they deserve our support.

Making the run-time library:

Step 1:
  Go to DOS with the Action! cartridge installed.
  Select the K command to write a binary file.
  At the prompt: SAVE - GIVE FILE,START,END,(INIT,LOAD)
  type: ACTION.COM,A000,B5FF

Step 2:
  Compile RTL.ACT, and from the monitor type: W "RTL.COM"
  Compile FIXACT.ACT, and from the monitor type: W "FIXACT.COM"
  Compile CLEANUP.ACT, and from the monitor type: W "CLEANUP.COM"

Step 3:

Remove the Action! cartridge and re-boot the system.
(on the XL & XE be sure to press the Option key when booting)
Select the L command from the DOS menu to load a binary file.
At the prompt: LOAD FROM WHAT FILE? type: RTL.COM
The screen will clear, and you'll return to DOS.
DOS gets confused because the screen was cleared and the cursor was moved.
Press return, and DOS will redraw the menu and come back to its senses.
Select the L command again.
At the prompt: LOAD FROM WHAT FILE? type: ACTION.COM
After the file has loaded, select the L command again.
At the prompt: LOAD FROM WHAT FILE? type: FIXACT.COM
After the file loads and runs, select the K (binary save) command.
At the prompt: SAVE - GIVE FILE,START,END,(INIT,RUN)
type: ACTION.COM,A000,B5FF

Three ways to use the run-time library:

1.) Manually:

Boot DOS without any cartridges installed (press option while booting on an XL or XE). Load RTL.COM from DOS, then load ACTION.COM from DOS, and finally load the compiled Action! program you wish to run. This is rather tedious process, but it's ideal for quickly testing the run-time library.

2.) With an AUTORUN.SYS file:
Make a compound binary load file from DOS. Select the C command and at the prompt: COPY--FROM,TO? type: ACTION.COM,RTL.COM/A
This will append ACTION.COM to the end of RTL.COM. When loaded with the L command, RTL.COM will now run the old RTL.COM program, and then load ACTION.COM. If you rename RTL.COM as AUTORUN.SYS, this process will automatically occur at boot time. After the boot, you can load compiled Action! files from DOS, and they should run properly. You probably would want to use this technique if you have a bunch of compiled Action! programs on the same disk.

3.) A fully linked file:
Take a file like the AUTORUN.SYS file above, and append your compiled Action! program to it. IE. choose the C command from DOS, and type "YOUR_PROGAM.COM, AUTORUN.SYS/A" at the prompt. If you want to have the run-time library clean up after itself when your program returns to DOS, you'll also want to append the CLEANUP.COM file too. IE. "C(opy) CLEANUP.COM,AUTORUN.SYS/A". You might want to change the name of the file to something other than AUTORUN.SYS, if you don't want it to run automatically at boot time. This technique has a disadvantage -- after all that appending, the load file gets quite large!
I have tried to pare the size down a little by only saving part of the Action! cartridge. Still, the ACTION.COM file is much larger than the 2K of routines we need. The problem is that the run-time library is scattered throughout the address space of the cartridge, and it's too much trouble to save it in

several fragmented files.

If it doesn't work:

If the run-time library doesn't run correctly, make sure you didn't make an error along the way. If you don't have version 3.6, try saving all of the cartridge in steps 1 & 3 (ie. K "ACTION.COM,A000,BFFF"). Your cartridge may use code in the $B600 - $BFFF area. If the library still doesn't work, try skipping the L "FIXACT.COM" step. It should probably work now, but ACTION.COM will likely contain self-modifying and bank-switching code. You're going to have to take a disassembler to ACTION.COM to ferret out the self-modifying and bank-switching code. Then you can bypass the code with NOPs.

Self modifying code is fairly easy to identify. It's usually something obvious like: "DEC $A000,X", however it might be some sort of manipulation of a variable like: "LDA #0; STA $A0; LDA #$A0; STA $A1; INC ($A0),X". Bank-switching code is very easy to spot. It involves a write to any address from $D500 - $D5FF. For example: STA $D509,X.

You can find the entry points for most of the library routines by using the monitor. ex. ? PrintF. A few routines can't be found this way. They are: the math routines: MULTIPLY, DIVIDE, MOD, LSH, RSH; a routine which saves parameters and checks for a break key abort when PROCs or FUNCs with more than 3 bytes of parameters are called, and a routine to print a PROC/FUNC name and its arguments when trace is set. The way I found the entry points to these routines was to disassemble a compiled Action! programs which called these routines. Good Luck! Please send me mail if you find a patch to FIXACT.ACT which works for your version.

And now the programs:

After all this build-up, the programs are probably a let-down. They're quite straight-forward if you know the CIO routine. If you don't, you might want to get the technical notes from Atari (if they still sell them!). It probably will be easier to down-load them in one big file, and then hack it to pieces with the Action! editor.

I'd like to know if this library works with the other versions of the Action! cartridge. Send me mail if it works with your version.

Thanks,

Paul van de Graaf sdcsvax!sdcc6!sdcc12!wa60 U. C. San Diego

--- CUT HERE ---

```
; RTL.ACT
; Copyright (c) 1986,  Paul van de Graaf
```

```
; Permission is granted for members of
; Usenet & Arpanet to use this program.
; It is NOT to be distributed beyond
; Usenet & Arpanet sites; sold, or traded
; without consent of the author.

SET $E   = $4000
SET $491 = $4000

DEFINE R     =   "4",  RW    =  "12",
       JMP   = "$4C",  EOL   = "$9B",
       CLEAR = "125",  OPEN  = "$03",
       PUTC  = "$0B",  CLOSE = "$0C"

BYTE IC0CMD  = $342,  IC7CMD  = $3B2,
     IC0AUX  = $34A,  IC7AUX  = $3BA,
     IC0AUX2 = $34B,  IC7AUX2 = $3BB,
     RAMSIZ  = $2E4,  RAMTOP  = $6A

CARD IC0BUF  = $344,  IC7BUF  = $3B4,
     IC0BLEN = $348,  IC7BLEN = $3B8

CHAR ARRAY editor = [ 'E ': EOL ],
           keybrd = [ 'K ': EOL ]

PROC CIO = $E456(BYTE Areg, Xreg)

PROC Put(CHAR chr)

  IC0CMD = PUTC
  IC0BLEN = 0
  CIO(chr,0)
RETURN

PROC Print(CHAR ARRAY msg)

  IC0CMD = PUTC
  IC0BUF = msg + 1
  IC0BLEN = msg(0)
  CIO(0,0)
RETURN

PROC main()
  BYTE POINTER ErrorAdr = Error

  IF RAMTOP >= $C0 THEN
    IC0CMD = CLOSE
    CIO(0,0)      ; Close(0)
```

```
    ; Bump down top of memory so the
    ; copy of the Action! cartridge
    ; won't clobber display memory.

    RAMTOP = $A0  RAMSIZ = $A0

    IC0CMD = OPEN
    IC0BUF = editor
    IC0AUX = RW
    IC0AUX2 = 0
    CIO(0,0)      ; Open(0,"E:",RW,0)

    IC7CMD = OPEN
    IC7BUF = keybrd
    IC7AUX = R
    IC7AUX2 = 0
    CIO(0,$70)    ; Open(7,"K:",R,0)

    ; Set up the Error Handler:
    ; On an Error, Action! jumps to
    ; the address Error, which we
    ; set to contain a JMP $A3A2.

    ErrorAdr^ = JMP
    Error = $A3A2
  ELSE
    Put(CLEAR)
    Print("The Home-Brew Action! ")
    Print(" Run-time Library")
    Print("requires at least 48K ")
    Print("bytes of memory.")
    Put(EOL)
    Put(EOL)
    Print("Please Remove Cartridge(s)!")
    DO
    OD          ; Hang in a loop.
  FI
RETURN

--- CUT HERE ---

; FIXACT.ACT
; Copyright (c) 1986,  Paul van de Graaf
; Permission is granted for members of
; Usenet & Arpanet to use this program.
; It is NOT to be distributed beyond
; Usenet & Arpanet sites; sold, or traded
```

```
; without consent of the author.

SET $E   = $4000
SET $491 = $4000

DEFINE NOP = "$EA", RTS = "$60"

PROC main()
  ; Change STA $A04A in the
  ; multiply routine to NOPs.

  Poke($A036,NOP)
  Poke($A037,NOP)
  Poke($A038,NOP)

  ; Change JMP $B889 in the default
  ; error handler to an RTS.

  Poke($A3A2,RTS)

  ; Change DEC $A395,X in the CIO
  ; error check routine to NOPs.

  Poke($A3BD,NOP)
  Poke($A3BE,NOP)
  Poke($A3BF,NOP)

  ; Change JSR $B8C2 and JMP $A339
  ; in the PrintF() routine to NOPs.

  Poke($A438,NOP)  Poke($A439,NOP)
  Poke($A43A,NOP)  Poke($A43B,NOP)
  Poke($A43C,NOP)  Poke($A43D,NOP)

  ; Change STA ($A5),Y in the
  ; Close() routine to NOPs.

  Poke($B02A,NOP)
  Poke($B02B,NOP)
RETURN

--- CUT HERE ---

; CLEANUP.ACT
; Copyright (c) 1986,  Paul van de Graaf
; Permission is granted for members of
; Usenet & Arpanet to use this program.
; It is NOT to be distributed beyond
```

```
; Usenet & Arpanet sites; sold, or traded
; without consent of the author.

SET $E   = $4000
SET $491 = $4000

DEFINE RW = "12"

BYTE RAMSIZ = $2E4, RAMTOP = $6A

PROC main()

  Close(7)
  Close(0)
  RAMSIZ = $C0  RAMTOP = $C0
  Open(0,"E:",RW,0)
RETURN
```