ACTION! Relocator#

Instructions for:

  RELGEN.ACT:== Relocation Generator
& REOCAT.ACT:== Run-time Relocator

  These programs were intended to create a self-relocating object file from either an ACTION! compiled program or an Assembled program. The original object file must be a single-stage boot with only one origin except for the trailing run or init address. The following instructions detail the steps to make the target object file. This file may be appended to other binary load files and may have other binary files appended to it.. The program will load at the next possible page boundary (increment of 256) after MEMLO. Because RELGEN compares two versions of you object file, you may want to init all variables to zero to keep the relocation table at a minimum. Stray data in the uninitialized variables may be interpreted as machine code that needs relocating.

1) Compile (or Assemble) your code at a convenient area but not conflicting with DOS. In ACTION!, use the following commands to force the program's origin to a specified value ($3000 for example):
```
  SET 14=$3000
  SET $491=$3000
```

2) Re-Compile your code at $100 higher than the first. For the above example, this would be at $3100.

3) From the ACTION! monitor, RUN the program RELGEN.ACT. It will prompt you for the filenames for the two object code files that you compiled above. Remember to give the Dn: prefix to the filenames. The program will compare the two object files and note their differences as offsets into the file. This information is saved in ACTION! form in a file with the original name and a ".GEN" extention. This will be used in the next step. Also, the program creates

an object file image of the original but with an origin of zero. This is done to make the relocation process easier and this file, with a ".REL" extention will be used in step 5.

  Note: RELGEN.ACT requires four open DOS files simultaneously. By default, DOS usually has buffers for only 3. You must use the command:

```
  SET $709=4
```

in the ACTION! monitor and type D for DOS. Rewrite DOS to the disk and reboot. Now, DOS will allow the four files to be opened.

4) Now, Read the program RELOC.ACT into the ACTION! Editor. This is a "generic" run-time relocator. The file generated with the RELGEN.ACT program (with the ".GEN" extention) must be merged into this program with the editor Read function. Position the cursor where instructed and read in the file. Compile this code but be sure that it is SET to compile above the expected end of YOUR program's target location. Save this object code to disk and go to DOS.

5) Using the DOS Copy command, append the ".REL" file generated in RELGEN.ACT, to the merged relocator file saved in step 4. For example:

```
C
Copy from,to:
TEST.REL,AUTORUN.SYS/A
```

This assumes that you saved the file in Step 4 as AUTORUN.SYS.

6) Finally, the appended file can be loaded from DOS or named AUTORUN.SYS as above for permanent applications.

If you have question, send E-Mail to:
  John DeMar 71066,337 on Compuserve or leave a message on the ACE-BASE BBS at (315)451-7747. Good Luck!

RELGEN.ACT#

```
MODULE ;RELGEN.ACT

;COPYRIGHT 1984, QMI, JS DeMar
```

```
;REV. 1.1, March 20, 1984

;OBJECT CODE RELOCATION GENERATOR for
;ACTION! compiled binary-load files.

;WARNING!!! This program requires
;four OPEN files simultaneously.
;Be sure that DOS is configured for
;this. With DOS 2.0, set $709 equal
;to at least 4, rewrite DOS and
;reboot.

;Requires the second file compiled
;at any even page increment higher
;than the first file, for example:
;$3000 and $3100.

;Generates a table of the locations
;that require relocating and saves
;it in a ".GEN" file in ACTION!.
;  The ".REL" file is the original
;object code with an origin of "0".
;The actual relocator is compiled
;from the generic relocator source
;called "RELOC.ACT" merged with the
;".GEN" file generated here. Append
;".REL" file to that code and it
;will load and relocate to MEMLO.

DEFINE in1="1",
       in2="2",
       out1="3",
       out2="4"

BYTE abrt

;---------------------------------

PROC MyError(BYTE a,x,y)

IF y=170 THEN
  PrintE("ERROR File not found!")
ELSE
  Print("ERROR! ")
  PrintBE(y)
FI
abrt=1
RETURN
;---------------------------------

PROC Ferror()

BYTE t,clock=$14

PrintE("ERROR in Output filespec!")
t=clock-$80
DO
UNTIL t=clock
OD
```

```
RETURN
;---------------------------------

PROC EndIt()

Close(in1)
Close(in2)
Close(out1)
Close(out2)
RETURN
;---------------------------------

PROC Main()

CARD start1,start2,end1,end2
CARD offsets,offsete,i,count,hits
CARD test1,test2,old1,old2,old3,old0
BYTE x,z,j,wnum,d1,d2,
     sthigh

BYTE ARRAY fname1(18),fname2(18),
           fnameout1(18),fnameout2(18)

DO
PrintE("Relocation code Generator II   ")
PrintE("JS DeMar, 8/84")
PutE()
PrintE("Requires two code files compiled")
PrintE("with an offset of $0100.")
PutE()

Print("Filespec for code A >")
InputMD(device,fname1,18)
PutE()
Print("Filespec for code B >")
InputMD(device,fname2,18)
PutE()

Scopy(fnameout1,fname1)
SCopy(fnameout2,fnameout1)
j=1
IF fnameout1(1)#'D
      OR fnameout1(0)<4 THEN
  Ferror()
ELSEIF fnameout1(2)=': THEN
  z=0
ELSEIF fnameout1(3)=': THEN
  z=1
FI
DO
  x=fnameout1(j)
  j==+1
  IF x=$20 THEN
    EXIT
  ELSEIF x='. THEN
    EXIT
  ELSEIF j>fnameout1(0) THEN
    j==+1
    EXIT
```

```
  ELSEIF j>11+z THEN
    Ferror()
  FI
OD

fnameout1(j-1)='.
fnameout1(j)='G
fnameout1(j+1)='E
fnameout1(j+2)='N
fnameout1(0)=j+2

j=1
IF fnameout2(1)#'D
      OR fnameout2(0)<4 THEN
  Ferror()
ELSEIF fnameout2(2)=': THEN
  z=0
  EXIT
ELSEIF fnameout2(3)=': THEN
  z=1
  EXIT
FI
OD
DO
  x=fnameout2(j)
  j==+1
  IF x=$20 THEN
    EXIT
  ELSEIF x='. THEN
    EXIT
  ELSEIF j>fnameout2(0) THEN
    j==+1
    EXIT
  ELSEIF j>11+z THEN
    Ferror()
    EXIT
  FI
OD

fnameout2(j-1)='.
fnameout2(j)='R
fnameout2(j+1)='E
fnameout2(j+2)='L
fnameout2(0)=j+2

Print("Generation file = ")
PrintE(fnameout1)
Print("Relocation file = ")
PrintE(fnameout2)

Error=MyError
abrt=0
Close(in1)
Close(in2)
Close(out1)
Close(out2)
Open(in1,fname1,4)
Open(in2,fname2,4)
IF abrt=1 THEN
```

```
  Close(1)
  Close(2)
  RETURN
FI
Open(out1,fnameout1,8)
Open(out2,fnameout2,8)

x=GetD(in1) ;throw away two $FF's.
x=GetD(in1)
PutD(out2,$FF)
PutD(out2,$FF)
x=GetD(in1)
PutD(out2,x)
start1=x    ;start addr of file1.
x=GetD(in1)
PutD(out2,x)
start1==+(x*256)
x=GetD(in1)
PutD(out2,x)
end1=x
x=GetD(in1)
PutD(out2,x)
end1==+(x*256) ;end addr of file1.

x=GetD(in2) ;throw away two $FF's.
x=GetD(in2)
x=GetD(in2)
start2=x    ;start addr of file2.
x=GetD(in2)
start2==+(x*256)
x=GetD(in2)
end2=x
x=GetD(in2)
end2==+(x*256) ;end addr of file2.

offsets=start2-start1
sthigh=start1/256
offsete=end2-end1


PrintDE(out1,"MODULE")
PrintD(out1,";For file ")
PrintDE(out1,fnameout2)
PrintDE(out1,"")
Print("Code starts at ")
PrintD(out1,"CARD start=[")
PrintCE(start1)
PrintCD(out1,start1)
PrintDE(out1,"]")
Print("   and ends at ")
PrintD(out1,"CARD finish=[")
PrintCE(end1)
PrintCD(out1,end1)
PrintDE(out1,"]")
Print("Compile offset was ")
PrintCE(offsets)

IF offsete#offsets THEN
  PrintE("Diferrent size files!")
```

```
  PrintE("ABORTED!")
  EndIt()
  RETURN
FI
PrintDE(out1,"")
PrintD(out1,"CARD ARRAY otable=[")
wnum=0
hits=0
count=0
FOR i=start1 TO end1
DO
  d1=GetD(in1)
  d2=GetD(in2)
  IF d1#d2 THEN
    hits==+1
    IF wnum=0 THEN
      PrintD(out1,"  ")
      Print(" ")
    ElSE
      PrintD(out1," ")
      Print(" ")
    FI
    PrintCD(out1,count)
    Print(" ")
    PrintC(count)
    wnum==+1
    IF wnum>4 THEN
      PrintDE(out1,"")
      PrintE("")
      wnum=0
    FI
    d1==-sthigh
  FI
  PutD(out2,d1)
  count==+1
OD
FOR i=0 TO 2
DO
  d1=GetD(in1)
  d2=GetD(in1)
OD
test1=d1
test1==+(d2*256)
IF test1>=start1 AND test1<=end1 THEN
  PrintDE(out1,"]")
  PrintE("]")
  PrintD(out1,"CARD hits=[")
  PrintCD(out1,hits)
  PrintDE(out1,"]")
  PrintDE(out1,"")
  Print("CARD hits=[")
  PrintC(hits)
  PrintE("]")
  PrintE("")
  PrintD(out1,"CARD runaddr=[")
  Print("CARD runaddr=[")
  test1==-start1
  PrintCD(out1,test1)
  PrintC(test1)
```

```
  PrintDE(out1,"]")
  PrintE("]")
ELSE
  PrintE("No Run Address! - ABORTED!")
FI
PrintE("")
PrintDE(out1,"")
EndIt()
PrintE("Finished!"")
RETURN
```

RELOC.ACT#

```
MODULE ;RELOCATE.ACT

;Run-time Relocator Code.
;For use with RELGEN.ACT
;COPYRIGHT 1984, JS DeMar
;Rev. 2.0, August 17,1984
;---------------------------------

SET 14=$6000
SET $0491=$6000
;---------------------------------
;The beginning of the relocator
;table and code should be higher
;than the end of the original
;compiled program. But, there must
;be enough space left for the table
;and the relocator code itself!
;---------------------------------



;---------------------------------
;Read the ".GEN" file above here.
;---------------------------------
;Compile this after reading in the
;".GEN" file above. Then append this
;code to the ".REL" file using the
;DOS C (COPY) command with /A after
;the filenames:
; PROGRAM.OBJ,PROGRAM.REL/A
;Then rename the ".REL" file to
;AUTORUN.SYS to run at boot-time.
;---------------------------------

PROC Relocate()

BYTE offset,memlohi=$02E8,x,y
CARD memlo=$02E7,i,j,top,entry
CARD POINTER p
BYTE ARRAY newplace

[$8E y $4E y $4E y $4E y $4E y]

newplace=memlo
newplace==&$FF00
offset=memlohi
```

```
i=memlo&$00FF
IF i#0 THEN
  newplace==+$0100
  offset==+1
FI

j=0
FOR i=start TO finish
DO
  p=i
  x=p^
  newplace(j)=x
  j==+1
OD

FOR i=0 TO hits-1
DO
  entry=otable(i)
  newplace(entry)==+offset
OD
runaddr==+newplace
[$6C runaddr]
```