

```

1      .INCLUDE GRAFTYPES.TEXT
2      ;-----
3      ;
4      ;
5      ;      ****      *****      ***      ***      ***      *      *      ***
6      ;      *      *      *      *      *      *      *      *      *      *      *
7      ;      *      *      *      *      *      *      *      **      *      *
8      ;      ****      ***      *      **      *      *      *      *      *      *      ***
9      ;      *      *      *      *      *      *      *      *      *      **      *
10     ;      *      *      *      *      *      *      *      *      *      *      *      *
11     ;      *      *      *****      ***      ***      ***      *      *      ***
12     ;
13     ;
14     ;
15     ; QuickDraw Routines to operate on Regions.
16     ;
17
18     .PROC StdRgn,2
19     .REF CheckPic,PutPicVerb,DPutPicByte,PutPicRgn
20     .REF PutRgn,FrRgn,PushVerb,DrawRgn
21     ;-----
22     ;
23     ; PROCEDURE StdRgn(verb: GrafVerb; rgn: RgnHandle);
24     ;
25     ; A6 OFFSETS OF PARAMS AFTER LINK:
26     ;
27     PARAMSIZE      .EQU      6
28     VERB            .EQU      PARAMSIZE+8-2      ;GRAFVERB
29     RGN             .EQU      VERB-4            ;LONG, RGNHANDLE
30
31     LINK           A6,#0                        ;NO LOCALS
32     MOVEM.L D6-D7/A2-A4,-(SP)                  ;SAVE REGS
33     MOVE.B VERB(A6),D7                          ;GET VERB
34     JSR CHECKPIC                                     ;SET UP A4,A3 AND CHECK PICS
35     BLE.S NOTPIC                                     ;BRANCH IF NOT PICS
36
37     MOVE.B D7,-(SP)                                ;PUSH VERB
38     JSR PutPicVerb                                  ;PUT ADDITIONAL PARAMS TO THEPIC
39     MOVE #$80,D0                                    ;PUT RGNNOUN IN HI NIBBLE
40     ADD D7,D0                                       ;PUT VERB IN LO NIBBLE
41     JSR DPutPicByte                                ;PUT OPCODE TO THEPIC
42     MOVE.L RGN(A6),-(SP)                          ;PUSH RGNHANDLE
43     JSR PutPicRgn                                  ;PUT REGION TO THEPIC
44
45 NOTPIC MOVE.L RGN(A6),-(SP)                      ;PUSH RGNHANDLE
46     JSR PushVerb                                    ;PUSH MODE AND PATTERN
47     TST.B D7                                       ;IS VERB FRAME ?
48     BNE.S NOTFR                                    ;NO, CONTINUE

```

```

49         TST.L    RGNSAVE(A3)                ;YES, IS RGNSAVE TRUE ?
50         BEQ.S    NOTRGN                     ;NO, CONTINUE
51         MOVE.L    RGN(A6),-(SP)              ;YES, PUSH RGNHANDLE
52         MOVE.L    RGNBUF(A4),-(SP)          ;PUSH RGNBUF
53         PEA       RGNINDEX(A4)              ;PUSH VAR RGNINDEX
54         PEA       RGNMAX(A4)                ;PUSH VAR RGNMAX
55         JSR       PutRgn                    ;ADD INVERSION PTS TO THERGN
56
57 NOTRGN    JSR     FrRgn                     ;FrRgn(rgn,pnMode,pnPat)
58         BRA.S     GOHOME
59
60 NOTFR     JSR     DrawRgn                  ;DrawRgn(rgn,mode,pat);
61
62 GOHOME    MOVEM.L (SP)+,D6-D7/A2-A4         ;RESTORE REGS
63         UNLINK    PARAMSIZE,'STDRGN'
64
65
66
67         .PROC    FrameRgn,1
68         .DEF     CallRgn,PaintRgn,EraseRgn,InvertRgn,FillRgn
69         .REF     StdRgn
70 ;-----
71 ;
72 ;   PROCEDURE FrameRgn(* rgn: RgnHandle *);
73 ;
74         MOVEQ     #FRAME,D0                ;VERB = FRAME
75         BRA.S     CallRgn                  ;SHARE COMMON CODE
76
77
78 ;-----
79 ;
80 ;   PROCEDURE PaintRgn(* rgn: RgnHandle *);
81 ;
82 PaintRgn
83         MOVEQ     #PAINT,D0                ;VERB = PAINT
84         BRA.S     CallRgn                  ;SHARE COMMON CODE
85
86
87 ;-----
88 ;
89 ;   PROCEDURE EraseRgn(* rgn: RgnHandle *);
90 ;
91 EraseRgn
92         MOVEQ     #ERASE,D0                ;VERB = ERASE
93         BRA.S     CallRgn                  ;SHARE COMMON CODE
94
95
96 ;-----

```

```

97 ;
98 ; PROCEDURE InvertRgn(* rgn: RgnHandle *);
99 ;
100 InvertRgn
101     MOVEQ    #INVERT,D0                ;VERB = INVERT
102     BRA.S    CallRgn                  ;SHARE COMMON CODE
103
104
105 ;-----
106 ;
107 ; PROCEDURE FillRgn(* rgn: RgnHandle; pat: Pattern *);
108 ;
109 FillRgn
110     MOVE.L    (SP)+,A0                ;POP RETURN ADDR
111     MOVE.L    (SP)+,A1                ;POP ADDR OF PATTERN
112     MOVE.L    A0,-(SP)                ;PUT RETURN ADDR BACK
113     MOVE.L    GRAFGLOBALS(A5),A0      ;POINT TO LISAGRAF GLOBALS
114     MOVE.L    THEPORT(A0),A0          ;GET CURRENT GRAFPORT
115     LEA        FILLPAT(A0),A0         ;POINT TO FILLPAT
116     MOVE.L    (A1)+,(A0)+            ;COPY PAT INTO FILLPAT
117     MOVE.L    (A1)+,(A0)+            ;ALL EIGHT BYTES
118     MOVEQ     #FILL,D0                ;VERB = FILL
119     BRA.S     CallRgn                  ;SHARE COMMON CODE
120
121
122
123 ;-----
124 ;
125 ; PROCEDURE CallRgn(rgn: RgnHandle);
126 ;
127 ; code shared by FrameRgn, PaintRgn, EraseRgn, InvertRgn, and FillRgn.
128 ; enter with verb in D0.
129 ;
130 CallRgn
131     MOVE.L    (SP)+,A0                ;POP RETURN ADDR
132     MOVE.L    (SP)+,A1                ;POP RGN
133     MOVE.B     D0,-(SP)                ;PUSH VERB
134     MOVE.L     A1,-(SP)                ;PUSH RGN
135     MOVE.L     A0,-(SP)                ;RESTORE RETURN ADDR
136     MOVE.L     GRAFGLOBALS(A5),A0      ;POINT TO LISAGRAF GLOBALS
137     MOVE.L     THEPORT(A0),A0          ;GET CURRENT GRAFPORT
138     MOVE.L     GRAFPROCS(A0),D0        ;IS GRAFPROCS NIL ?
139     LEA        STDRGN,A0
140     BEQ.S      USESTD                  ;YES, USE STD PROC
141     MOVE.L     D0,A0
142     MOVE.L     RGNPROC(A0),A0          ;NO, GET PROC PTR
143 USESTD JMP      (A0)                  ;GO TO IT
144

```

```

145
146     .PROC DrawRgn,3
147     .REF  RgnBlit
148 ;-----
149 ;
150 ;  PROCEDURE DrawRgn(rgn: RgnHandle; mode: INTEGER; pat: Pattern);
151 ;
152 ;  A6 OFFSETS OF PARAMS AFTER LINK:
153 ;
154 PARAMSIZE      .EQU      10
155 RGN             .EQU      PARAMSIZE+8-4           ;LONG, RGNHANDLE
156 MODE           .EQU      RGN-2                   ;WORD
157 PAT            .EQU      MODE-4                   ;LONG, ADDR OF PATTERN
158
159     LINK        A6,#0
160     MOVE.L      GRAFGLOBALS(A5),A0                ;POINT TO LISAGRAF GLOBALS
161     MOVE.L      THEPORT(A0),A0                    ;GET CURRENT PORT
162     TST         PNVIS(A0)                          ;IS PNVIS NEG ?
163     BMI.S       DONE                               ;YES, QUIT
164     PEA         PORTBITS(A0)                       ;PUSH SRCBITS
165     MOVE.L      (SP),-(SP)                         ;PUSH DSTBITS
166     PEA         PORTBITS+BOUNDS(A0)                ;PUSH SRCRECT
167     MOVE.L      (SP),-(SP)                         ;PUSH DSTRECT
168     MOVE        MODE(A6),-(SP)                     ;PUSH MODE
169     MOVE.L      PAT(A6),-(SP)                      ;PUSH PAT
170     MOVE.L      CLIPRGN(A0),-(SP)                  ;PUSH CLIPRGN
171     MOVE.L      VISRGN(A0),-(SP)                   ;PUSH VISRGN
172     MOVE.L      RGN(A6),-(SP)                      ;PUSH RGN
173     JSR         RGNBLT                             ;CALL RGNBLT
174 DONE    UNLINK   PARAMSIZE,'DRAWRGN '
175
176
177
178     .PROC FrRgn,3
179     .REF  FrRect,NewRgn,CopyRgn,InsetRgn,DiffRgn,DrawRgn
180 ;-----
181 ;
182 ;  PROCEDURE FrRgn(rgn: RgnHandle; mode: INTEGER; pat: Pattern);
183 ;
184 ;  A6 OFFSETS OF PARAMS AFTER LINK:
185 ;
186 PARAMSIZE      .EQU      10
187 RGN             .EQU      PARAMSIZE+8-4           ;LONG, RGNHANDLE
188 MODE           .EQU      RGN-2                   ;WORD
189 PAT            .EQU      MODE-4                   ;LONG, ADDR OF PATTERN
190
191     LINK        A6,#0
192     MOVEM.L     D7/A3-A4,-(SP)                     ;SAVE REGS

```

```

193         MOVE.L   GRAFGLOBALS(A5),A4           ;POINT TO LISAGRAF GLOBALS
194         MOVE.L   THEPORT(A4),A3               ;GET CURRENT PORT
195         TST      PNVIS(A3)                     ;IS PNVIS NEG ?
196         BMI.S    DONE                          ;YES, QUIT
197 ;
198 ; special case rectangular region for speed.
199 ;
200         MOVE.L   RGN(A6),A0                     ;GET RGNHANDLE
201         MOVE.L   (A0),A0                       ;DE-REFERENCE IT
202         CMP      #10,RGNSIZE(A0)               ;IS IT RECTANGULAR ?
203         BNE.S    NOTRECT                       ;NO, CONTINUE
204         PEA      RGNBBOX(A0)                   ;YES, PUSH ADDR OF BBOX
205         JSR      FRRECT                         ;FRAME IT
206         BRA.S    DONE                          ;AND QUIT
207
208 NOTRECT CLR.L    -(SP)                         ;MAKE ROOM FOR FCN RESULT
209         JSR      NEWRGN                         ;ALLOCATE TEMPRGN
210         MOVE.L   (A7)+,D7                       ;PUT TEMPRGN IN D7
211         MOVE.L   RGN(A6),-(SP)                 ;PUSH RGN
212         MOVE.L   D7,-(SP)                     ;PUSH TEMPRGN
213         JSR      COPYRGN                       ;COPY RGN INTO TEMPRGN
214         MOVE.L   D7,-(SP)                     ;PUSH TEMPRGN
215         MOVE.L   PNSIZE(A3),-(SP)              ;PUSH PNSIZE
216         JSR      INSETRGN                      ;InsetRgn(tempRgn,pnSize);
217         MOVE.L   RGN(A6),-(SP)                 ;PUSH RGN
218         MOVE.L   D7,-(SP)                     ;PUSH TEMPRGN
219         MOVE.L   D7,-(SP)                     ;PUSH TEMPRGN
220         JSR      DIFFRGN                       ;DiffRgn(rgn,tempRgn,tempRgn)
221         MOVE.L   D7,-(SP)                     ;PUSH TEMPRGN
222         MOVE     MODE(A6),-(SP)                ;PUSH MODE
223         MOVE.L   PAT(A6),-(SP)                ;PUSH PAT
224         JSR      DRAWRGN                       ;DrawRgn(tempRgn,mode,pat);
225         MOVE.L   D7,A0                         ;GET TEMPRGN
226         _DisposHandle                          ;DISCARD IT
227 DONE    MOVEM.L (SP)+,D7/A3-A4                ;RESTORE REGS
228         UNLINK   PARAMSIZE,'FRRGN'
229
230
231
232         .FUNC NewRgn,0
233         .REF  NewHandle
234 ;-----
235 ;
236 ; FUNCTION NewRgn;
237 ; Allocate a new region and set it to the empty region.
238 ;
239         MOVEM.L D3/A2,-(SP)                    ;SAVE REGS
240         CLR.L    -(SP)                         ;MAKE ROOM FOR FCN RESULT

```



```

289      .PROC CloseRgn,1
290      .REF  ShowPen,SortPoints,CullPoints,PackRgn
291  ;-----
292  ;
293  ;  PROCEDURE CloseRgn(* dstRgn: RgnHandle *);
294  ;  Sort array of inversion points and pack into region.
295  ;
296  ;  A6 OFFSETS OF PARAMS AND LOCALS AFTER LINK:
297  ;
298  PARAMSIZE      .EQU      4                      ;TOTAL BYTES OF PARAMETERS
299  DSTRGN         .EQU      PARAMSIZE+8-4          ;LONG, RGNHANDLE
300  PTCOUNT        .EQU      -2                     ;WORD
301  VARSIZE        .EQU      PTCOUNT                ;SIZE OF LOCAL VARS
302
303
304      LINK      A6,#VARSIZE
305      MOVEM.L   D3/A2-A4,-(SP)                    ;SAVE REGS
306      MOVE.L    GRAFGLOBALS(A5),A4                ;POINT TO QUICKDRAW GLOBALS
307      MOVE.L    THEPORT(A4),A0                    ;GET CURRENT GRAFPORT
308      TST.L     RGNSAVE(A0)                        ;IS RGNSAVE TRUE ?
309      BEQ.S     DONE                               ;NO, ABORT
310      CLR.L     RGNSAVE(A0)                        ;YES, RGNSAVE := FALSE
311      JSR       SHOWPEN                            ;UNDO THE HIDEPEN FROM OPENR
312      MOVE      RGNINDEX(A4),D0                    ;GET CURRENT RGNINDEX
313      LSR       #2,D0                              ;DIV BY 4
314      MOVE      D0,PTCOUNT(A6)                    ;FOR PTCOUNT
315      MOVE.L    RGNBUF(A4),A3                      ;GET RGNBUF HANDLE
316      MOVE.L    (A3),-(SP)                          ;PUSH BUF POINTER
317      MOVE      D0,-(SP)                            ;PUSH PTCOUNT
318      JSR       SORTPOINTS                          ;QUICKSORT IN VH ORDER
319      MOVE.L    (A3),-(SP)                          ;PUSH BUF POINTER
320      PEA       PTCOUNT(A6)                          ;PUSH VAR PTCOUNT
321      JSR       CULLPOINTS                          ;CANCEL DUPLICATE PAIRS
322      MOVE.L    A3,-(SP)                            ;PUSH RGNBUF HANDLE
323      MOVE      PTCOUNT(A6),-(SP)                    ;PUSH (UPDATED) PTCOUNT
324      MOVE.L    DSTRGN(A6),-(SP)                    ;PUSH DSTRGN
325      JSR       PACKRGN                            ;PACK POINTS INTO DSTRGN
326      MOVE.L    A3,A0                              ;GET RGNBUF HANDLE
327      _DisposHandle                                ;DISCARD IT
328  DONE  MOVEM.L (SP)+,D3/A2-A4                      ;RESTORE REGS
329      UNLINK    PARAMSIZE, 'CLOSERGN'
330
331
332
333      .PROC CopyRgn,1
334      .REF  SetSize
335  ;-----
336  ;

```

```

337 ;   PROCEDURE CopyRgn(* srcRgn,dstRgn: RgnHandle *);
338 ;
339 PARAMSIZE      .EQU      8
340 SRCRGN         .EQU      PARAMSIZE+8-4           ;RGNHANDLE
341 DSTRGN         .EQU      SRCRGN-4               ;RGNHANDLE
342
343         LINK      A6,#0                          ;ESTABLISH STACK FRAME
344         MOVE.L    SRCRGN(A6),A0                  ;GET SRCRGN HANDLE
345         MOVE.L    DSTRGN(A6),A1                  ;GET DSTRGN HANDLE
346         CMP.L     A0,A1                          ;ARE THEY THE SAME?
347         BEQ.S     DONE                          ;YES, QUIT
348
349         MOVE.L    (A0),A0                        ;DE-REFERENCE SRCRGN HANDLE
350         MOVE.L    (A1),A1                        ;DE-REFERENCE DSTRGN HANDLE
351         MOVE      RGNSIZE(A0),D0                 ;GET SRC SIZE
352         CMP       RGNSIZE(A1),D0                 ;IS DST SIZE SAME AS SRC ?
353         BEQ.S     COPY                          ;YES, CONTINUE
354
355         MOVEM.L   D0/D3/A2,-(SP)                 ;SAVE REGS AND BYTECOUNT
356         MOVE.L    DSTRGN(A6),-(SP)              ;PUSH DSTRGN HANDLE
357         MOVE      D0,-(SP)                      ;PUSH NEWSIZE=SRC SIZE
358         JSR       SETSIZE                       ;CHANGE SIZE OF DST
359         MOVEM.L   (SP)+,D0/D3/A2                ;RESTORE REGS AND BYTECOUNT
360         MOVE.L    SRCRGN(A6),A0                  ;GET SRCRGN HANDLE
361         MOVE.L    DSTRGN(A6),A1                  ;GET DSTRGN HANDLE
362         MOVE.L    (A0),A0                        ;DE-REFERENCE SRCRGN HANDLE
363         MOVE.L    (A1),A1                        ;DE-REFERENCE DSTRGN HANDLE
364
365 COPY      LSR      #2,D0                        ;LONGCOUNT := BYTECOUNT DIV
366           BCC.S    EVEN                        ;WAS THERE AN ODD WORD ?
367           MOVE     (A0)+,(A1)+                  ;YES, DO IT TO MAKE EVEN
368           BRA.S    EVEN                        ;AND CONTINUE
369 COPYLP    MOVE.L   (A0)+,(A1)+                  ;COPY A LONG OF SRC TO DST
370 EVEN      DBRA     D0,COPYLP                    ;LOOP FOR ALL LONGS
371
372 DONE      UNLINK   PARAMSIZE,'COPYRGN '
373
374
375
376         .PROC SetEmptyRgn,1
377         .REF  SetRectRgn
378 ;-----
379 ;
380 ;   PROCEDURE SetEmptyRgn(rgn: RgnHandle);
381 ;
382         MOVE.L    (SP)+,A0                      ;POP RETURN ADDR
383         CLR.L     -(SP)                         ;PUSH LEFT, TOP
384         CLR.L     -(SP)                         ;PUSH RIGHT,BOTTOM

```



```

385         MOVE.L  A0,-(SP)                ;RESTORE RETURN ADDR
386         JMP      SETRECTRGN             ;DOUBLE UP ON CODE
387
388
389
390         .PROC  SetRectRgn,5
391         .REF   SetSize
392 ;-----
393 ;
394 ;  PROCEDURE SetRectRgn(rgn: RgnHandle; left,top,right,bottom: INTEGER);
395 ;  make a rectangular region from 4 integers.
396 ;
397         LINK      A6,#0                  ;ESTABLISH STACK FRAME
398         MOVEM.L   D3/A2,-(SP)             ;SAVE REGS
399         MOVE.L    16(A6),A0              ;GET RGN HANDLE
400         MOVE.L    (A0),A1                ;DE-REFERENCE HANDLE
401         MOVEQ     #10,D0
402         CMP       RGNSIZE(A1),D0         ;IS RGNSIZE ALREADY 10 ?
403         BEQ.S     SIZEOK                 ;YES, CONTINUE
404         MOVE.L    A0,-(SP)               ;PUSH RGNHANDLE
405         MOVE      D0,-(SP)               ;PUSH SIZE = 10 BYTES
406         JSR       SETSIZE                 ;CHANGE SIZE OF REGION
407         MOVE.L    16(A6),A0              ;GET RGN HANDLE
408         MOVE.L    (A0),A1                ;DE-REFERENCE HANDLE
409         MOVE      #10,RGNSIZE(A1)        ;INSTALL SIZE = 10
410 SIZEOK  MOVE.L    12(A6),RGNBBOX+TOPLEFT(A1) ;INSTALL RGNBBOX TOPLEFT
411         MOVE.L    8(A6),RGNBBOX+BOTRIGHT(A1) ;INSTALL RGNBBOX BOTRIGHT
412         MOVE      RGNBBOX+LEFT(A1),D0
413         CMP       RGNBBOX+RIGHT(A1),D0   ;IS LEFT >= RIGHT ?
414         BGE.S     EMPTY                  ;YES, SET TO EMPTY
415
416         MOVE      RGNBBOX+TOP(A1),D0
417         CMP       RGNBBOX+BOTTOM(A1),D0   ;IS TOP < BOTTOM ?
418         BLT.S     DONE                   ;YES, CONTINUE
419
420 EMPTY   CLR.L     RGNBBOX+TOPLEFT(A1)
421         CLR.L     RGNBBOX+BOTRIGHT(A1)
422 DONE    MOVEM.L   (SP)+,D3/A2            ;RESTORE REGS
423         UNLINK    12,'SETRECTR'
424
425
426
427         .PROC  RectRgn,2
428         .REF   SetRectRgn
429 ;-----
430 ;
431 ;  PROCEDURE RectRgn(rgn: RgnHandle; r: Rect);
432 ;  make a rectangular region from a rectangle

```

```

433 ;
434     MOVE.L    (SP)+,A0                ;POP RETURN ADDR
435     MOVE.L    (SP)+,A1                ;POP ADDR OF RECT
436     MOVE.L    (A1)+,-(SP)            ;PUSH LEFT,TOP
437     MOVE.L    (A1)+,-(SP)            ;PUSH RIGHT,BOTTOM
438     MOVE.L    A0,-(SP)                ;RESTORE RETURN ADDR
439     JMP        SETRECTRGN            ;DOUBLE UP ON CODE
440
441
442
443     .PROC OffsetRgn,3
444 ;-----
445 ;
446 ;   PROCEDURE OffsetRgn(rgn: RgnHandle; dh,dv: INTEGER);
447 ;
448     MOVE.L    (SP)+,A1                ;POP RETURN ADDR
449     MOVE      (SP)+,D1                ;POP DV
450     MOVE      (SP)+,D0                ;POP DH
451     MOVE.L    (SP)+,A0                ;POP RGN HANDLE
452     MOVE.L    (A0),A0                ;DE-REFERENCE IT
453     ADD       #2,A0                  ;POINT TO RGNBBBOX
454
455 ;-----
456 ;
457 ;   OFFSET THE BOUNDING BOX
458 ;
459     ADD       D1,(A0)+                ;ADD DV TO TOP
460     ADD       D0,(A0)+                ;ADD DH TO LEFT
461     ADD       D1,(A0)+                ;ADD DV TO BOTTOM
462     ADD       D0,(A0)+                ;ADD DH TO RIGHT
463
464
465 ;-----
466 ;
467 ;   IF NON-RECTANGULAR REGION, OFFSET THE COORDINATES TOO.
468 ;
469     CMP       #10,-10(A0)            ;IS REGION RECTANGULAR ?
470     BEQ.S     DONE                   ;IF SO, WE'RE ALL DONE
471 NXTVERT     ADD       D1,(A0)+        ;OFFSET VERTICAL COORD DV
472 NXTHOR      ADD       D0,(A0)+        ;OFFSET LEFT COORD DH
473            ADD       D0,(A0)+        ;OFFSET RIGHT COORD DH
474            CMP       #32767,(A0)      ;HORIZ TERMINATOR ?
475            BNE       NXTHOR           ;NO, LOOP
476            ADD       #2,A0            ;YES, SKIP OVER TERMINATOR
477            CMP       #32767,(A0)      ;IS NEXT VERT = 32767 ?
478            BNE       NXTVERT         ;NO, LOOP FOR MORE
479 DONE        JMP       (A1)            ;RETURN
480

```

```

481
482
483     .PROC InsetRgn,3
484     .REF InsetRect,SortPoints
485     .REF NewHandle,RgnOp,PackRgn
486 ;-----
487 ;
488 ; PROCEDURE InsetRgn(rgn: RgnHandle; dh,dv: INTEGER);
489 ;
490 ; Inset a region by (dh,dv). Outset if dh or dv neg
491 ;
492 ; A6 OFFSETS OF PARAMS AFTER LINK:
493 ;
494 PARAMSIZE      .EQU      8
495 RGN             .EQU      PARAMSIZE+8-4           ;LONG, RGNHANDLE
496 DH             .EQU      RGN-2                   ;WORD
497 DV             .EQU      DH-2                     ;WORD
498
499
500 LINK           A6,#0                               ;NO LOCAL VARS
501 MOVEM.L D3-D7/A2-A4,-(SP)                          ;SAVE REGS
502 MOVE.L DV(A6),D6                                     ;GET DV AND DH BOTH
503 BEQ.S JGOHOME                                       ;QUIT IF BOTH ARE ZERO
504 MOVE.L RGN(A6),A4                                   ;GET RGNHANDLE
505 MOVE.L (A4),A3                                       ;DE-REFERENCE IT
506 MOVE (A3)+,D7                                       ;GET RGNSIZE
507
508
509 ;-----
510 ;
511 ; IF RECTANGULAR REGION, CALL INSETRECT AND CHECK FOR EMPTY.
512 ;
513 CMP #10,D7                                           ;IS RGN RECTANGULAR ?
514 BNE.S NOTRECT                                       ;NO, CONTINUE
515 MOVE.L A3,-(SP)                                     ;PUSH ADDR OF RGNBBOX
516 MOVE.L D6,-(SP)                                     ;PUSH DH AND DV
517 JSR INSETRECT                                       ;InsetRect(rgn^^.rgnbbox,dh,
518 MOVE LEFT(A3),D0                                     ;GET BBOX LEFT
519 CMP RIGHT(A3),D0                                    ;IS LEFT >= RIGHT ?
520 BGE.S EMPTY                                         ;YES, RETURN EMPTY RGN
521 MOVE TOP(A3),D0                                     ;GET BBOX TOP
522 CMP BOTTOM(A3),D0                                   ;IS TOP >= BOTTOM ?
523 BLT.S JGOHOME                                       ;NO, CONTINUE
524 EMPTY CLR.L (A3)+                                   ;SET RGNBBOX TO (0,0,0,0) TO
525 CLR.L (A3)+                                         ;RETURN EMPTY REGION
526 JGOHOME BRA.S GOHOME                               ;ALL DONE
527
528

```

```

529 ;-----
530 ;
531 ; THE REGION IS NOT RECTANGULAR. ALLOCATE A POINT BUFFER IN THE HEAP.
532 ;
533 NOTRECT ADD      D7,D7                      ;TRY BYTES NEEDED = RGNSIZE*
534          CLR.L   -(SP)                     ;ROOM FOR FCN RESULT
535          MOVE    D7,-(SP)                   ;PUSH BYTESNEEDED
536          JSR     NEWHANDLE                   ;ALLOCATE PTBUF
537          MOVE.L  (SP)+,A3                    ;PUT PTBUF HANDLE IN A3
538          BSR.S   HINSET                      ;INSET HORIZ AND FLIP
539          SWAP    D6                          ;GET DV INSTEAD OF DH
540          BSR.S   HINSET                      ;INSET VERTICAL AND FLIP BAC
541          BRA.S   DONE                        ;ALL DONE
542
543
544 ;-----
545 ;
546 ; LOCAL ROUTINE TO INSET HORIZONTALLY, SWAP H AND V COORDS, AND RE-SORT
547 ;
548 ; RgnOp(rgn,rgn,bufHandle,maxBytes,op,dh,TRUE);
549 ;
550 HINSET CLR.W      -(SP)                      ;ROOM FOR FCN RESULT
551          MOVE.L   A4,-(SP)                   ;PUSH RGNA = USER RGN
552          MOVE.L   A4,-(SP)                   ;PUSG RGNB = USER RGN ALSO
553          MOVE.L   A3,-(SP)                   ;PUSH BUFHANDLE
554          MOVE     D7,-(SP)                   ;PUSH MAXBYTES
555          MOVE     #8,-(SP)                   ;PUSH OP = INSET
556          MOVE     D6,-(SP)                   ;PUSH DH
557          ST       -(SP)                      ;PUSH OKGROW = TRUE
558          JSR      RGNOP
559          MOVE.W   (SP)+,D5                    ;GET PTCOUNT IN D5
560 ;
561 ; SWAP VERT AND HORIZ COORDS OF ALL INVERSION POINTS
562 ;
563          MOVE.L   (A3),A0                     ;DE-REFERENCE PTBUF HANDLE
564          MOVE     D5,D1                       ;COPY PTCOUNT
565          BRA.S    SWAP2                       ;GO TO LOOP START
566 SWAPLP  MOVE.L   (A0),D0                     ;GET A POINT
567          SWAP     D0                          ;SWAP ITS COORDINATES
568          MOVE.L   D0,(A0)+                    ;PUT IT BACK INTO ARRAY
569 SWAP2   DBRA     D1,SWAPLP                   ;LOOP FOR ALL POINTS
570 ;
571 ; RE-SORT THE POINTS IN V.H ORDER
572 ;
573          MOVE.L   (A3),-(SP)                   ;PUSH PTBUF PTR
574          MOVE     D5,-(SP)                   ;PUSH PTCOUNT
575          JSR      SORTPOINTS                   ;RE-SORT INTO VH ORDER
576 ;

```

```

577 ;   PACK THE RESULTING POINTS
578 ;
579     MOVE.L   A3,-(SP)                ;PUSH PTBUF HANDLE
580     MOVE     D5,-(SP)                ;PUSH PTCOUNT
581     MOVE.L   A4,-(SP)                ;PUSH RGN HANDLE
582     JSR      PACKRGN                 ;PackRgn(ptBuf,ptCount,rgn);
583     RTS                                           ;RETURN TO LOCAL CALLER
584
585
586 ;-----
587 ;
588 ;   DISCARD POINT BUFFER AND QUIT.
589 ;
590 DONE    MOVE.L   A3,A0                ;GET PTBUF HANDLE
591         _DisposHandle                ;DISCARD IT
592
593 GOHOME  MOVEM.L  (SP)+,D3-D7/A2-A4    ;RESTORE REGS
594         UNLINK   PARAMSIZE,'INSETRGN'
595
596
597
598     .FUNC EmptyRgn,1
599     .REF  EmptyRect
600 ;-----
601 ;
602 ;   FUNCTION EmptyRgn(rgn: RgnHandle): BOOLEAN;
603 ;
604     MOVE.L   (SP)+,A0                ;POP RETURN ADDR
605     MOVE.L   (SP)+,A1                ;POP RGNHANDLE
606     MOVE.L   (A1),A1                ;DE-REFERENCE HANDLE
607     PEA      RGNBBOX(A1)            ;PUSH RGNBBOX
608     MOVE.L   A0,-(SP)                ;PUSH RETURN ADDR
609     JMP      EMPTYRECT              ;USE EMPTYRECT CODE
610
611
612
613     .FUNC EqualRgn,2
614 ;-----
615 ;
616 ;   FUNCTION EqualRgn(rgnA,rgnB: RgnHandle): BOOLEAN;
617 ;
618 ;   RETURNS TRUE IF RGNA AND RGNB DESCRIBE THE SAME AREA
619 ;
620     MOVE.L   4(SP),A0                ;GET RGNA HANDLE
621     MOVE.L   8(SP),A1                ;GET RGNB HANDLE
622     CMP.L    A0,A1                  ;ARE THEY THE SAME ?
623     BEQ.S    TRUE                    ;YES, THE REGIONS ARE THE SAME
624     MOVE.L   (A0),A0                ;DE-REFERENCE RGN A

```

```

625         MOVE.L   (A1),A1                ;DE-REFERENCE RGN B
626         MOVE     (A0),D0                ;GET RGNSIZE A
627         MOVE     D0,D1                   ;MAKE AN EXTRA COPY
628         LSR       #2,D1                   ;DIV BY 4 FOR LONG COUNT
629         SUB       #1,D1                   ;INIT DBRA LOOP COUNT
630 LOOP     CMPM.L   (A0)+,(A1)+             ;COMPARE A LONG
631         DBNE      D1,LOOP                 ;LOOK TILL DIFFERENT OR COUNT
632         BNE.S     FALSE                   ;DIFFERENT --> FALSE
633         AND        #3,D0                   ;GET 0..3 FINISH UP BYTES
634         BEQ.S     TRUE                     ;IF NO MORE, WE MADE IT
635 LOOP2    CMPM.B   (A0)+,(A1)+             ;COMPARE A BYTE
636         BNE.S     FALSE                   ;BR IF DIFFERENT
637         SUB       #1,D0
638         BNE.S     LOOP2                   ;LOOP LAST 1..3 BYTES
639 TRUE     MOVE.L   #1,D0                   ;TRUE
640         BRA.S     DONE
641 FALSE    MOVE.L   #0,D0                   ;FALSE
642 DONE     MOVE.B   D0,#12(SP)              ;SET RESULT
643         MOVE.L   (SP)+,A0                 ;POP RETURN ADDR
644         ADD      #8,SP                     ;STRIP PARAMETERS
645         JMP      (A0)                     ;RETURN

```

```

646
647
648
649         .PROC   SectRgn,3
650         .DEF    DoRgnOp,UnionRgn,DiffRgn,XorRgn
651         .REF    EqualRgn,CopyRgn,Rsect,RectRgn,SetEmptyRgn
652         .REF    NewHandle,RgnOp,PackRgn
653 ;-----
654 ;
655 ;   PROCEDURE SectRgn(srcRgnA,srcRgnB,dstRgn: RgnHandle);
656 ;   calculate the intersection of two regions.
657 ;
658         MOVEQ     #0,D0                    ;OP = SECT
659         BRA.S     DoRgnOp                  ;SHARE COMMON CODE
660
661
662

```

```

663 ;-----
664 ;
665 ;   PROCEDURE UnionRgn(srcRgnA,srcRgnB,dstRgn: RgnHandle);
666 ;   calculate the union of two regions.
667 ;
668 UnionRgn
669         MOVEQ     #4,D0                    ;OP = UNION
670         BRA.S     DoRgnOp                  ;SHARE COMMON CODE
671
672

```

```

673
674 ;-----
675 ;
676 ; PROCEDURE DiffRgn(srcRgnA,srcRgnB,dstRgn: RgnHandle);
677 ; calculate the difference A-B of two regions
678 ;
679 DiffRgn MOVEQ    #2,D0                      ;OP = DIFF
680          BRA.S    DoRgnOp                    ;SHARE COMMON CODE
681
682
683
684 ;-----
685 ;
686 ; PROCEDURE XorRgn(srcRgnA,srcRgnB,dstRgn: RgnHandle);
687 ; calculate the exclusive or of two regions
688 ;
689 XorRgn  MOVEQ    #6,D0                      ;OP = DIFF
690          BRA.S    DoRgnOp                    ;SHARE COMMON CODE
691
692
693
694 ;-----
695 ;
696 ; PROCEDURE DoRgnOp(srcRgnA,srcRgnB,dstRgn: RgnHandle);
697 ;
698 ; Computes the Intersection, Difference, Union, or Xor of two regions.
699 ;
700 ; enter with op in D0.
701 ; op = 0: SECT
702 ;       2: DIFF A-B
703 ;       4: UNION
704 ;       6: XOR
705 ;
706 ;
707 ; A6 OFFSETS OF PARAMS AND LOCALS AFTER LINK:
708 ;
709 PARAMSIZE      .EQU    12
710 RGNA           .EQU    PARAMSIZE+8-4        ;LONG, RGNHANDLE
711 RGNB           .EQU    RGNA-4                ;LONG, RGNHANDLE
712 DSTRGN         .EQU    RGNB-4                ;LONG, RGNHANDLE
713 ;
714 TEMPRECT       .EQU    -8                    ;RECT
715 VARSIZE        .EQU    TEMPRECT              ;TOTAL LOCALS
716 ;
717 DoRgnOp
718     LINK        A6,#VARSIZE                  ;ALLOCATE STACK FRAME
719     MOVEM.L     D3-D7/A2-A4,-(SP)             ;SAVE REGS
720     MOVE        D0,D5                        ;COPY OP INTO D5

```

```

721         MOVE.L   RGNA(A6),A2                ;GET RGNA
722         MOVE.L   RGNB(A6),A3                ;GET RGNB
723         MOVE.L   DSTRGN(A6),A4              ;GET DSTRGN
724         MOVE.L   #2,D7
725 ;
726 ;   ARE THE TWO INPUT REGIONS THE SAME ?
727 ;
728         CLR.B     -(SP)                      ;MAKE ROOM FOR FCN RESULT
729         MOVE.L   A2,-(SP)                    ;PUSH RGNA
730         MOVE.L   A3,-(SP)                    ;PUSH RGNB
731         JSR      EQUALRGN                    ;CALL EQUALRGN
732         TST.B    (SP)+                       ;ARE THEY THE SAME ?
733         BEQ.S    NOTSAME                     ;NO, CONTINUE
734 ;
735 ;   THE TWO REGIONS ARE THE SAME.  IF SECT OR UNION
736 ;   THEN COPY RGNA INTO DSTRGN, ELSE ZERO OUT DSTRGN.
737 ;
738         AND      D7,D5                      ;WAS OP SECT OR UNION ?
739         BNE.S    ZERO                        ;NO, ZERO OUT DSTRGN
740 COPY    MOVE.L   A2,-(SP)                    ;PUSH RGNA
741         MOVE.L   A4,-(SP)                    ;PUSH DSTRGN
742         JSR      COPYRGN                     ;COPY RGNA INTO DSTRGN
743         BRA.S    JDONE                       ;AND QUIT
744 ZERO    MOVE.L   A4,-(SP)                    ;PUSH DSTRGN
745         JSR      SetEmptyRgn                 ;SET IT TO EMPTY
746         BRA.S    JDONE                       ;AND QUIT
747 ;
748 ;
749 ;   IF OP = DIFF AND RGNB = EMPTY,  COPY RGNA INTO DST
750 ;
751 NOTSAME MOVE.L   (A2),A0                     ;DE-REFERENCE RGNA
752         MOVE.L   (A3),A1                     ;DE-REFERENCE RGNB
753         CMP      D7,D5                       ;IS OP = DIFF ?
754         BGT.S    UNIXOR                      ;NO, ITS UNION OR XOR
755         BLT.S    BBOXES                      ;NO, IT'S SECT
756         MOVE     RGNBBOX+LEFT(A1),D0         ;GET BBOX LEFT
757         CMP      RGNBBOX+RIGHT(A1),D0       ;IS BBOX LEFT >= RIGHT ?
758         BGE      COPY                        ;YES, COPY RGNA INTO DST
759 ;
760 ;
761 ;   IF op = SECT OR DIFF, THEN INTERSECT THE BOUNDING BOXES.
762 ;
763 BBOXES  PEA      RGNBBOX(A0)                 ;PUSH RGNA^.RGNBBOX
764         PEA      RGNBBOX(A1)                 ;PUSH RGNB^.RGNBBOX
765         MOVE     D7,-(SP)                     ;PUSH NRECTS = 2
766         PEA      TEMPRECT(A6)                ;PUSH DST = TEMPRECT
767         JSR      RSECT                       ;CALC INTERSECTION
768         BNE.S    NOTEMPTY                    ;BR IF RESULT NOT EMPTY

```



```

769 ;
770 ; THE BOUNDING BOXES DON'T INTERSECT.
771 ; IF OP = SECT, THEN RETURN EMPTY.
772 ; IF OP = DIFF, THEN COPY RGNA INTO DSTRGN.
773 ;
774         TST      D5                      ;IS OP = SECT ?
775         BEQ      ZERO                    ;YES, RETURN EMPTY
776         BRA      COPY                    ;NO, COPY SRCA INTO DSTRGN
777
778 ;
779 ; IF OP = SECT, THEN CHECK FOR BOTH INPUTS RECTANGULAR
780 ;
781 NOTEMPTY MOVE.L (A2),A0                  ;DE-REFERENCE RGNA
782         MOVE.L (A3),A1                  ;DE-REFERENCE RGNB
783         TST      D5                      ;IS OP = SECT ?
784         BNE.S    NOTEASY                 ;NO, CONTINUE
785         MOVEQ    #10,D0
786         CMP      RGNSIZE(A0),D0          ;IS RGNA RECTANGULAR ?
787         BNE.S    NOTEASY                 ;NO, CONTINUE
788         CMP      RGNSIZE(A1),D0          ;IS RGNB RECTANGULAR ?
789         BNE.S    NOTEASY                 ;NO, CONTINUE
790         MOVE.L   A4,-(SP)                 ;PUSH DSTRGN
791         PEA      TEMPRECT(A6)             ;PUSH TEMPRECT
792         JSR      RECTRGN                  ;RectRgn(dstRgn,tempRect);
793 JDONE    BRA.S   DONE
794
795 ;
796 ; OP = UNION OR XOR: IF EITHER REGION IS EMPTY, COPY THE OTHER.
797 ;
798 UNIXOR   MOVE    RGNBBOX+LEFT(A1),D0      ;GET RGNB BBOX LEFT
799         CMP      RGNBBOX+RIGHT(A1),D0    ;IS RGNB BBOX LEFT >= RIGHT
800         BGE      COPY                    ;YES, COPY RGNA INTO DST
801
802         MOVE     RGNBBOX+LEFT(A0),D0      ;GET RGNA BBOX LEFT
803         CMP      RGNBBOX+RIGHT(A0),D0    ;IS RGNA BBOX LEFT >= RIGHT
804         BLT.S    NOTEASY                 ;NO, CONTINUE
805         MOVE.L   A3,A2                    ;YES, GET RGNB INSTEAD
806         BRA      COPY                    ;COPY RGNB INTO DST
807
808
809 NOTEASY  MOVE     RGNSIZE(A0),D4           ;GET RGNA RGNSIZE
810         ADD      RGNSIZE(A1),D4           ;ADD RGNB RGNSIZE
811         ADD      D4,D4                     ;TRY DOUBLE FOR BYTECOUNT
812         CLR.L    -(SP)                     ;MAKE ROOM FOR FCN RESULT
813         MOVE     D4,-(SP)                 ;PUSH BYTECOUNT
814         JSR      NEWHANDLE                 ;ALLOCATE PTBUF
815         MOVE.L   (SP)+,A3                 ;GET PTBUF HANDLE IN A3
816 ;

```



```

865 ;-----
866 ;
867 ; FIRST CHECK BOUNDING BOX
868 ;
869         MOVE.L   RGN(A6),A0                ;GET RGN HANDLE
870         MOVE.L   (A0),A0                  ;DE-REFERENCE IT
871         CMP      RGNBBOX+LEFT(A0),D1       ;IS PT.H < BBOX LEFT ?
872         BLT.S    DONE                     ;YES, RETURN FALSE
873         CMP      RGNBBOX+RIGHT(A0),D1      ;IS PT.H >= BBOX RIGHT ?
874         BGE.S    DONE                     ;YES, RETURN FALSE
875         CMP      RGNBBOX+TOP(A0),D2        ;IS PT.V < BBOX TOP ?
876         BLT.S    DONE                     ;YES, RETURN FALSE
877         CMP      RGNBBOX+BOTTOM(A0),D2     ;IS PT.V >= BBOX BOT ?
878         BGE.S    DONE                     ;YES, RETURN FALSE
879         CMP      #10,RGNSIZE(A0)           ;IS REGION RECTANGULAR ?
880         BNE.S    NOTRECT                   ;NO, CONTINUE
881         NOT      D3                        ;YES, RETURN TRUE
882         BRA.S    DONE
883
884
885 ;-----
886 ;
887 ; PT IS INSIDE BOUNDING BOX AND REGION IS NOT RECTANGULAR.
888 ; LOOK AT THE INVERSION POINTS TO DETERMINE IF PT IN REGION.
889 ;
890 NOTRECT LEA      RGNDATA(A0),A0            ;POINT TO FIRST VERT COORD
891 NXTVERT CMP      (A0)+,D2                  ;IS NEXT VERT > PT.V ?
892         BLT.S    DONE                     ;YES, QUIT
893 NEXTHOR MOVE     (A0)+,D0                  ;GET HORIZ COORD
894         CMP      #32767,D0                ;IS IT THE TERMINATOR ?
895         BEQ      NXTVERT                   ;YES, GET NEXT VERT COORD
896         CMP      D1,D0                    ;IS HORIZ <= PT.H ?
897         BGT      NEXTHOR                   ;NO, IGNORE THIS POINT
898         NOT      D3                        ;YES, TOGGLE INSIDE
899         BRA      NEXTHOR                   ;AND GO FOR MORE POINTS
900 DONE  NEG.B      D3                        ;BOOLEAN RESULT IS 0 OR 1
901         MOVE.B   D3,RESULT(A6)             ;RETURN BOOLEAN FCN RESULT
902         MOVE.L   (SP)+,D3                  ;RESTORE REG
903         UNLINK   PARAMSIZE,'PTINRGN '
904
905
906
907
908         .FUNC RectInRgn,2
909         .REF  RSect,InitRgn,SeekRgn
910 ;-----
911 ;
912 ; FUNCTION RectInRgn(r: Rect; rgn: RgnHandle): BOOLEAN;

```

```

913 ;
914 ; Returns TRUE if any part of the rectangle intersects the region.
915 ;
916 ;
917 ; A6 OFFSETS OF PARAMETERS AFTER LINK:
918 ;
919 PARAMSIZE      .EQU      8                ;TOTAL SIZE OF PARAMETERS
920 RESULT         .EQU      PARAMSIZE+8      ;BYTE, BOOLEAN
921 RECT           .EQU      RESULT-4         ;LONG, VAR ADDR
922 RGN            .EQU      RECT-4           ;LONG, RGNHANDLE
923
924
925 ;-----
926 ;
927 ; A6 OFFSETS OF LOCAL VARIABLES AFTER LINK:
928 ;
929 MINRECT        .EQU      -8               ;RECTANGLE
930 SAVESTACK      .EQU      MINRECT-4        ;LONG
931 STATE          .EQU      SAVESTACK-RGNREC ;REGION STATE RECORD
932 VARSIZE        .EQU      STATE           ;TOTAL SIZE OF VARIABLES
933
934
935 LINK           A6,#VARSIZE                ;ALLOCATE LOCAL VARIABLES
936 MOVEM.L D0-D7/A1-A5,-(SP)                ;SAVE REGISTERS
937 MOVE.L SP,SAVESTACK(A6)                  ;REMEMBER STACK START
938 CLR.B RESULT(A6)                         ;INIT BOOLEAN RESULT TO FALSE
939 MOVE.L RGN(A6),A1                         ;GET REGION HANDLE
940 MOVE.L (A1),A1                            ;DE-REFERENCE IT
941
942
943 ;-----
944 ;
945 ; FIRST CHECK IF RECTANGLE INTERSECTS BOUNDING BOX OF REGION
946 ;
947 MOVE.L RECT(A6),-(SP)                    ;PUSH POINTER TO RECT
948 PEA RGNBBOX(A1)                          ;PUSH POINTER TO RGN BBOX
949 MOVE #2,-(SP)                            ;PUSH NRECTS=2
950 PEA MINRECT(A6)                          ;PUSH ADDR WHERE TO PUT RESULT
951 JSR RSECT                                ;CALC INTERSECTION
952 BEQ.S GOHOME                             ;QUIT IF NO INTERSECTION
953 CMP #10,RGNSIZE(A1)                      ;IS REGION RECTANGULAR ?
954 BEQ.S TRUE                                ;YES, RETURN TRUE
955
956
957 ;-----
958 ;
959 ; THE REGION IS NON-RECTANGULAR AND THE RECTANGLE INTERSECTS
960 ; THE REGION'S BOUNDING BOX. WE WILL PLAY BACK THE PORTION OF

```

```

961 ; THE REGION WITHIN MINRECT AND SEE IF ANY PART OF THE REGION
962 ; IS ACTUALLY INSIDE THE RECTANGLE.
963 ;
964
965
966 ;-----
967 ;
968 ; INITIALIZE RGN STATE RECORD AT TOP.
969 ;
970         MOVE.L   A1,A0                      ;GET RGNPTR IN A0
971         LEA      STATE(A6),A1                ;GET STATE RECORD IN A1
972         MOVE     MINRECT+LEFT(A6),D0         ;MINH IN D0
973         MOVE     MINRECT+RIGHT(A6),D1        ;MAXH IN D1
974         MOVE     D0,D2                      ;BUFLEFT:=MINH
975         JSR      INITRGN                    ;INIT RECORD, ALLOCATE BUFFE
976
977
978 ;-----
979 ;
980 ; PLAY THE REGION BACK INTO SCAN BUFFER UNTIL IT GETS DOWN TO
981 ; MINRECT, THEN CHECK EACH SCANLINE FOR NON-ZERO PLAYBACK.
982 ; QUIT AND RETURN TRUE IF NON-ZERO FOUND BEFORE RGN GOES BEYOND MINRECT.
983 ;
984         MOVE     SCANSIZE(A1),D5              ;GET BUFSIZE= # LONGS -1
985         MOVE     MINRECT+TOP(A6),D0
986         JSR      SEEKRGN                     ;SEEK THE REGION TO MINRECT
987         MOVE     MINRECT+BOTTOM(A6),D6
988 TESTBUF MOVE.L   SCANBUF(A1),A0              ;POINT TO BUFFER START
989         MOVE     D5,D0                      ;INIT LOOP COUNT TO BUFSIZE
990 NXTLONG TST.L    (A0)+                      ;IS SCAN BUF NON-ZERO ?
991         DBNE     D0,NXTLONG                 ;TEST LONGS TILL NON ZERO OR
992         BNE.S    TRUE                      ;WE FOUND A NON-ZERO, RETURN
993         MOVE     NEXTV(A1),D0               ;GET NEXT VERTICAL IN RGN
994         CMP      D0,D6                      ;IS NEXT RGN VERT BEYOND BOT
995         BLE.S    GOHOME                    ;YES, RETURN FALSE
996         JSR      SEEKRGN                    ;NO, SEEK TO NEXT VERT CHANG
997         BRA.S    TESTBUF                    ;AND SEE IF IT IS NON-ZERO
998
999 TRUE     ADDQ.B   #1,RESULT(A6)              ;SET BOOLEAN RESULT TO TRUE
1000 GOHOME  MOVE.L   SAVESTACK(A6),SP           ;STRIP SCAN BUFFER IF ANY
1001         MOVEM.L  (SP)+,D0-D7/A1-A5         ;RESTORE REGISTERS
1002         UNLINK   PARAMSIZE,'RECTINRG'
1003
1004
1005
1006         .FUNC TrimRect,2
1007         .REF  RgnOp
1008 ;-----

```

```

1009 ;
1010 ; FUNCTION TrimRect(rgn: RgnHandle; VAR dstRect: Rect): CCR TRISTATE;
1011 ;
1012 ; RESULT IN CONDITION CODES:
1013 ;
1014 ; = RESULT RECTANGULAR, DSTRECT TRIMMED
1015 ; < RESULT EMPTY, DSTRECT NOT MODIFIED
1016 ; > RESULT NON-RECT, DSTRECT NOT MODIFIED
1017 ;
1018 ; If the intersection of rgn and dstRect is rectangular,
1019 ; then return EQUAL and put the intersection into dstRect.
1020 ; If the intersection is empty or not rectangular, then
1021 ; return FALSE and don't modify dstRect.
1022 ;
1023 ; Does not call the storage allocator.
1024 ;
1025 ; 1. Fake up a rect rgn on the stack from dstRect
1026 ; 2. Call RgnOp with max bytes = 24, OKGROW = FALSE
1027 ; 3a. If ptCount = 4 THEN result rect, return TRUE and update dstRect.
1028 ; 3b. If ptCount < 4 THEN result empty, return TRUE and clear dstRect.
1029 ; 3c. If ptCount > 4 THEN result not rect, return FALSE
1030 ;
1031 PARAMSIZE      .EQU      8
1032 RGN             .EQU      PARAMSIZE+8-4           ;LONG, RGNHANDLE
1033 DSTRECT        .EQU      RGN-4                   ;LONG, ADDR OF DSTRECT
1034
1035 PTDATA         .EQU      -24                     ;ROOM FOR 6 POINTS
1036 PTMASTER      .EQU      PTDATA-4                ;LONG, FAKE MASTER
1037 REGION         .EQU      PTMASTER-10            ;ROOM FOR RECT RGN DATA
1038 RGNMASTER     .EQU      REGION-4                ;LONG
1039 VARSIZE       .EQU      RGNMASTER
1040
1041 LINK          A6,#VARSIZE                        ;ALLOCATE STACK FRAME
1042 MOVEM.L       D0-D2/A1,-(SP)                     ;SAVE ALL REGS USED
1043 LEA           PTDATA(A6),A0                      ;POINT TO BUFFER
1044 MOVE.L        A0,PTMASTER(A6)                   ;INSTALL INTO FAKE MASTER
1045 LEA           REGION(A6),A1                      ;POINT TO REGION DATA
1046 MOVE.L        A1,RGNMASTER(A6)                  ;INSTALL INTO FAKE MASTER
1047 MOVE          #10,(A1)+                          ;INSTALL RGNSIZE = 10
1048 MOVE.L        DSTRECT(A6),A0                    ;POINT TO DSTRECT
1049 MOVE.L        (A0)+,(A1)+                        ;COPY DSTRECT TOPLEFT
1050 MOVE.L        (A0)+,(A1)+                        ;COPY DSTRECT BOTRIGHT
1051 ;
1052 ; RgnOp(rgn,rectRgn,BufHandle,24,sect,0,FALSE)
1053 ;
1054 CLR.W         -(SP)                              ;ROOM FOR FCN RESULT
1055 MOVE.L        RGN(A6),-(SP)                      ;PUSH RGN
1056 PEA          RGNMASTER(A6)                      ;PUSH FAKE RGNHANDLE

```

```

1057      PEA      PTMASTER(A6)                ;PUSH BUFHANDLE
1058      MOVE     #24,-(SP)                    ;PUSH MAXBYTES = 24
1059      CLR.L     -(SP)                        ;PUSH OP = SECT, DH = 0
1060      CLR.B     -(SP)                        ;PUSH OKGROW = FALSE
1061      JSR       RGNOP                        ;RgnOp(rgn,rectRgn,buf,64,op
1062      MOVE.L    DSTRECT(A6),A0               ;POINT TO DSTRECT
1063      CMP       #4,(SP)+                    ;IS PT COUNT = 4 ?
1064      BNE.S     DONE                        ;NO, RETURN < OR > IN CCR
1065      MOVE.L    PTDATA(A6),(A0)+            ;UPDATE DSTRECT.TOPLEFT
1066      MOVE.L    PTDATA+12(A6),(A0)+        ;UPDATE DSTRECT.BOTRIGHT
1067      SUB       D0,D0                        ;SET EQUAL FLAG
1068 DONE    MOVEM.L (SP)+,D0-D2/A1             ;RESTORE ALL REGS
1069      UNLK      A6                          ;RELEASE STACK FRAME
1070      MOVE.L    (SP)+,A0                    ;POP RETURN ADDR INTO A0
1071      ADD       #PARAMSIZE,SP               ;STRIP PARAMETERS
1072      JMP       (A0)                        ;JUMP THRU A0 TO RETURN
1073
1074
1075
1076      .PROC MapRgn,3
1077      .REF MapRect,NewHandle,PutRgn,MapPt
1078      .REF SortPoints,CullPoints,PackRgn
1079 ;-----
1080 ;
1081 ;  PROCEDURE MapRgn(rgn: RgnHandle; fromRect,toRect: Rect);
1082 ;
1083 ;  A6 OFFSETS OF PARAMETERS AND LOCALS AFTER LINK:
1084 ;
1085 PARAMSIZE      .EQU      12
1086 RGN             .EQU      PARAMSIZE+8-4      ;LONG, RGNHANDLE
1087 FROMRECT       .EQU      RGN-4              ;LONG, ADDR OF RECT
1088 TORECT         .EQU      FROMRECT-4         ;LONG, ADDR OF RECT
1089
1090 INDEX          .EQU      -2                  ;INTEGER
1091 SIZE           .EQU      INDEX-2            ;WORD
1092 PTCOUNT        .EQU      SIZE-2            ;WORD
1093 VARSIZE        .EQU      PTCOUNT           ;TOTAL BYTES
1094
1095
1096      LINK      A6,#VARSIZE                  ;ALLOCATE STACK FRAME
1097      MOVEM.L   D3/D6-D7/A2-A4,-(SP)         ;SAVE REGS
1098 ;
1099 ;  QUIT FAST IF FROMRECT = TORECT
1100 ;
1101      MOVE.L    FROMRECT(A6),A0              ;POINT TO FROMRECT
1102      MOVE.L    TORECT(A6),A1                ;POINT TO TORECT
1103      CMPM.L    (A0)+,(A1)+                  ;IS TOPLEFT SAME ?
1104      BNE.S     NOTSAME                      ;NO, CONTINUE

```

```

1105      CMPM.L   (A0)+,(A1)+      ;YES, IS BOTRIGHT SAME TOO ?
1106      BEQ.S    JDONE            ;IF SO, JUST QUIT
1107 ;
1108 ;   SPECIAL CASE RECTANGULAR RGN
1109 ;
1110 NOTSAME MOVE.L   RGN(A6),A4      ;GET RGNHANDLE
1111      MOVE.L   (A4),A0          ;DE-REFERENCE RGN
1112      CMP      #10,RGNSIZE(A0)   ;IS RGN RECTANGULAR ?
1113      BNE.S    NOTRECT          ;NO, CONTINUE
1114      PEA      RGNBBOX(A0)       ;YES, PUSH RGNBBOX
1115      MOVE.L   FROMRECT(A6),-(SP) ;PUSH FROMRECT
1116      MOVE.L   TORECT(A6),-(SP)  ;PUSH TO RECT
1117      JSR      MAPRECT          ;MapRect(rgn^.rgnBBox,from,
1118 JDONE  BRA.S    DONE            ;
1119 ;
1120 NOTRECT CLR.L    -(SP)          ;ROOM FOR FCN RESULT
1121      MOVE     #256,-(SP)        ;PUSH BYTECOUNT = 256
1122      MOVE     (SP),SIZE(A6)     ;SIZE := 256 BYTES
1123      JSR      NEWHANDLE        ;ALLOCATE PTBUF
1124      MOVE.L   (SP)+,A3         ;GET PTBUF HANDLE IN A3
1125 ;
1126      CLR      INDEX(A6)        ;INDEX := 0
1127      MOVE.L   A4,-(SP)         ;PUSH RGN
1128      MOVE.L   A3,-(SP)         ;PUSH PTBUF HANDLE
1129      PEA      INDEX(A6)        ;PUSH VAR INDEX
1130      PEA      SIZE(A6)         ;PUSH VAR SIZE
1131      JSR      PUTRGN           ;UNPACK RGN INTO INVERSION P
1132      MOVE     INDEX(A6),D7      ;GET INDEX
1133      LSR      #2,D7            ;PTCOUNT := INDEX DIV 4
1134 ;
1135 ;   MAP ALL INVERSION POINTS
1136 ;
1137      MOVE     D7,D6            ;COPY PTCOUNT FOR LOOP COUNT
1138      MOVE.L   (A3),A2          ;DE-REFERENCE PTBUF HANDLE
1139      BRA.S    MORE             ;GO TO LOOP START
1140 LOOP  MOVE.L   A2,-(SP)         ;PUSH ADDR OF AN INV PT
1141      MOVE.L   FROMRECT(A6),-(SP) ;PUSH FROMRECT
1142      MOVE.L   TORECT(A6),-(SP)  ;PUSH TORECT
1143      JSR      MAPPT           ;MAP THIS POINT
1144      ADD.L    #4,A2            ;BUMP TO NEXT POINT
1145 MORE  DBRA     D6,LOOP         ;LOOP ALL INVERSION POINTS
1146 ;
1147      MOVE.L   (A3),-(SP)        ;PUSH PTBUF PTR
1148      MOVE     D7,-(SP)         ;PUSH PTCOUNT
1149      JSR      SORTPOINTS       ;SortPoints(ptBuf^,ptCount)
1150 ;
1151      MOVE.L   (A3),-(SP)        ;PUSH PTBUF PTR
1152      MOVE     D7,PTCOUNT(A6)  ;PUT PTCOUNT IN MEMORY

```


1153	PEA	PTCOUNT(A6)	;PUSH VAR PTCOUNT
1154	JSR	CULLPOINTS	;CullPoints(ptBuf^,ptCount)
1155			
1156	MOVE.L	A3,-(SP)	;PUSH PTBUF HANDLE
1157	MOVE	PTCOUNT(A6),-(SP)	;PUSH PTCOUNT
1158	MOVE.L	A4,-(SP)	;PUSH RGN
1159	JSR	PACKRGN	;PackRgn(ptBuf,ptCount,rgn);
1160			
1161	MOVE.L	A3,A0	;PUSH PTBUF HANDLE
1162		_DisposHandle	;DISCARD IT
1163			
1164	DONE	MOVEM.L (SP)+,D3/D6-D7/A2-A4	;RESTORE REGS
1165		UNLINK PARAMSIZE, 'MAPRGN '	
1166			
1167			
1168			
1169			
1170		.END	
1171			