```
        .INCLUDE GRAFTYPES.TEXT
;------------------------------------------------------
;
;
;      ****   *****   ***   ***   ***   *   *   ***
;      *   *  *      *   *   *   *   *  *   *  *   *
;      *   *  *      *       *   *   *  **  *  *
;      ****   ***    *  **   *   *   *  * * *   ***
;      * *    *      *   *   *   *   *  *  **      *
;      *  *   *      *   *   *   *   *  *   *  *   *
;      *   *  *****   ***   ***   ***   *   *   ***
;
;
;
;  QuickDraw Routines to operate on Regions.
;

        .PROC StdRgn,2
        .REF  CheckPic,PutPicVerb,DPutPicByte,PutPicRgn
        .REF  PutRgn,FrRgn,PushVerb,DrawRgn
;-------------------------------------------------------------
;
;  PROCEDURE StdRgn(verb: GrafVerb; rgn: RgnHandle);
;
;  A6 OFFSETS OF PARAMS AFTER LINK:
;
PARAMSIZE       .EQU    6
VERB            .EQU    PARAMSIZE+8-2           ;GRAFVERB
RGN             .EQU    VERB-4                  ;LONG, RGNHANDLE

        LINK    A6,#0                           ;NO LOCALS
        MOVEM.L D6-D7/A2-A4,-(SP)               ;SAVE REGS
        MOVE.B  VERB(A6),D7                     ;GET VERB
        JSR     CHECKPIC                        ;SET UP A4,A3 AND CHECK PICSA
        BLE.S   NOTPIC                          ;BRANCH IF NOT PICSAVE

        MOVE.B  D7,-(SP)                        ;PUSH VERB
        JSR     PutPicVerb                      ;PUT ADDIONAL PARAMS TO THEPI
        MOVE    #$80,D0                         ;PUT RGNNOUN IN HI NIBBLE
        ADD     D7,D0                           ;PUT VERB IN LO NIBBLE
        JSR     DPutPicByte                     ;PUT OPCODE TO THEPIC
        MOVE.L  RGN(A6),-(SP)                   ;PUSH RGNHANDLE
        JSR     PutPicRgn                       ;PUT REGION TO THEPIC

NOTPIC  MOVE.L  RGN(A6),-(SP)                   ;PUSH RGNHANDLE
        JSR     PushVerb                        ;PUSH MODE AND PATTERN
        TST.B   D7                              ;IS VERB FRAME ?
        BNE.S   NOTFR                           ;NO, CONTINUE
```

```
        TST.L   RGNSAVE(A3)                     ;YES, IS RGNSAVE TRUE ?
        BEQ.S   NOTRGN                          ;NO, CONTINUE
        MOVE.L  RGN(A6),-(SP)                   ;YES, PUSH RGNHANDLE
        MOVE.L  RGNBUF(A4),-(SP)                ;PUSH RGNBUF
        PEA     RGNINDEX(A4)                    ;PUSH VAR RGNINDEX
        PEA     RGNMAX(A4)                      ;PUSH VAR RGNMAX
        JSR     PutRgn                          ;ADD INVERSION PTS TO THERGN

NOTRGN  JSR     FrRgn                           ;FrRgn(rgn,pnMode,pnPat)
        BRA.S   GOHOME

NOTFR   JSR     DrawRgn                         ;DrawRgn(rgn,mode,pat);

GOHOME  MOVEM.L (SP)+,D6-D7/A2-A4               ;RESTORE REGS
        UNLINK  PARAMSIZE,'STDRGN  '



        .PROC FrameRgn,1
        .DEF  CallRgn,PaintRgn,EraseRgn,InvertRgn,FillRgn
        .REF  StdRgn
;-------------------------------------------------
;
;  PROCEDURE FrameRgn(* rgn: RgnHandle *);
;
        MOVEQ   #FRAME,D0                       ;VERB = FRAME
        BRA.S   CallRgn                         ;SHARE COMMON CODE


;-------------------------------------------------
;
;  PROCEDURE PaintRgn(* rgn: RgnHandle *);
;
PaintRgn
        MOVEQ   #PAINT,D0                       ;VERB = PAINT
        BRA.S   CallRgn                         ;SHARE COMMON CODE


;-------------------------------------------------
;
;  PROCEDURE EraseRgn(* rgn: RgnHandle *);
;
EraseRgn
        MOVEQ   #ERASE,D0                       ;VERB = ERASE
        BRA.S   CallRgn                         ;SHARE COMMON CODE


;-------------------------------------------------
```

```
;
;  PROCEDURE InvertRgn(* rgn: RgnHandle *);
;
InvertRgn
        MOVEQ   #INVERT,D0                      ;VERB = INVERT
        BRA.S   CallRgn                         ;SHARE COMMON CODE


;-------------------------------------------------------
;
;  PROCEDURE FillRgn(* rgn: RgnHandle; pat: Pattern *);
;
FillRgn
        MOVE.L  (SP)+,A0                        ;POP RETURN ADDR
        MOVE.L  (SP)+,A1                        ;POP ADDR OF PATTERN
        MOVE.L  A0,-(SP)                        ;PUT RETURN ADDR BACK
        MOVE.L  GRAFGLOBALS(A5),A0              ;POINT TO LISAGRAF GLOBALS
        MOVE.L  THEPORT(A0),A0                  ;GET CURRENT GRAFPORT
        LEA     FILLPAT(A0),A0                  ;POINT TO FILLPAT
        MOVE.L  (A1)+,(A0)+                     ;COPY PAT INTO FILLPAT
        MOVE.L  (A1)+,(A0)+                     ;ALL EIGHT BYTES
        MOVEQ   #FILL,D0                        ;VERB = FILL
        BRA.S   CallRgn                         ;SHARE COMMON CODE


;-------------------------------------------------------------
;
;  PROCEDURE CallRgn(rgn: RgnHandle);
;
;  code shared by FrameRgn, PaintRgn, EraseRgn, InvertRgn, and FillRgn.
;  enter with verb in D0.
;
CallRgn
        MOVE.L  (SP)+,A0                        ;POP RETURN ADDR
        MOVE.L  (SP)+,A1                        ;POP RGN
        MOVE.B  D0,-(SP)                        ;PUSH VERB
        MOVE.L  A1,-(SP)                        ;PUSH RGN
        MOVE.L  A0,-(SP)                        ;RESTORE RETURN ADDR
        MOVE.L  GRAFGLOBALS(A5),A0              ;POINT TO LISAGRAF GLOBALS
        MOVE.L  THEPORT(A0),A0                  ;GET CURRENT GRAFPORT
        MOVE.L  GRAFPROCS(A0),D0                ;IS GRAFPROCS NIL ?
        LEA     STDRGN,A0
        BEQ.S   USESTD                          ;YES, USE STD PROC
        MOVE.L  D0,A0
        MOVE.L  RGNPROC(A0),A0                  ;NO, GET PROC PTR
USESTD  JMP     (A0)                            ;GO TO IT

```

```
145
146         .PROC DrawRgn,3
147         .REF  RgnBlt
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
```

```
241         MOVE    #10,-(SP)                       ;PUSH BYTECOUNT=10
242         JSR     NEWHANDLE                       ;ALLOCATE A RELOCATABLE OBJEC
243         MOVE.L  (SP)+,A0                        ;POP RESULTING HANDLE
244         MOVEM.L (SP)+,D3/A2                     ;RESTORE REGS
245         MOVE.L  A0,4(SP)                        ;STORE INTO NEWRGN RESULT
246         MOVE.L  (A0),A0                         ;DE-REFERENCE HANDLE
247         MOVE    #10,(A0)+                       ;INSTALL RGNSIZE=10
248         CLR.L   (A0)+                           ;INSTALL RGNBBOX=(0,0,0,0)
249         CLR.L   (A0)+
250         RTS                                     ;RETURN TO CALLER
251
252
253
254         .PROC DisposeRgn,1
255 ;-----------------------------------------------
256 ;
257 ;  PROCEDURE DisposeRgn(rgn: RgnHandle);
258 ;
259         MOVE.L  (SP)+,A1                        ;pop return addr
260         MOVE.L  (SP)+,A0                        ;pop handle
261         _DisposHandle                           ;discard it
262         JMP     (A1)                            ;and return
263
264
265
266         .PROC OpenRgn,0
267         .REF  NewHandle,HidePen
268 ;-----------------------------------------------
269 ;
270 ;  PROCEDURE OpenRgn;
271 ;
272         MOVEM.L D3/A2-A4,-(SP)                  ;SAVE REGS
273         MOVE.L  GRAFGLOBALS(A5),A4              ;POINT TO QUICKDRAW GLOBALS
274         MOVE.L  THEPORT(A4),A0                  ;GET CURRENT GRAFPORT
275         MOVE.L  #1,D0
276         MOVE.L  D0,RGNSAVE(A0)                  ;RGNSAVE := TRUE
277         CLR     RGNINDEX(A4)                    ;RGNINDEX := 0
278         CLR.L   -(SP)                           ;MAKE ROOM FOR FCN RESULT
279         MOVE    #256,-(SP)                      ;PUSH BYTE COUNT = 256
280         MOVE    (SP),RGNMAX(A4)                 ;RGNMAX := 256 TOO;
281         JSR     NEWHANDLE
282         MOVE.L  (SP)+,RGNBUF(A4)                ;RGNBUF := NEWHANDLE(RGNMAX)
283         JSR     HidePen
284         MOVEM.L (SP)+,D3/A2-A4                  ;RESTORE REGS
285         RTS                                     ;AND RETURN
286
287
288
```

```
        .PROC CloseRgn,1
        .REF  ShowPen,SortPoints,CullPoints,PackRgn
;-----------------------------------------------
;
;  PROCEDURE CloseRgn(* dstRgn: RgnHandle *);
;  Sort array of inversion points and pack into region.
;
;  A6 OFFSETS OF PARAMS AND LOCALS AFTER LINK:
;
PARAMSIZE       .EQU    4                       ;TOTAL BYTES OF PARAMETERS
DSTRGN          .EQU    PARAMSIZE+8-4           ;LONG, RGNHANDLE
PTCOUNT         .EQU    -2                      ;WORD
VARSIZE         .EQU    PTCOUNT                 ;SIZE OF LOCAL VARS


        LINK    A6,#VARSIZE
        MOVEM.L D3/A2-A4,-(SP)                  ;SAVE REGS
        MOVE.L  GRAFGLOBALS(A5),A4              ;POINT TO QUICKDRAW GLOBALS
        MOVE.L  THEPORT(A4),A0                  ;GET CURRENT GRAFPORT
        TST.L   RGNSAVE(A0)                     ;IS RGNSAVE TRUE ?
        BEQ.S   DONE                            ;NO, ABORT
        CLR.L   RGNSAVE(A0)                     ;YES, RGNSAVE := FALSE
        JSR     SHOWPEN                         ;UNDO THE HIDEPEN FROM OPENR
        MOVE    RGNINDEX(A4),D0                 ;GET CURRENT RGNINDEX
        LSR     #2,D0                           ;DIV BY 4
        MOVE    D0,PTCOUNT(A6)                  ;FOR PTCOUNT
        MOVE.L  RGNBUF(A4),A3                   ;GET RGNBUF HANDLE
        MOVE.L  (A3),-(SP)                      ;PUSH BUF POINTER
        MOVE    D0,-(SP)                        ;PUSH PTCOUNT
        JSR     SORTPOINTS                      ;QUICKSORT IN VH ORDER
        MOVE.L  (A3),-(SP)                      ;PUSH BUF POINTER
        PEA     PTCOUNT(A6)                     ;PUSH VAR PTCOUNT
        JSR     CULLPOINTS                      ;CANCEL DUPLICATE PAIRS
        MOVE.L  A3,-(SP)                        ;PUSH RGNBUF HANDLE
        MOVE    PTCOUNT(A6),-(SP)               ;PUSH (UPDATED) PTCOUNT
        MOVE.L  DSTRGN(A6),-(SP)                ;PUSH DSTRGN
        JSR     PACKRGN                         ;PACK POINTS INTO DSTRGN
        MOVE.L  A3,A0                           ;GET RGNBUF HANDLE
        _DisposHandle                           ;DISCARD IT
DONE    MOVEM.L (SP)+,D3/A2-A4                  ;RESTORE REGS
        UNLINK  PARAMSIZE, 'CLOSERGN'


        .PROC CopyRgn,1
        .REF  SetSize
;-----------------------------------------------
;
```

```
;  PROCEDURE CopyRgn(* srcRgn,dstRgn: RgnHandle *);
;
PARAMSIZE       .EQU    8
SRCRGN          .EQU    PARAMSIZE+8-4           ;RGNHANDLE
DSTRGN          .EQU    SRCRGN-4                ;RGNHANDLE

        LINK    A6,#0                           ;ESTABLISH STACK FRAME
        MOVE.L  SRCRGN(A6),A0                   ;GET SRCRGN HANDLE
        MOVE.L  DSTRGN(A6),A1                   ;GET DSTRGN HANDLE
        CMP.L   A0,A1                           ;ARE THEY THE SAME?
        BEQ.S   DONE                            ;YES, QUIT

        MOVE.L  (A0),A0                         ;DE-REFERENCE SRCRGN HANDLE
        MOVE.L  (A1),A1                         ;DE-REFERENCE DSTRGN HANDLE
        MOVE    RGNSIZE(A0),D0                  ;GET SRC SIZE
        CMP     RGNSIZE(A1),D0                  ;IS DST SIZE SAME AS SRC ?
        BEQ.S   COPY                            ;YES, CONTINUE

        MOVEM.L D0/D3/A2,-(SP)                  ;SAVE REGS AND BYTECOUNT
        MOVE.L  DSTRGN(A6),-(SP)                ;PUSH DSTRGN HANDLE
        MOVE    D0,-(SP)                        ;PUSH NEWSIZE=SRC SIZE
        JSR     SETSIZE                         ;CHANGE SIZE OF DST
        MOVEM.L (SP)+,D0/D3/A2                  ;RESTORE REGS AND BYTECOUNT
        MOVE.L  SRCRGN(A6),A0                   ;GET SRCRGN HANDLE
        MOVE.L  DSTRGN(A6),A1                   ;GET DSTRGN HANDLE
        MOVE.L  (A0),A0                         ;DE-REFERENCE SRCRGN HANDLE
        MOVE.L  (A1),A1                         ;DE-REFERENCE DSTRGN HANDLE

COPY    LSR     #2,D0                           ;LONGCOUNT := BYTECOUNT DIV
        BCC.S   EVEN                            ;WAS THERE AN ODD WORD ?
        MOVE    (A0)+,(A1)+                     ;YES, DO IT TO MAKE EVEN
        BRA.S   EVEN                            ;AND CONTINUE
COPYLP  MOVE.L  (A0)+,(A1)+                     ;COPY A LONG OF SRC TO DST
EVEN    DBRA    D0,COPYLP                       ;LOOP FOR ALL LONGS

DONE    UNLINK  PARAMSIZE,'COPYRGN '



        .PROC SetEmptyRgn,1
        .REF  SetRectRgn
;-----------------------------------------------
;
;  PROCEDURE SetEmptyRgn(rgn: RgnHandle);
;
        MOVE.L  (SP)+,A0                        ;POP RETURN ADDR
        CLR.L   -(SP)                           ;PUSH LEFT, TOP
        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
```

```
;
        MOVE.L  (SP)+,A0                        ;POP RETURN ADDR
        MOVE.L  (SP)+,A1                        ;POP ADDR OF RECT
        MOVE.L  (A1)+,-(SP)                     ;PUSH LEFT,TOP
        MOVE.L  (A1)+,-(SP)                     ;PUSH RIGHT,BOTTOM
        MOVE.L  A0,-(SP)                        ;RESTORE RETURN ADDR
        JMP     SETRECTRGN                      ;DOUBLE UP ON CODE



        .PROC OffsetRgn,3
;------------------------------------------------------
;
;  PROCEDURE OffsetRgn(rgn: RgnHandle; dh,dv: INTEGER);
;
        MOVE.L  (SP)+,A1                        ;POP RETURN ADDR
        MOVE    (SP)+,D1                        ;POP DV
        MOVE    (SP)+,D0                        ;POP DH
        MOVE.L  (SP)+,A0                        ;POP RGN HANDLE
        MOVE.L  (A0),A0                         ;DE-REFERENCE IT
        ADD     #2,A0                           ;POINT TO RGNBBOX

;------------------------------------------------------
;
;  OFFSET THE BOUNDING BOX
;
        ADD     D1,(A0)+                        ;ADD DV TO TOP
        ADD     D0,(A0)+                        ;ADD DH TO LEFT
        ADD     D1,(A0)+                        ;ADD DV TO BOTTOM
        ADD     D0,(A0)+                        ;ADD DH TO RIGHT


;-----------------------------------------------------------------
;
;  IF NON-RECTANGULAR REGION, OFFSET THE COORDINATES TOO.
;
        CMP     #10,-10(A0)                     ;IS REGION RECTANGULAR ?
        BEQ.S   DONE                            ;IF SO, WE'RE ALL DONE
NXTVERT ADD     D1,(A0)+                        ;OFFSET VERTICAL COORD DV
NXTHOR  ADD     D0,(A0)+                        ;OFFSET LEFT COORD DH
        ADD     D0,(A0)+                        ;OFFSET RIGHT COORD DH
        CMP     #32767,(A0)                     ;HORIZ TERMINATOR ?
        BNE     NXTHOR                          ;NO, LOOP
        ADD     #2,A0                           ;YES, SKIP OVER TERMINATOR
        CMP     #32767,(A0)                     ;IS NEXT VERT = 32767 ?
        BNE     NXTVERT                         ;NO, LOOP FOR MORE
DONE    JMP     (A1)                            ;RETURN

```

```
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
```

```
;-------------------------------------------------
;
;  THE REGION IS NOT RECTANGULAR. ALLOCATE A POINT BUFFER IN THE HEAP.
;
NOTRECT ADD     D7,D7                           ;TRY BYTES NEEDED = RGNSIZE*
        CLR.L   -(SP)                           ;ROOM FOR FCN RESULT
        MOVE    D7,-(SP)                        ;PUSH BYTESNEEDED
        JSR     NEWHANDLE                       ;ALLOCATE PTBUF
        MOVE.L  (SP)+,A3                        ;PUT PTBUF HANDLE IN A3
        BSR.S   HINSET                          ;INSET HORIZ AND FLIP
        SWAP    D6                              ;GET DV INSTEAD OF DH
        BSR.S   HINSET                          ;INSET VERTICAL AND FLIP BAC
        BRA.S   DONE                            ;ALL DONE


;-----------------------------------------------------------------------
;
;  LOCAL ROUTINE TO INSET HORIZONTALLY, SWAP H AND V COORDS, AND RE-SORT
;
;  RgnOp(rgn,rgn,bufHandle,maxBytes,op,dh,TRUE);
;
HINSET  CLR.W   -(SP)                           ;ROOM FOR FCN RESULT
        MOVE.L  A4,-(SP)                        ;PUSH RGNA = USER RGN
        MOVE.L  A4,-(SP)                        ;PUSG RGNB = USER RGN ALSO
        MOVE.L  A3,-(SP)                        ;PUSH BUFHANDLE
        MOVE    D7,-(SP)                        ;PUSH MAXBYTES
        MOVE    #8,-(SP)                        ;PUSH OP = INSET
        MOVE    D6,-(SP)                        ;PUSH DH
        ST      -(SP)                           ;PUSH OKGROW = TRUE
        JSR     RGNOP
        MOVE.W  (SP)+,D5                        ;GET PTCOUNT IN D5
;
;  SWAP VERT AND HORIZ COORDS OF ALL INVERSION POINTS
;
        MOVE.L  (A3),A0                         ;DE-REFERENCE PTBUF HANDLE
        MOVE    D5,D1                           ;COPY PTCOUNT
        BRA.S   SWAP2                           ;GO TO LOOP START
SWAPLP  MOVE.L  (A0),D0                         ;GET A POINT
        SWAP    D0                              ;SWAP ITS COORDINATES
        MOVE.L  D0,(A0)+                        ;PUT IT BACK INTO ARRAY
SWAP2   DBRA    D1,SWAPLP                       ;LOOP FOR ALL POINTS
;
;  RE-SORT THE POINTS IN V.H ORDER
;
        MOVE.L  (A3),-(SP)                      ;PUSH PTBUF PTR
        MOVE    D5,-(SP)                        ;PUSH PTCOUNT
        JSR     SORTPOINTS                      ;RE-SORT INTO VH ORDER
;
```

```
;  PACK THE RESULTING POINTS
;
        MOVE.L  A3,-(SP)                        ;PUSH PTBUF HANDLE
        MOVE    D5,-(SP)                        ;PUSH PTCOUNT
        MOVE.L  A4,-(SP)                        ;PUSH RGN HANDLE
        JSR     PACKRGN                         ;PackRgn(ptBuf,ptCount,rgn);
        RTS                                     ;RETURN TO LOCAL CALLER


;-----------------------------------------
;
;  DISCARD POINT BUFFER AND QUIT.
;
DONE    MOVE.L  A3,A0                           ;GET PTBUF HANDLE
        _DisposHandle                           ;DISCARD IT

GOHOME  MOVEM.L (SP)+,D3-D7/A2-A4               ;RESTORE REGS
        UNLINK  PARAMSIZE,'INSETRGN'



        .FUNC EmptyRgn,1
        .REF  EmptyRect
;-----------------------------------------------
;
;  FUNCTION EmptyRgn(rgn: RgnHandle): BOOLEAN;
;
        MOVE.L  (SP)+,A0                        ;POP RETURN ADDR
        MOVE.L  (SP)+,A1                        ;POP RGNHANDLE
        MOVE.L  (A1),A1                         ;DE-REFERENCE HANDLE
        PEA     RGNBBOX(A1)                     ;PUSH RGNBBOX
        MOVE.L  A0,-(SP)                        ;PUSH RETURN ADDR
        JMP     EMPTYRECT                       ;USE EMPTYRECT CODE



        .FUNC EqualRgn,2
;-------------------------------------------------
;
;  FUNCTION EqualRgn(rgnA,rgnB: RgnHandle): BOOLEAN;
;
;  RETURNS TRUE IF RGNA AND RGNB DESCRIBE THE SAME AREA
;
        MOVE.L  4(SP),A0                ;GET RGNA HANDLE
        MOVE.L  8(SP),A1                ;GET RGNB HANDLE
        CMP.L   A0,A1                   ;ARE THEY THE SAME ?
        BEQ.S   TRUE                    ;YES, THE REGIONS ARE THE SAME
        MOVE.L  (A0),A0                 ;DE-REFERENCE RGN A
```

```
        MOVE.L  (A1),A1                 ;DE-REFERENCE RGN B
        MOVE    (A0),D0                 ;GET RGNSIZE A
        MOVE    D0,D1                   ;MAKE AN EXTRA COPY
        LSR     #2,D1                   ;DIV BY 4 FOR LONG COUNT
        SUB     #1,D1                   ;INIT DBRA LOOP COUNT
LOOP    CMPM.L  (A0)+,(A1)+             ;COMPARE A LONG
        DBNE    D1,LOOP                 ;LOOK TILL DIFFERENT OR COUNT
        BNE.S   FALSE                   ;DIFFERENT --> FALSE
        AND     #3,D0                   ;GET 0..3 FINISH UP BYTES
        BEQ.S   TRUE                    ;IF NO MORE, WE MADE IT
LOOP2   CMPM.B  (A0)+,(A1)+             ;COMPARE A BYTE
        BNE.S   FALSE                   ;BR IF DIFFERENT
        SUB     #1,D0
        BNE.S   LOOP2                   ;LOOP LAST 1..3 BYTES
TRUE    MOVE.L  #1,D0                   ;TRUE
        BRA.S   DONE
FALSE   MOVE.L  #0,D0                   ;FALSE
DONE    MOVE.B  D0,12(SP)               ;SET RESULT
        MOVE.L  (SP)+,A0                ;POP RETURN ADDR
        ADD     #8,SP                   ;STRIP PARAMETERS
        JMP     (A0)                    ;RETURN



        .PROC SectRgn,3
        .DEF  DoRgnOp,UnionRgn,DiffRgn,XorRgn
        .REF  EqualRgn,CopyRgn,RSect,RectRgn,SetEmptyRgn
        .REF  NewHandle,RgnOp,PackRgn
;-------------------------------------------------------
;
;  PROCEDURE SectRgn(srcRgnA,srcRgnB,dstRgn: RgnHandle);
;  calculate the intersection of two regions.
;
        MOVEQ   #0,D0                           ;OP = SECT
        BRA.S   DoRgnOp                         ;SHARE COMMON CODE



;-------------------------------------------------------
;
;  PROCEDURE UnionRgn(srcRgnA,srcRgnB,dstRgn: RgnHandle);
;  calculate the union of two regions.
;
UnionRgn
        MOVEQ   #4,D0                           ;OP = UNION
        BRA.S   DoRgnOp                         ;SHARE COMMON CODE

```

```
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
```

```
;
;  THE BOUNDING BOXES DON'T INTERSECT.
;  IF OP = SECT, THEN RETURN EMPTY.
;  IF OP = DIFF, THEN COPY RGNA INTO DSTRGN.
;
        TST     D5                              ;IS OP = SECT ?
        BEQ     ZERO                            ;YES, RETURN EMPTY
        BRA     COPY                            ;NO, COPY SRCA INTO DSTRGN

;
;  IF OP = SECT, THEN CHECK FOR BOTH INPUTS RECTANGULAR
;
NOTEMPTY MOVE.L (A2),A0                         ;DE-REFERENCE RGNA
        MOVE.L  (A3),A1                         ;DE-REFERENCE RGNB
        TST     D5                              ;IS OP = SECT ?
        BNE.S   NOTEASY                         ;NO, CONTINUE
        MOVEQ   #10,D0
        CMP     RGNSIZE(A0),D0                  ;IS RGNA RECTANGULAR ?
        BNE.S   NOTEASY                         ;NO, CONTINUE
        CMP     RGNSIZE(A1),D0                  ;IS RGNB RECTANGULAR ?
        BNE.S   NOTEASY                         ;NO, CONTINUE
        MOVE.L  A4,-(SP)                        ;PUSH DSTRGN
        PEA     TEMPRECT(A6)                    ;PUSH TEMPRECT
        JSR     RECTRGN                         ;RectRgn(dstRgn,tempRect);
JDONE   BRA.S   DONE

;
;  OP = UNION OR XOR:  IF EITHER REGION IS EMPTY, COPY THE OTHER.
;
UNIXOR  MOVE    RGNBBOX+LEFT(A1),D0             ;GET RGNB BBOX LEFT
        CMP     RGNBBOX+RIGHT(A1),D0            ;IS RGNB BBOX LEFT >= RIGHT
        BGE     COPY                            ;YES, COPY RGNA INTO DST

        MOVE    RGNBBOX+LEFT(A0),D0             ;GET RGNA BBOX LEFT
        CMP     RGNBBOX+RIGHT(A0),D0            ;IS RGNA BBOX LEFT >= RIGHT
        BLT.S   NOTEASY                         ;NO, CONTINUE
        MOVE.L  A3,A2                           ;YES, GET RGNB INSTEAD
        BRA     COPY                            ;COPY RGNB INTO DST


NOTEASY MOVE    RGNSIZE(A0),D4                  ;GET RGNA RGNSIZE
        ADD     RGNSIZE(A1),D4                  ;ADD RGNB RGNSIZE
        ADD     D4,D4                           ;TRY DOUBLE FOR BYTECOUNT
        CLR.L   -(SP)                           ;MAKE ROOM FOR FCN RESULT
        MOVE    D4,-(SP)                        ;PUSH BYTECOUNT
        JSR     NEWHANDLE                       ;ALLOCATE PTBUF
        MOVE.L  (SP)+,A3                        ;GET PTBUF HANDLE IN A3
;
```

```
817 ;  PtCount := RgnOp(srcA,srcB,bufHandle,maxBytes,op,0,TRUE);
818 ;
819         CLR.W   -(SP)                           ;MAKE ROOM FOR FCN RESULT
820         MOVE.L  RGNA(A6),-(SP)                  ;PUSH RGNA
821         MOVE.L  RGNB(A6),-(SP)                  ;PUSH RGNB
822         MOVE.L  A3,-(SP)                        ;PUSH BUFHANDLE
823         MOVE    D4,-(SP)                        ;PUSH MAXBYTES
824         MOVE    D5,-(SP)                        ;PUSH OP
825         CLR.W   -(SP)                           ;PUSH DH=0
826         ST      -(SP)                           ;PUSH OKGROW = TRUE
827         JSR     RGNOP
828         MOVE    (SP)+,D6                        ;GET PTCOUNT
829
830         MOVE.L  A3,-(SP)                        ;PUSH PTBUF HANDLE
831         MOVE    D6,-(SP)                        ;PUSH PTCOUNT
832         MOVE.L  A4,-(SP)                        ;PUSH DSTRGN
833         JSR     PACKRGN                         ;PackRgn(ptBuf,ptCount,dstRg
834
835         MOVE.L  A3,A0                           ;GET PTBUF HANDLE
836         _DisposHandle                           ;DISCARD IT
837
838 DONE    MOVEM.L (SP)+,D3-D7/A2-A4               ;RESTORE REGS
839         UNLINK  PARAMSIZE,'DORGNOP '
840
841
842
843         .FUNC PtInRgn,2
844 ;---------------------------------------------------------
845 ;
846 ;  FUNCTION PtInRgn(pt: Point; rgn: RgnHandle): BOOLEAN;
847 ;
848 ;  TESTS IF A GIVEN POINT IS INSIDE A REGION.
849 ;
850 ;  A6 OFFSETS OF PARAMETERS AFTER LINK:
851 ;
852 PARAMSIZE       .EQU    8                       ;SIZE OF PARAMETERS
853 RESULT          .EQU    PARAMSIZE+8             ;BOOLEAN
854 PT              .EQU    RESULT-4                ;POINT, VALUE
855 RGN             .EQU    PT-4                    ;LONG, HANDLE
856
857
858 ENTRY   LINK    A6,#0                           ;NO LOCAL VARS
859         MOVE.L  D3,-(SP)                        ;SAVE REG
860         MOVE    PT+H(A6),D1                     ;GET TEST HORIZ
861         MOVE    PT+V(A6),D2                     ;GET TEST VERT
862         CLR     D3                              ;INIT INSIDE:=FALSE
863
864
```

```
;-------------------------------------------------------
;
;  FIRST CHECK BOUNDING BOX
;
        MOVE.L  RGN(A6),A0                      ;GET RGN HANDLE
        MOVE.L  (A0),A0                         ;DE-REFERENCE IT
        CMP     RGNBBOX+LEFT(A0),D1             ;IS PT.H < BBOX LEFT ?
        BLT.S   DONE                            ;YES, RETURN FALSE
        CMP     RGNBBOX+RIGHT(A0),D1            ;IS PT.H >= BBOX RIGHT ?
        BGE.S   DONE                            ;YES, RETURN FALSE
        CMP     RGNBBOX+TOP(A0),D2              ;IS PT.V < BBOX TOP ?
        BLT.S   DONE                            ;YES, RETURN FALSE
        CMP     RGNBBOX+BOTTOM(A0),D2           ;IS PT.V >= BBOX BOT ?
        BGE.S   DONE                            ;YES, RETURN FALSE
        CMP     #10,RGNSIZE(A0)                 ;IS REGION RECTANGULAR ?
        BNE.S   NOTRECT                         ;NO, CONTINUE
        NOT     D3                              ;YES, RETURN TRUE
        BRA.S   DONE


;-------------------------------------------------------------
;
;  PT IS INSIDE BOUNDING BOX AND REGION IS NOT RECTANGULAR.
;  LOOK AT THE INVERSION POINTS TO DETERMINE IF PT IN REGION.
;
NOTRECT LEA     RGNDATA(A0),A0                  ;POINT TO FIRST VERT COORD
NXTVERT CMP     (A0)+,D2                        ;IS NEXT VERT > PT.V ?
        BLT.S   DONE                            ;YES, QUIT
NEXTHOR MOVE    (A0)+,D0                        ;GET HORIZ COORD
        CMP     #32767,D0                       ;IS IT THE TERMINATOR ?
        BEQ     NXTVERT                         ;YES, GET NEXT VERT COORD
        CMP     D1,D0                           ;IS HORIZ <= PT.H ?
        BGT     NEXTHOR                         ;NO, IGNORE THIS POINT
        NOT     D3                              ;YES, TOGGLE INSIDE
        BRA     NEXTHOR                         ;AND GO FOR MORE POINTS
DONE    NEG.B   D3                              ;BOOLEAN RESULT IS 0 OR 1
        MOVE.B  D3,RESULT(A6)                   ;RETURN BOOLEAN FCN RESULT
        MOVE.L  (SP)+,D3                        ;RESTORE REG
        UNLINK  PARAMSIZE,'PTINRGN '



        .FUNC RectInRgn,2
        .REF  RSect,InitRgn,SeekRgn
;-----------------------------------------------
;
;  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 FALS
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 RESU
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 ;-----------------------------------------------------
```

```
;
;  FUNCTION TrimRect(rgn: RgnHandle; VAR dstRect: Rect): CCR TRISTATE;
;
;  RESULT IN CONDITION CODES:
;
;  =  RESULT RECTANGULAR, DSTRECT TRIMMED
;  <  RESULT EMPTY, DSTRECT NOT MODIFIED
;  >  RESULT NON-RECT, DSTRECT NOT MODIFIED
;
;  If the intersection of rgn and dstRect is rectangular,
;  then return EQUAL and put the intersection into dstRect.
;  If the intersection is empty or not rectangular, then
;  return FALSE and don't modify dstRect.
;
;   Does not call the storage allocator.
;
;   1.  Fake up a rect rgn on the stack from dstRect
;   2.  Call RgnOp with max bytes = 24, OKGROW = FALSE
;   3a. If ptCount = 4 THEN result rect, return TRUE and update dstRect.
;   3b. If ptCount < 4 THEN result empty, return TRUE and clear dstRect.
;   3c. If ptCount > 4 THEN result not rect, return FALSE
;
PARAMSIZE       .EQU    8
RGN             .EQU    PARAMSIZE+8-4           ;LONG, RGNHANDLE
DSTRECT         .EQU    RGN-4                   ;LONG, ADDR OF DSTRECT

PTDATA          .EQU    -24                     ;ROOM FOR 6 POINTS
PTMASTER        .EQU    PTDATA-4                ;LONG, FAKE MASTER
REGION          .EQU    PTMASTER-10             ;ROOM FOR RECT RGN DATA
RGNMASTER       .EQU    REGION-4                ;LONG
VARSIZE         .EQU    RGNMASTER

        LINK    A6,#VARSIZE                     ;ALLOCATE STACK FRAME
        MOVEM.L D0-D2/A1,-(SP)                  ;SAVE ALL REGS USED
        LEA     PTDATA(A6),A0                   ;POINT TO BUFFER
        MOVE.L  A0,PTMASTER(A6)                 ;INSTALL INTO FAKE MASTER
        LEA     REGION(A6),A1                   ;POINT TO REGION DATA
        MOVE.L  A1,RGNMASTER(A6)                ;INSTALL INTO FAKE MASTER
        MOVE    #10,(A1)+                       ;INSTALL RGNSIZE = 10
        MOVE.L  DSTRECT(A6),A0                  ;POINT TO DSTRECT
        MOVE.L  (A0)+,(A1)+                     ;COPY DSTRECT TOPLEFT
        MOVE.L  (A0)+,(A1)+                     ;COPY DSTRECT BOTRIGHT
;
;  RgnOp(rgn,rectRgn,BufHandle,24,sect,0,FALSE)
;
        CLR.W   -(SP)                           ;ROOM FOR FCN RESULT
        MOVE.L  RGN(A6),-(SP)                   ;PUSH RGN
        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
```