SUMMARY OF QUICKDRAW

Constants

```
CONST { Source transfer modes }

      srcCopy     = 0;
      srcOr       = 1;
      srcXor      = 2;
      srcBic      = 3;
      notSrcCopy  = 4;
      notSrcOr    = 5;
      notSrcXor   = 6;
      notSrcBic   = 7;

      { Pattern transfer modes }

      patCopy     = 8;
      patOr       = 9;
      patXor      = 10;
      patBic      = 11;
      notPatCopy  = 12;
      notPatOr    = 13;
      notPatXor   = 14;
      notPatBic   = 15;

      { Standard colors for ForeColor and BackColor }

      blackColor    = 33;
      whiteColor    = 30;
      redColor      = 205;
      greenColor    = 341;
      blueColor     = 409;
      cyanColor     = 273;
      magentaColor  = 137;
      yellowColor   = 69;

      { Standard picture comments }

      picLParen = 0;
      picRParen = 1;
```

Data Types

```
TYPE  StyleItem  = (bold,italic,underline,outline,shadow,condense,extend);
      Style      = SET OF StyleItem;
```

```
VHSelect = (v,h);
Point    = RECORD CASE INTEGER OF
             0: (v: INTEGER;   {vertical coordinate}
                 h: INTEGER); {horizontal coordinate}
             1: (vh: ARRAY[VHSelect] OF INTEGER)
           END;

Rect = RECORD CASE INTEGER OF
         0: (top:      INTEGER;
             left:     INTEGER;
             bottom:   INTEGER;
             right:    INTEGER);
         1: (topLeft:   Point;
             botRight:  Point)
       END;

RgnHandle = ^RgnPtr;
RgnPtr    = ^Region;
Region    =  RECORD
               rgnSize: INTEGER;   {size in bytes}
               rgnBBox: Rect;      {enclosing rectangle}
               {more data if not rectangular}
             END;

BitMap = RECORD
           baseAddr: Ptr;       {pointer to bit image}
           rowBytes: INTEGER;   {row width}
           bounds:   Rect       {boundary rectangle}
         END;

Pattern = PACKED ARRAY[0..7] OF 0..255;

Bits16 = ARRAY[0..15] OF INTEGER;

Cursor = RECORD
           data:    Bits16;   {cursor image}
           mask:    Bits16;   {cursor mask}
           hotSpot: Point     {point aligned with mouse}
         END;
```

```
QDProcsPtr = ^QDProcs;
QDProcs    = RECORD
               textProc:    Ptr;  {text drawing}
               lineProc:    Ptr;  {line drawing}
               rectProc:    Ptr;  {rectangle drawing}
               rRectProc:   Ptr;  {roundRect drawing}
               ovalProc:    Ptr;  {oval drawing}
               arcProc:     Ptr;  {arc/wedge drawing}
               rgnProc:     Ptr;  {region drawing}
               bitsProc:    Ptr;  {bit transfer}
               commentProc: Ptr;  {picture comment processing}
               txMeasProc:  Ptr;  {text width measurement}
               getPicProc:  Ptr;  {picture retrieval}
               putPicProc:  Ptr   {picture saving}
             END;

GrafPtr   = ^GrafPort;
GrafPort  = RECORD
              device:     INTEGER;    {device-specific information}
              portBits:   BitMap;     {grafPort's bit map}
              portRect:   Rect;       {grafPort's rectangle}
              visRgn:     RgnHandle;  {visible region}
              clipRgn:    RgnHandle;  {clipping region}
              bkPat:      Pattern;    {background pattern}
              fillPat:    Pattern;    {fill pattern}
              pnLoc:      Point;      {pen location}
              pnSize:     Point;      {pen size}
              pnMode:     INTEGER;    {pen's transfer mode}
              pnPat:      Pattern;    {pen pattern}
              pnVis:      INTEGER;    {pen visibility}
              txFont:     INTEGER;    {font number for text}
              txFace:     Style;      {text's character style}
              txMode:     INTEGER;    {text's transfer mode}
              txSize:     INTEGER;    {font size for text}
              spExtra:    Fixed;      {extra space}
              fgColor:    LONGINT;    {foreground color}
              bkColor:    LONGINT;    {background color}
              colrBit:    INTEGER;    {color bit}
              patStretch: INTEGER;    {used internally}
              picSave:    Handle;     {picture being saved}
              rgnSave:    Handle;     {region being saved}
              polySave:   Handle;     {polygon being saved}
              grafProcs:  QDProcsPtr  {low-level drawing routines}
            END;

PicHandle = ^PicPtr;
PicPtr    = ^Picture;
Picture   = RECORD
              picSize:  INTEGER;  {size in bytes}
              picFrame: Rect;     {picture frame}
              {picture definition data}
            END;
```

```
PolyHandle = ^PolyPtr;
PolyPtr    = ^Polygon;
Polygon    = RECORD
               polySize:   INTEGER;    {size in bytes}
               polyBBox:   Rect;       {enclosing rectangle}
               polyPoints: ARRAY[0..0] OF Point
             END;

PenState = RECORD
             pnLoc:   Point;    {pen location}
             pnSize:  Point;    {pen size}
             pnMode:  INTEGER;  {pen's transfer mode}
             pnPat:   Pattern   {pen pattern}
           END;

FontInfo = RECORD
             ascent:  INTEGER;  {ascent}
             descent: INTEGER;  {descent}
             widMax:  INTEGER;  {maximum character width}
             leading: INTEGER   {leading}
           END;

GrafVerb = (frame,paint,erase,invert,fill);
```

Variables

```
VAR thePort:    GrafPtr;   {pointer to current grafPort}
    white:      Pattern;   {all-white pattern}
    black:      Pattern;   {all-black pattern}
    gray:       Pattern;   {50% gray pattern}
    ltGray:     Pattern;   {25% gray pattern}
    dkGray:     Pattern;   {75% gray pattern}
    arrow:      Cursor;    {standard arrow cursor}
    screenBits: BitMap;    {the entire screen}
    randSeed:   LONGINT;   {determines where Random sequence begins}
```

Routines

GrafPort Routines

```
PROCEDURE InitGraf     (globalPtr: Ptr);
PROCEDURE OpenPort     (port: GrafPtr);
PROCEDURE InitPort     (port: GrafPtr);
PROCEDURE ClosePort    (port: GrafPtr);
PROCEDURE SetPort      (port: GrafPtr);
PROCEDURE GetPort      (VAR port: GrafPtr);
PROCEDURE GrafDevice   (device: INTEGER);
PROCEDURE SetPortBits  (bm: BitMap);
PROCEDURE PortSize     (width,height: INTEGER);
```

```
PROCEDURE MovePortTo    (leftGlobal,topGlobal: INTEGER);
PROCEDURE SetOrigin     (h,v: INTEGER);
PROCEDURE SetClip       (rgn: RgnHandle);
PROCEDURE GetClip       (rgn: RgnHandle);
PROCEDURE ClipRect      (r: Rect);
PROCEDURE BackPat       (pat: Pattern);
```

Cursor Handling

```
PROCEDURE InitCursor;
PROCEDURE SetCursor (crsr: Cursor);
PROCEDURE HideCursor;
PROCEDURE ShowCursor;
PROCEDURE ObscureCursor;
```

Pen and Line Drawing

```
PROCEDURE HidePen;
PROCEDURE ShowPen;
PROCEDURE GetPen        (VAR pt: Point);
PROCEDURE GetPenState   (VAR pnState: PenState);
PROCEDURE SetPenState   (pnState: PenState);
PROCEDURE PenSize       (width,height: INTEGER);
PROCEDURE PenMode       (mode: INTEGER);
PROCEDURE PenPat        (pat: Pattern);
PROCEDURE PenNormal;
PROCEDURE MoveTo        (h,v: INTEGER);
PROCEDURE Move          (dh,dv: INTEGER);
PROCEDURE LineTo        (h,v: INTEGER);
PROCEDURE Line          (dh,dv: INTEGER);
```

Text Drawing

```
PROCEDURE TextFont      (font: INTEGER);
PROCEDURE TextFace      (face: Style);
PROCEDURE TextMode      (mode: INTEGER);
PROCEDURE TextSize      (size: INTEGER);
PROCEDURE SpaceExtra    (extra: Fixed);
PROCEDURE DrawChar      (ch: CHAR);
PROCEDURE DrawString    (s: Str255);
PROCEDURE DrawText      (textBuf: Ptr; firstByte,byteCount: INTEGER);
FUNCTION  CharWidth     (ch: CHAR) : INTEGER;
FUNCTION  StringWidth   (s: Str255) : INTEGER;
FUNCTION  TextWidth     (textBuf: Ptr; firstByte,byteCount: INTEGER) :
                         INTEGER;
PROCEDURE GetFontInfo   (VAR info: FontInfo);
```

Drawing in Color

```
PROCEDURE ForeColor  (color: LONGINT);
PROCEDURE BackColor  (color: LONGINT);
PROCEDURE ColorBit   (whichBit: INTEGER);
```

Calculations with Rectangles

```
PROCEDURE SetRect     (VAR r: Rect; left,top,right,bottom: INTEGER);
PROCEDURE OffsetRect  (VAR r: Rect; dh,dv: INTEGER);
PROCEDURE InsetRect   (VAR r: Rect; dh,dv: INTEGER);
FUNCTION  SectRect    (src1,src2: Rect; VAR dstRect: Rect) : BOOLEAN;
PROCEDURE UnionRect   (src1,src2: Rect; VAR dstRect: Rect);
FUNCTION  PtInRect    (pt: Point; r: Rect) : BOOLEAN;
PROCEDURE Pt2Rect     (pt1,pt2: Point; VAR dstRect: Rect);
PROCEDURE PtToAngle   (r: Rect; pt: Point; VAR angle: INTEGER);
FUNCTION  EqualRect   (rect1,rect2: Rect) : BOOLEAN;
FUNCTION  EmptyRect   (r: Rect) : BOOLEAN;
```

Graphic Operations on Rectangles

```
PROCEDURE FrameRect   (r: Rect);
PROCEDURE PaintRect   (r: Rect);
PROCEDURE EraseRect   (r: Rect);
PROCEDURE InvertRect  (r: Rect);
PROCEDURE FillRect    (r: Rect; pat: Pattern);
```

Graphic Operations on Ovals

```
PROCEDURE FrameOval   (r: Rect);
PROCEDURE PaintOval   (r: Rect);
PROCEDURE EraseOval   (r: Rect);
PROCEDURE InvertOval  (r: Rect);
PROCEDURE FillOval    (r: Rect; pat: Pattern);
```

Graphic Operations on Rounded-Corner Rectangles

```
PROCEDURE FrameRoundRect   (r: Rect; ovalWidth,ovalHeight: INTEGER);
PROCEDURE PaintRoundRect   (r: Rect; ovalWidth,ovalHeight: INTEGER);
PROCEDURE EraseRoundRect   (r: Rect; ovalWidth,ovalHeight: INTEGER);
PROCEDURE InvertRoundRect  (r: Rect; ovalWidth,ovalHeight: INTEGER);
PROCEDURE FillRoundRect    (r: Rect; ovalWidth,ovalHeight: INTEGER; pat:
                            Pattern);
```

Graphic Operations on Arcs and Wedges

```
PROCEDURE FrameArc   (r: Rect; startAngle,arcAngle: INTEGER);
PROCEDURE PaintArc   (r: Rect; startAngle,arcAngle: INTEGER);
PROCEDURE EraseArc   (r: Rect; startAngle,arcAngle: INTEGER);
PROCEDURE InvertArc  (r: Rect; startAngle,arcAngle: INTEGER);
PROCEDURE FillArc    (r: Rect; startAngle,arcAngle: INTEGER; pat:
                      Pattern);
```

Calculations with Regions

```
FUNCTION  NewRgn :       RgnHandle;
PROCEDURE OpenRgn;
PROCEDURE CloseRgn      (dstRgn: RgnHandle);
PROCEDURE DisposeRgn    (rgn: RgnHandle);
PROCEDURE CopyRgn       (srcRgn,dstRgn: RgnHandle);
PROCEDURE SetEmptyRgn   (rgn: RgnHandle);
PROCEDURE SetRectRgn    (rgn: RgnHandle; left,top,right,bottom: INTEGER);
PROCEDURE RectRgn       (rgn: RgnHandle; r: Rect);
PROCEDURE OffsetRgn     (rgn: RgnHandle; dh,dv: INTEGER);
PROCEDURE InsetRgn      (rgn: RgnHandle; dh,dv: INTEGER);
PROCEDURE SectRgn       (srcRgnA,srcRgnB,dstRgn: RgnHandle);
PROCEDURE UnionRgn      (srcRgnA,srcRgnB,dstRgn: RgnHandle);
PROCEDURE DiffRgn       (srcRgnA,srcRgnB,dstRgn: RgnHandle);
PROCEDURE XorRgn        (srcRgnA,srcRgnB,dstRgn: RgnHandle);
FUNCTION  PtInRgn       (pt: Point; rgn: RgnHandle) : BOOLEAN;
FUNCTION  RectInRgn     (r: Rect; rgn: RgnHandle) : BOOLEAN;
FUNCTION  EqualRgn      (rgnA,rgnB: RgnHandle) : BOOLEAN;
FUNCTION  EmptyRgn      (rgn: RgnHandle) : BOOLEAN;
```

Graphic Operations on Regions

```
PROCEDURE FrameRgn   (rgn: RgnHandle);
PROCEDURE PaintRgn   (rgn: RgnHandle);
PROCEDURE EraseRgn   (rgn: RgnHandle);
PROCEDURE InvertRgn  (rgn: RgnHandle);
PROCEDURE FillRgn    (rgn: RgnHandle; pat: Pattern);
```

Bit Transfer Operations

```
PROCEDURE ScrollRect   (r: Rect; dh,dv: INTEGER; updateRgn: RgnHandle);
PROCEDURE CopyBits     (srcBits,dstBits: BitMap; srcRect,dstRect: Rect;
                        mode: INTEGER; maskRgn: RgnHandle);
```

Pictures

```
FUNCTION  OpenPicture   (picFrame: Rect) : PicHandle;
PROCEDURE PicComment    (kind,dataSize: INTEGER; dataHandle: Handle);
PROCEDURE ClosePicture;
PROCEDURE DrawPicture   (myPicture: PicHandle; dstRect: Rect);
PROCEDURE KillPicture   (myPicture: PicHandle);
```

Calculations with Polygons

```
FUNCTION  OpenPoly :    PolyHandle;
PROCEDURE ClosePoly;
PROCEDURE KillPoly     (poly: PolyHandle);
PROCEDURE OffsetPoly   (poly: PolyHandle; dh,dv: INTEGER);
```

Graphic Operations on Polygons

```
PROCEDURE FramePoly    (poly: PolyHandle);
PROCEDURE PaintPoly    (poly: PolyHandle);
PROCEDURE ErasePoly    (poly: PolyHandle);
PROCEDURE InvertPoly   (poly: PolyHandle);
PROCEDURE FillPoly     (poly: PolyHandle; pat: Pattern);
```

Calculations with Points

```
PROCEDURE AddPt           (srcPt: Point; VAR dstPt: Point);
PROCEDURE SubPt           (srcPt: Point; VAR dstPt: Point);
PROCEDURE SetPt           (VAR pt: Point; h,v: INTEGER);
FUNCTION  EqualPt         (pt1,pt2: Point) : BOOLEAN;
PROCEDURE LocalToGlobal   (VAR pt: Point);
PROCEDURE GlobalToLocal   (VAR pt: Point);
```

Miscellaneous Routines

```
FUNCTION  Random :   INTEGER;
FUNCTION  GetPixel  (h,v: INTEGER) : BOOLEAN;
PROCEDURE StuffHex  (thingPtr: Ptr; s: Str255);
PROCEDURE ScalePt   (VAR pt: Point; srcRect,dstRect: Rect);
PROCEDURE MapPt     (VAR pt: Point; srcRect,dstRect: Rect);
PROCEDURE MapRect   (VAR r: Rect; srcRect,dstRect: Rect);
PROCEDURE MapRgn    (rgn: RgnHandle; srcRect,dstRect: Rect);
PROCEDURE MapPoly   (poly: PolyHandle; srcRect,dstRect: Rect);
```

Customizing QuickDraw Operations

```
PROCEDURE SetStdProcs   (VAR procs: QDProcs);
PROCEDURE StdText       (byteCount: INTEGER; textBuf: Ptr; numer,denom:
                         Point);
PROCEDURE StdLine       (newPt: Point);
PROCEDURE StdRect       (verb: GrafVerb; r: Rect);
PROCEDURE StdRRect      (verb: GrafVerb; r: Rect; ovalwidth,ovalHeight:
                         INTEGER);
PROCEDURE StdOval       (verb: GrafVerb; r: Rect);
PROCEDURE StdArc        (verb: GrafVerb; r: Rect; startAngle,arcAngle:
                         INTEGER);
PROCEDURE StdPoly       (verb: GrafVerb; poly: PolyHandle);
PROCEDURE StdRgn        (verb: GrafVerb; rgn: RgnHandle);
PROCEDURE StdBits       (VAR srcBits: BitMap; VAR srcRect,dstRect: Rect;
                         mode: INTEGER; maskRgn: RgnHandle);
PROCEDURE StdComment    (kind,dataSize: INTEGER; dataHandle: Handle);
FUNCTION  StdTxMeas     (byteCount: INTEGER; textAddr: Ptr; VAR numer,
                         denom: Point; VAR info: FontInfo) : INTEGER;
PROCEDURE StdGetPic     (dataPtr: Ptr; byteCount: INTEGER);
PROCEDURE StdPutPic     (dataPtr: Ptr; byteCount: INTEGER);
```

Assembly-Language Information

Constants

```
; Size in bytes of QuickDraw global variables

grafSize        .EQU    206

; Source transfer modes

srcCopy         .EQU    0
srcOr           .EQU    1
srcXor          .EQU    2
srcBic          .EQU    3
notSrcCopy      .EQU    4
notSrcOr        .EQU    5
notSrcXor       .EQU    6
notSrcBic       .EQU    7

; Pattern transfer modes

patCopy         .EQU    8
patOr           .EQU    9
patXor          .EQU    10
patBic          .EQU    11
notPatCopy      .EQU    12
notPatOr        .EQU    13
notPatXor       .EQU    14
notPatBic       .EQU    15
```

```
; Standard colors for ForeColor and BackColor

blackColor      .EQU        33
whiteColor      .EQU        30
redColor        .EQU        205
greenColor      .EQU        341
blueColor       .EQU        409
cyanColor       .EQU        273
magentaColor    .EQU        137
yellowColor     .EQU        69

; Standard picture comments

picLParen       .EQU        0
picRParen       .EQU        1

; Character style

boldBit         .EQU        0
italicBit       .EQU        1
ulineBit        .EQU        2
outlineBit      .EQU        3
shadowBit       .EQU        4
condenseBit     .EQU        5
extendBit       .EQU        6

; Graphic operations

frame           .EQU        0
paint           .EQU        1
erase           .EQU        2
invert          .EQU        3
fill            .EQU        4
```

Point Data Structure

v	Vertical coordinate (word)
h	Horizontal coordinate (word)

Rectangle Data Structure

top	Vertical coordinate of top left corner (word)
left	Horizontal coordinate of top left corner (word)
bottom	Vertical coordinate of bottom right corner (word)
right	Horizontal coordinate of bottom right corner (word)
topLeft	Top left corner (point; long)
botRight	Bottom right corner (point; long)

Region Data Structure

rgnSize	Size in bytes (word)
rgnBBox	Enclosing rectangle (8 bytes)
rgnData	More data if not rectangular

Bit Map Data Structure

baseAddr	Pointer to bit image
rowBytes	Row width (word)
bounds	Boundary rectangle (8 bytes)
bitMapRec	Size in bytes of bit map data structure

Cursor Data Structure

data	Cursor image (32 bytes)
mask	Cursor mask (32 bytes)
hotSpot	Point aligned with mouse (long)
cursRec	Size in bytes of cursor data structure

Structure of QDProcs Record

textProc	Address of text-drawing routine
lineProc	Address of line-drawing routine
rectProc	Address of rectangle-drawing routine
rRectProc	Address of roundRect-drawing routine
ovalProc	Address of oval-drawing routine
arcProc	Address of arc/wedge-drawing routine
polyProc	Address of polygon-drawing routine
rgnProc	Address of region-drawing routine
bitsProc	Address of bit-transfer routine
commentProc	Address of routine for processing picture comments
txMeasProc	Address of routine for measuring text width
getPicProc	Address of picture-retrieval routine
putPicProc	Address of picture-saving routine
qdProcsRec	Size in bytes of QDProcs record

GrafPort Data Structure

device	Font-specific information (word)
portBits	GrafPort's bit map (bitMapRec bytes)
portBounds	Boundary rectangle of grafPort's bit map (8 bytes)
portRect	GrafPort's rectangle (8 bytes)
visRgn	Handle to visible region
clipRgn	Handle to clipping region
bkPat	Background pattern (8 bytes)
fillPat	Fill pattern (8 bytes)
pnLoc	Pen location (point; long)

pnSize	Pen size (point; long)
pnMode	Pen's transfer mode (word)
pnPat	Pen pattern (8 bytes)
pnVis	Pen visibility (word)
txFont	Font number for text (word)
txFace	Text's character style (word)
txMode	Text's transfer mode (word)
txSize	Font size for text (word)
spExtra	Extra space (long)
fgColor	Foreground color (long)
bkColor	Background color (long)
colrBit	Color bit (word)
picSave	Handle to picture being saved
rgnSave	Handle to region being saved
polySave	Handle to polygon being saved
grafProcs	Pointer to QDProcs record
portRec	Size in bytes of grafPort

Picture Data Structure

picSize	Size in bytes (word)
picFrame	Picture frame (rectangle; 8 bytes)
picData	Picture definition data

Polygon Data Structure

polySize	Size in bytes (word)
polyBBox	Enclosing rectangle (8 bytes)
polyPoints	Polygon points

Pen State Data Structure

psLoc	Pen location (point; long)
psSize	Pen size (point; long)
psMode	Pen's transfer mode (word)
psPat	Pen pattern (8 bytes)
psRec	Size in bytes of pen state data structure

Font Information Data Structure

ascent	Ascent (word)
descent	Descent (word)
widMax	Maximum character width (word)
leading	Leading (word)

Special Macro Names

Pascal name	Macro name
SetPortBits	_SetPBits
InvertRect	_InverRect
InvertRoundRect	_InverRoundRect
DisposeRgn	_DisposRgn
SetRectRgn	_SetRecRgn
OffsetRgn	_OfSetRgn
InvertRgn	_InverRgn
ClosePoly	_ClosePgon

Variables

RndSeed	Random number seed (long)