Package Name: quickdraw

Return to: Package List

```
with Interfaces.C; use Interfaces.C;
with Interfaces.C.Extensions; use Interfaces.C.Extensions;
with Types; use Types;
with QuickdrawText; use QuickdrawText;

package Quickdraw is
   --
   --  File:       Quickdraw.h
   --
   --  Contains:   QuickDraw Graphics Interfaces.
   --
   --  Version:    Technology: System 7.5
   --              Package:    Universal Interfaces 2.1 in “MPW Latest” on ETO
#18
   --
   --  Copyright:  © 1984-1995 by Apple Computer, Inc.
   --              All rights reserved.

   --
        invalColReq : constant := -1; --invalid color table request
   --  Transfer modes
        srcCopy : constant := 0;      --the 16 transfer modes
        srcOr : constant := 1;
        srcXor : constant := 2;
        srcBic : constant := 3;
        notSrcCopy : constant := 4;
        notSrcOr : constant := 5;
        notSrcXor : constant := 6;
        notSrcBic : constant := 7;
        patCopy : constant := 8;
        patOr : constant := 9;
        patXor : constant := 10;
        patBic : constant := 11;
        notPatCopy : constant := 12;
        notPatOr : constant := 13;
        notPatXor : constant := 14;
        notPatBic : constant := 15;
   --  Special Text Transfer Mode
        grayishTextOr : constant := 49;
        hilitetransfermode : constant := 50;
   --  Arithmetic transfer modes
        blend : constant := 32;

        addPin : constant := 33;
   --  Transparent mode constant
   --  QuickDraw color separation constants
        addOver : constant := 34;
        subPin : constant := 35;
        addMax : constant := 37;
```

```
      adMax : constant := 37;
      subOver : constant := 38;
      adMin : constant := 39;
      ditherCopy : constant := 64;
-- Transparent mode constant
      transparent: constant := 36;
      italicBit  : constant := 1;
      ulineBit   : constant := 2;
      outlineBit : constant := 3;
      shadowBit  : constant := 4;
      condenseBit: constant := 5;
      extendBit  : constant := 6;
-- QuickDraw color separation constants
      normalBit  : constant := 0; -- normal screen mapping
      inverseBit : constant := 1; -- inverse screen mapping
      redBit     : constant := 4; -- RGB additive mapping
      greenBit   : constant := 3;
      blueBit    : constant := 2;
      cyanBit    : constant := 8; -- CMYBk subtractive mapping
      magentaBit : constant := 7;
      yellowBit  : constant := 6;
      blackBit   : constant := 5;
      blackColor : constant := 33; -- colors expressed in these mappings
      whiteColor : constant := 30;

      redColor   : constant := 205;

subtype GrafVerb is SInt8;

subtype PixelType is SInt8;
type Short_Integer_Vec_16 is array (0 .. 15) of Short_Integer;

subtype Bits16 is Short_Integer_Vec_16;
type Pattern is
   record
      pat : UInt8_Vec_8;
   end record;
type PatPtr is Access Pattern;

type PatHandle is Access PatPtr;

subtype QDByte is SignedByte;

subtype QDErr is Short_Integer;

subtype DeviceLoopFlags is unsigned_long;
type BitMap is
   record
      baseAddr : Ptr;
      rowBytes : Short_Integer;
      bounds : Rect;
   end record;
type BitMapPtr is Access BitMap;

type BitMapHandle is Access BitMapPtr;
```

```
type Cursor is
   record
      data : Bits16;
      mask : Bits16;
      hotSpot : Point;
   end record;
type CursPtr is Access Cursor;

type CursHandle is Access CursPtr;
type PenState is
   record
      pnLoc : Point;
      pnSize : Point;
      pnMode : Short_Integer;
      pnPat : Pattern;

   end record;
type Region is
   record
      rgnSize : Short_Integer; --  size in bytes

      rgnBBox : Rect;          --  enclosing rectangle
   end record;
type RgnPtr is Access Region;

type RgnHandle is Access RgnPtr;
type Picture is
   record
      picSize : Short_Integer;
      picFrame : Rect;
   end record;
type PicPtr is Access Picture;

type PicHandle is Access PicPtr;
type Polygon is
   record
      polySize : Short_Integer;
      polyBBox : Rect;
      polyPoints : Point_Vec_1;
   end record;
type PolyPtr is Access Polygon;

type PolyHandle is Access PolyPtr;
type QDTextProcPtr is Access procedure
   (byteCount : in     Short_Integer;
    textBuf   : in     Ptr;
    numer     : in     Point;
    denom     : in     Point);
type QDLineProcPtr is Access procedure
   (newPt     : in     Point);
type QDRectProcPtr is Access procedure
   (verb      : in     GrafVerb;
    r : access Rect);
type QDRRectProcPtr is Access procedure
   (verb      : in     GrafVerb;
```

```
    r         : access Rect;
    ovalWidth : in     Short_Integer;
    ovalHeight: in     Short_Integer);
type QDOvalProcPtr is Access procedure
   (verb      : in     GrafVerb;
    r         : access Rect);
type QDArcProcPtr is Access procedure
   (verb      : in     GrafVerb;
    r         : access Rect;
    startAngle: in     Short_Integer;
    arcAngle  : in     Short_Integer);
type QDPolyProcPtr is Access procedure
   (verb      : in     GrafVerb;
    poly      : in     PolyHandle);
type QDRgnProcPtr is Access procedure
   (verb      : in     GrafVerb;
    rgn       : in     RgnHandle);
type QDBitsProcPtr is Access procedure
   (srcBits   : access BitMap;
    srcRect   : access Rect;
    dstRect   : access Rect;
    mode      : in     Short_Integer;
    maskRgn   : in     RgnHandle);
type QDCommentProcPtr is Access procedure
   (kind      : in     Short_Integer;
    dataSize  : in     Short_Integer;
    dataHandle: in     Handle);
type QDTxMeasProcPtr is access function
   (byteCount : in     Short_Integer;
    textAddr  : in     Ptr;
    numer     : access Point;
    denom     : access Point;
    info      : access FontInfo)
    return Short_Integer;
type QDGetPicProcPtr is Access procedure
   (dataPtr   : in     Ptr;
    byteCount : in     Short_Integer);
type QDPutPicProcPtr is Access procedure
   (dataPtr   : in     Ptr;
    byteCount : in     Short_Integer);
type QDOpcodeProcPtr is Access procedure
   (fromRect  : access Rect;
    toRect    : access Rect;
    opcode    : in     Short_Integer;
    version   : in     Short_Integer);
type QDJShieldCursorProcPtr is Access procedure
   (left      : in     Short_Integer;
    top       : in     Short_Integer;
    right     : in     Short_Integer;

    bottom    : in     Short_Integer);
subtype QDTextUPP is UniversalProcPtr;
subtype QDLineUPP is UniversalProcPtr;
subtype QDRectUPP is UniversalProcPtr;
subtype QDRRectUPP is UniversalProcPtr;
```

```
   subtype QDOvalUPP is UniversalProcPtr;
   subtype QDArcUPP is UniversalProcPtr;
   subtype QDPolyUPP is UniversalProcPtr;
   subtype QDRgnUPP is UniversalProcPtr;
   subtype QDBitsUPP is UniversalProcPtr;
   subtype QDCommentUPP is UniversalProcPtr;
   subtype QDTxMeasUPP is UniversalProcPtr;
   subtype QDGetPicUPP is UniversalProcPtr;
   subtype QDPutPicUPP is UniversalProcPtr;
   subtype QDOpcodeUPP is UniversalProcPtr;

   subtype QDJShieldCursorUPP is UniversalProcPtr;
   type QDProcs is
      record
         textProc : QDTextUPP;
         lineProc : QDLineUPP;
         rectProc : QDRectUPP;
         rRectProc : QDRRectUPP;
         ovalProc : QDOvalUPP;
         arcProc : QDArcUPP;
         polyProc : QDPolyUPP;
         rgnProc : QDRgnUPP;
         bitsProc : QDBitsUPP;
         commentProc : QDCommentUPP;
         txMeasProc : QDTxMeasUPP;
         getPicProc : QDGetPicUPP;
         putPicProc : QDPutPicUPP;
      end record;

   type QDProcsPtr is Access QDProcs;
   type GrafPort is
      record
         device : Short_Integer;
         portBits : aliased BitMap;
         portRect : aliased Rect;
         visRgn : RgnHandle;
         clipRgn : RgnHandle;
         bkPat : Pattern;
         fillPat : Pattern;
         pnLoc : Point;
         pnSize : Point;
         pnMode : Short_Integer;
         pnPat : Pattern;
         pnVis : Short_Integer;
         txFont : Short_Integer;
         txFace : Style;                --  txFace is unpacked byte but push as
short
         filler : SInt8;
         txMode : Short_Integer;
         txSize : Short_Integer;
         spExtra : Fixed;
         fgColor : Long_Integer;
         bkColor : Long_Integer;
         colrBit : Short_Integer;
         patStretch : Short_Integer;
```

```
         picSave : Handle;
         rgnSave : Handle;
         polySave : Handle;
         grafProcs : QDProcsPtr;
      end record;

   type GrafPtr is Access GrafPort;
   --  This set of definitions "belongs" in Windows.
   --  But, there is a circularity in the headers where Windows includes
Controls and
   --  Controls includes Windows. To break the circle, the information

   --  needed by Controls is moved from Windows to Quickdraw.
   subtype WindowPtr is GrafPtr;
   subtype WindowRef is WindowPtr;

   subtype DragConstraint is UInt16;

   --  Here ends the list of things that "belong" in Windows.
   type RGBColor is
      record
         red : unsigned_short;                    --  magnitude of red component
         green : unsigned_short;                  --  magnitude of green
component
         blue : unsigned_short;                   --  magnitude of blue
component
      end record;
   type RGBColorPtr is Access RGBColor;

   type RGBColorHdl is Access RGBColorPtr;
   type DragGrayRgnProcPtr is Access procedure ;
   type ColorSearchProcPtr is access function
      (rgb      : access RGBColor;
       position : access Long_Integer)
       return Boolean;
   type ColorComplementProcPtr is access function
      (rgb      : access RGBColor)

       return Boolean;
   subtype DragGrayRgnUPP is UniversalProcPtr;
   subtype ColorSearchUPP is UniversalProcPtr;

   subtype ColorComplementUPP is UniversalProcPtr;
   type ColorSpec is
      record
         value : Short_Integer;                   --  index or other value

         rgb : RGBColor;                          --  true color
      end record;
   type ColorSpecPtr is Access ColorSpec;
   type ColorSpec_Vec_1 is array (0 .. 0) of ColorSpec;
   type ColorSpec_Vec_4 is array (0 .. 3) of ColorSpec;

   type ColorSpec_Vec_5 is array (0 .. 4) of ColorSpec;
```

```
   subtype CSpecArray is ColorSpec_Vec_1;
   type xColorSpec is
      record
         value : Short_Integer;                    --  index or other value
         rgb : RGBColor;                           --  true color
         xalpha : Short_Integer;
      end record;
   type xColorSpec_Vec_1 is array (0 .. 0) of xColorSpec;

   type xColorSpecPtr is Access xColorSpec;

   subtype xCSpecArray is xColorSpec_Vec_1;
   type ColorTable is
      record
         ctSeed : Long_Integer;                    --  unique identifier for
table
         ctFlags : Short_Integer;                  --  high bit: 0 = PixMap; 1 =
device
         ctSize : Short_Integer;                   --  number of entries in
CTTable
         ctTable : CSpecArray;                     --  array [0..0] of ColorSpec
      end record;
   type CTabPtr is Access ColorTable;

   type CTabHandle is Access CTabPtr;
   type MatchRec is
      record
         red : unsigned_short;
         green : unsigned_short;
         blue : unsigned_short;
         matchData : Long_Integer;

      end record;
   type PixMap is
      record
         baseAddr : Ptr;                           --  pointer to pixels
         rowBytes : Short_Integer;                 --  offset to next line
         bounds : Rect;                            --  encloses bitmap
         pmVersion : Short_Integer;                --  pixMap version number
         packType : Short_Integer;                 --  defines packing format
         packSize : Long_Integer;                  --  length of pixel date
         hRes : Fixed;                             --  horiz. resolution (ppi)
         vRes : Fixed;                             --  vert. resolution (ppi)
         pixelType : Short_Integer;                --  defines pixel type
         pixelSize : Short_Integer;                --  # bits in pixel
         cmpCount : Short_Integer;                 --  # components in pixel
         cmpSize : Short_Integer;                  --  # bits per component
         planeBytes : Long_Integer;                --  offset to next plan
         pmTable : CTabHandle;                     --  color map for this pixMap
         pmReserved : Long_Integer;                --  for future use. MUST BE 0
      end record;
   type PixMapPtr is Access PixMap;

   type PixMapHandle is Access PixMapPtr;
   type PixPat is
```

```
      record
         patType : Short_Integer;                  --  type of pattern
         patMap : PixMapHandle;                    --  the pattern's pixMap
         patData : Handle;                         --  pixmap's data
         patXData : Handle;                        --  expanded Pattern data
         patXValid : Short_Integer;                --  flags whether expanded
Pattern valid
         patXMap : Handle;                         --  Handle to expanded Pattern
data
         pat1Data : Pattern;                       --  old-Style pattern/RGB
color
      end record;
   type PixPatPtr is Access PixPat;

   type PixPatHandle is Access PixPatPtr;
   type CCrsr is
      record
         crsrType : Short_Integer;                 --  type of cursor
         crsrMap : PixMapHandle;                   --  the cursor's pixmap
         crsrData : Handle;                        --  cursor's data
         crsrXData : Handle;                       --  expanded cursor data
         crsrXValid : Short_Integer;               --  depth of expanded data (0
if none)
         crsrXHandle : Handle;                     --  future use
         crsr1Data : Bits16;                       --  one-bit cursor
         crsrMask : Bits16;                        --  cursor's mask
         crsrHotSpot : Point;                      --  cursor's hotspot
         crsrXTable : Long_Integer;                --  private
         crsrID : Long_Integer;                    --  private
      end record;
   type CCrsrPtr is Access CCrsr;

   type CCrsrHandle is Access CCrsrPtr;
   type Short_Integer_Vec_1 is array (0 .. 0) of Short_Integer;
   type CIcon is
      record
         iconPMap : PixMap;                        --  the icon's pixMap
         iconMask : BitMap;                        --  the icon's mask
         iconBMap : BitMap;                        --  the icon's bitMap
         iconData : Handle;                        --  the icon's data
         iconMaskData : Short_Integer_Vec_1;       --  icon's mask and BitMap
data
      end record;
   type CIconPtr is Access CIcon;

   type CIconHandle is Access CIconPtr;
   type GammaTbl is
      record
         gVersion : Short_Integer;                 --  gamma version number
         gType : Short_Integer;                    --  gamma data type
         gFormulaSize : Short_Integer;             --  Formula data size
         gChanCnt : Short_Integer;                 --  number of channels of data
         gDataCnt : Short_Integer;                 --  number of values/channel
         gDataWidth : Short_Integer;               --  bits/corrected value (data
packed to next larger byte size)
```

```
         gFormulaData : Short_Integer_Vec_1;      --  data for formulas followed
by gamma value
      end record;
   type GammaTbl_Ptr is Access GammaTbl;
   type GammaTblPtr is Access GammaTbl;

   type GammaTblHandle is Access GammaTblPtr;
   type ITab is
      record
         iTabSeed : Long_Integer;                 --  copy of CTSeed from source
CTable
         iTabRes : Short_Integer;                 --  bits/channel resolution of
iTable
         iTTable : Byte_Vec_1;                    --  byte colortable index
values
      end record;
   type ITabPtr is Access ITab;

   type ITabHandle is Access ITabPtr;
   type SProcRec is
      record
         nxtSrch : Handle;                        --  SProcHndl Handle to next
SProcRec
         srchProc : ColorSearchUPP;               --  search procedure proc ptr
      end record;
   type SProcPtr is Access SProcRec;

   type SProcHndl is Access SProcPtr;
   type CProcRec is
      record
         nxtComp : Handle;                        --  CProcHndl Handle to next
CProcRec
         compProc : ColorComplementUPP;           --  complement procedure proc
ptr
      end record;
   type CProcPtr is Access CProcRec;

   type CProcHndl is Access CProcPtr;
   type GDevice is
      record
         gdRefNum : Short_Integer;                --  driver's unit number
         gdID : Short_Integer;                    --  client ID for search procs
         gdType : Short_Integer;                  --  fixed/CLUT/direct
         gdITable : ITabHandle;                   --  Handle to inverse lookup
table
         gdResPref : Short_Integer;               --  preferred resolution of
GDITable
         gdSearchProc : SProcHndl;                --  search proc list head
         gdCompProc : CProcHndl;                  --  complement proc list
         gdFlags : Short_Integer;                 --  grafDevice flags word
         gdPMap : PixMapHandle;                   --  describing pixMap
         gdRefCon : Long_Integer;                 --  reference value
         gdNextGD : Handle;                       --  GDHandle Handle of next
gDevice
```

```
         gdRect : Rect;                          --  device's bounds in global
coordinates
         gdMode : Long_Integer;                  --  device's current mode
         gdCCBytes : Short_Integer;              --  depth of expanded cursor
data
         gdCCDepth : Short_Integer;              --  depth of expanded cursor
data
         gdCCXData : Handle;                     --  Handle to cursor's
expanded data
         gdCCXMask : Handle;                     --  Handle to cursor's
expanded mask
         gdReserved : Long_Integer;              --  future use. MUST BE 0
      end record;
   type GDPtr is Access GDevice;

   type GDHandle is Access GDPtr;
   type GrafVars is
      record
         rgbOpColor : RGBColor;                  --  color for addPin  subPin
and average
         rgbHiliteColor : RGBColor;              --  color for hiliting
         pmFgColor : Handle;                     --  palette Handle for
foreground color
         pmFgIndex : Short_Integer;              --  index value for foreground
         pmBkColor : Handle;                     --  palette Handle for
background color
         pmBkIndex : Short_Integer;              --  index value for background
         pmFlags : Short_Integer;                --  flags for Palette Manager
      end record;
   type GVarPtr is Access GrafVars;

   type GVarHandle is Access GVarPtr;
   type CQDProcs is
      record
         textProc : QDTextUPP;
         lineProc : QDLineUPP;
         rectProc : QDRectUPP;
         rRectProc : QDRRectUPP;
         ovalProc : QDOvalUPP;
         arcProc : QDArcUPP;
         polyProc : QDPolyUPP;
         rgnProc : QDRgnUPP;
         bitsProc : QDBitsUPP;
         commentProc : QDCommentUPP;
         txMeasProc : QDTxMeasUPP;
         getPicProc : QDGetPicUPP;
         putPicProc : QDPutPicUPP;
         opcodeProc : QDOpcodeUPP;               --  fields added to QDProcs
         newProc1 : UniversalProcPtr;
         newProc2 : UniversalProcPtr;
         newProc3 : UniversalProcPtr;
         newProc4 : UniversalProcPtr;
         newProc5 : UniversalProcPtr;
         newProc6 : UniversalProcPtr;
      end record;
```

```
   type CQDProcsPtr is Access CQDProcs;
   type CGrafPort is
      record
         device : Short_Integer;
         portPixMap : aliased PixMapHandle;       --  port's pixel map
         portVersion : Short_Integer;             --  high 2 bits always set
         grafVars : Handle;                       --  Handle to more fields
         chExtra : Short_Integer;                 --  character extra
         pnLocHFrac : Short_Integer;              --  pen fraction
         portRect : Rect;
         visRgn : RgnHandle;
         clipRgn : RgnHandle;
         bkPixPat : PixPatHandle;                 --  background pattern
         rgbFgColor : RGBColor;                   --  RGB components of fg
         rgbBkColor : RGBColor;                   --  RGB components of bg
         pnLoc : Point;
         pnSize : Point;
         pnMode : Short_Integer;
         pnPixPat : PixPatHandle;                 --  pen's pattern
         fillPixPat : PixPatHandle;               --  fill pattern
         pnVis : Short_Integer;
         txFont : Short_Integer;
         txFace : Style;                          --  txFace is unpacked byte
push as short
         filler : SInt8;
         txMode : Short_Integer;
         txSize : Short_Integer;
         spExtra : Fixed;
         fgColor : Long_Integer;
         bkColor : Long_Integer;
         colrBit : Short_Integer;
         patStretch : Short_Integer;
         picSave : Handle;
         rgnSave : Handle;
         polySave : Handle;
         grafProcs : CQDProcsPtr;
      end record;

   type CGrafPtr is Access CGrafPort;

   subtype CWindowPtr is CGrafPtr;
   type ReqListRec is
      record
         reqLSize : Short_Integer;                --  request list size

         reqLData : Short_Integer_Vec_1;          --  request list data

      end record;
   type OpenCPicParams is
      record
         srcRect : Rect;
         hRes : Fixed;
         vRes : Fixed;
         version : Short_Integer;
```

```
      reserved1 : Short_Integer;
      reserved2 : Long_Integer;

   end record;
type CursorImageRec is
   record
      majorVersion : UInt16;
      minorVersion : UInt16;
      cursorPixMap : PixMapHandle;
      cursorBitMask : BitMapHandle;
   end record;

type CursorImagePtr is Access CursorImageRec;
type DeviceLoopDrawingProcPtr is Access procedure
   (depth       : in     Short_Integer;
    deviceFlags : in     Short_Integer;
    targetDevice: in     GDHandle;
    userData    : in     Long_Integer);

subtype DeviceLoopDrawingUPP is UniversalProcPtr;
type char_Vec_76 is array (0 .. 75) of char;
type QDGlobals is
   record
      privates : char_Vec_76;
      randSeed : Long_Integer;
      screenBits : BitMap;
      arrow : Cursor;
      dkGray : Pattern;
      ltGray : Pattern;
      gray : Pattern;
      black : Pattern;
      white : Pattern;
      thePort : aliased GrafPtr;
   end record;
type QDGlobalsPtr is Access QDGlobals;

type QDGlobalsHdl is Access QDGlobalsPtr;
qd                 : QDGlobals;

pragma Import (C, qd, "qd", "qd");
procedure InitGraf
  (globalPtr          : access GrafPtr);

pragma Import (C, InitGraf, "InitGraf", "InitGraf");
procedure OpenPort
  (port               : in     GrafPtr);

pragma Import (C, OpenPort, "OpenPort", "OpenPort");
procedure InitPort
  (port               : in     GrafPtr);

pragma Import (C, InitPort, "InitPort", "InitPort");
procedure ClosePort
  (port               : in     GrafPtr);
```

```
pragma Import (C, ClosePort, "ClosePort", "ClosePort");
procedure SetPort
  (port               : in     GrafPtr);

pragma Import (C, SetPort, "SetPort", "SetPort");
procedure GetPort
  (port               : access GrafPtr);

pragma Import (C, GetPort, "GetPort", "GetPort");
procedure GrafDevice
  (device             : in     Short_Integer);

pragma Import (C, GrafDevice, "GrafDevice", "GrafDevice");
procedure SetPortBits
  (bm                 : access BitMap);

pragma Import (C, SetPortBits, "SetPortBits", "SetPortBits");
procedure PortSize
  (width              : in     Short_Integer;
   height             : in     Short_Integer);

pragma Import (C, PortSize, "PortSize", "PortSize");
procedure MovePortTo
  (leftGlobal         : in     Short_Integer;
   topGlobal          : in     Short_Integer);

pragma Import (C, MovePortTo, "MovePortTo", "MovePortTo");
procedure SetOrigin
  (h                  : in     Short_Integer;
   v                  : in     Short_Integer);

pragma Import (C, SetOrigin, "SetOrigin", "SetOrigin");
procedure SetClip
  (rgn                : in     RgnHandle);

pragma Import (C, SetClip, "SetClip", "SetClip");
procedure GetClip
  (rgn                : in     RgnHandle);

pragma Import (C, GetClip, "GetClip", "GetClip");
procedure ClipRect
  (r                  : access Rect);

pragma Import (C, ClipRect, "ClipRect", "ClipRect");
procedure BackPat
  (pat                : access Pattern);

pragma Import (C, BackPat, "BackPat", "BackPat");
procedure InitCursor;

pragma Import (C, InitCursor, "InitCursor", "InitCursor");
procedure SetCursor
  (crsr               : access Cursor);

pragma Import (C, SetCursor, "SetCursor", "SetCursor");
```

```
procedure HideCursor;

pragma Import (C, HideCursor, "HideCursor", "HideCursor");
procedure ShowCursor;

pragma Import (C, ShowCursor, "ShowCursor", "ShowCursor");
procedure ObscureCursor;

pragma Import (C, ObscureCursor, "ObscureCursor", "ObscureCursor");
procedure HidePen;

pragma Import (C, HidePen, "HidePen", "HidePen");
procedure ShowPen;

pragma Import (C, ShowPen, "ShowPen", "ShowPen");
procedure GetPen
  (pt                 : access Point);

pragma Import (C, GetPen, "GetPen", "GetPen");
procedure GetPenState
  (pnState            : access PenState);

pragma Import (C, GetPenState, "GetPenState", "GetPenState");
procedure SetPenState
  (pnState            : access PenState);

pragma Import (C, SetPenState, "SetPenState", "SetPenState");
procedure PenSize
  (width              : in     Short_Integer;
   height             : in     Short_Integer);

pragma Import (C, PenSize, "PenSize", "PenSize");
procedure PenMode
  (mode               : in     Short_Integer);

pragma Import (C, PenMode, "PenMode", "PenMode");
procedure PenPat
  (pat                : access Pattern);

pragma Import (C, PenPat, "PenPat", "PenPat");
procedure PenNormal;

pragma Import (C, PenNormal, "PenNormal", "PenNormal");
procedure MoveTo
  (h                  : in     Short_Integer;
   v                  : in     Short_Integer);

pragma Import (C, MoveTo, "MoveTo", "MoveTo");
procedure Move
  (dh                 : in     Short_Integer;
   dv                 : in     Short_Integer);

pragma Import (C, Move, "Move", "Move");
procedure LineTo
  (h                  : in     Short_Integer;
```

```
   v                  : in     Short_Integer);

pragma Import (C, LineTo, "LineTo", "LineTo");
procedure Line
  (dh                 : in     Short_Integer;
   dv                 : in     Short_Integer);

pragma Import (C, Line, "Line", "Line");
procedure ForeColor
  (color              : in     Long_Integer);

pragma Import (C, ForeColor, "ForeColor", "ForeColor");
procedure BackColor
  (color              : in     Long_Integer);

pragma Import (C, BackColor, "BackColor", "BackColor");
procedure ColorBit
  (whichBit           : in     Short_Integer);

pragma Import (C, ColorBit, "ColorBit", "ColorBit");
procedure SetRect
  (r                  : access Rect;
   left               : in     Short_Integer;
   top                : in     Short_Integer;
   right              : in     Short_Integer;
   bottom             : in     Short_Integer);

pragma Import (C, SetRect, "SetRect", "SetRect");
procedure OffsetRect
  (r                  : access Rect;
   dh                 : in     Short_Integer;
   dv                 : in     Short_Integer);

pragma Import (C, OffsetRect, "OffsetRect", "OffsetRect");
procedure InsetRect
  (r                  : access Rect;
   dh                 : in     Short_Integer;
   dv                 : in     Short_Integer);

pragma Import (C, InsetRect, "InsetRect", "InsetRect");
function SectRect
  (src1               : access Rect;
   src2               : access Rect;
   dstRect            : access Rect)
   return                      Boolean;

pragma Import (C, SectRect, "SectRect", "SectRect");
procedure UnionRect
  (src1               : access Rect;
   src2               : access Rect;
   dstRect            : access Rect);

pragma Import (C, UnionRect, "UnionRect", "UnionRect");
function EqualRect
  (rect1              : access Rect;
```

```
   rect2              : access Rect)
   return                      Boolean;

pragma Import (C, EqualRect, "EqualRect", "EqualRect");
function EmptyRect
  (r                  : access Rect)
   return                      Boolean;

pragma Import (C, EmptyRect, "EmptyRect", "EmptyRect");
procedure FrameRect
  (r                  : access Rect);

pragma Import (C, FrameRect, "FrameRect", "FrameRect");
procedure PaintRect
  (r                  : access Rect);

pragma Import (C, PaintRect, "PaintRect", "PaintRect");
procedure EraseRect
  (r                  : access Rect);

pragma Import (C, EraseRect, "EraseRect", "EraseRect");
procedure InvertRect
  (r                  : access Rect);

pragma Import (C, InvertRect, "InvertRect", "InvertRect");
procedure FillRect
  (r                  : access Rect;
   pat                : access Pattern);

pragma Import (C, FillRect, "FillRect", "FillRect");
procedure FrameOval
  (r                  : access Rect);

pragma Import (C, FrameOval, "FrameOval", "FrameOval");
procedure PaintOval
  (r                  : access Rect);

pragma Import (C, PaintOval, "PaintOval", "PaintOval");
procedure EraseOval
  (r                  : access Rect);

pragma Import (C, EraseOval, "EraseOval", "EraseOval");
procedure InvertOval
  (r                  : access Rect);

pragma Import (C, InvertOval, "InvertOval", "InvertOval");
procedure FillOval
  (r                  : access Rect;
   pat                : access Pattern);

pragma Import (C, FillOval, "FillOval", "FillOval");
procedure FrameRoundRect
  (r                  : access Rect;
   ovalWidth          : in     Short_Integer;
   ovalHeight         : in     Short_Integer);
```

```
pragma Import (C, FrameRoundRect, "FrameRoundRect", "FrameRoundRect");
procedure PaintRoundRect
  (r                  : access Rect;
   ovalWidth          : in     Short_Integer;
   ovalHeight         : in     Short_Integer);

pragma Import (C, PaintRoundRect, "PaintRoundRect", "PaintRoundRect");
procedure EraseRoundRect
  (r                  : access Rect;
   ovalWidth          : in     Short_Integer;
   ovalHeight         : in     Short_Integer);

pragma Import (C, EraseRoundRect, "EraseRoundRect", "EraseRoundRect");
procedure InvertRoundRect
  (r                  : access Rect;
   ovalWidth          : in     Short_Integer;
   ovalHeight         : in     Short_Integer);

pragma Import (C, InvertRoundRect, "InvertRoundRect", "InvertRoundRect");
procedure FillRoundRect
  (r                  : access Rect;
   ovalWidth          : in     Short_Integer;
   ovalHeight         : in     Short_Integer;
   pat                : access Pattern);

pragma Import (C, FillRoundRect, "FillRoundRect", "FillRoundRect");
procedure FrameArc
  (r                  : access Rect;
   startAngle         : in     Short_Integer;
   arcAngle           : in     Short_Integer);

pragma Import (C, FrameArc, "FrameArc", "FrameArc");
procedure PaintArc
  (r                  : access Rect;
   startAngle         : in     Short_Integer;
   arcAngle           : in     Short_Integer);

pragma Import (C, PaintArc, "PaintArc", "PaintArc");
procedure EraseArc
  (r                  : access Rect;
   startAngle         : in     Short_Integer;
   arcAngle           : in     Short_Integer);

pragma Import (C, EraseArc, "EraseArc", "EraseArc");
procedure InvertArc
  (r                  : access Rect;
   startAngle         : in     Short_Integer;
   arcAngle           : in     Short_Integer);

pragma Import (C, InvertArc, "InvertArc", "InvertArc");
procedure FillArc
  (r                  : access Rect;
   startAngle         : in     Short_Integer;
   arcAngle           : in     Short_Integer;
```

```
   pat                 : access Pattern);
pragma Import (C, FillArc, "FillArc", "FillArc");
function NewRgn
   return                      RgnHandle;

pragma Import (C, NewRgn, "NewRgn", "NewRgn");
procedure OpenRgn;

pragma Import (C, OpenRgn, "OpenRgn", "OpenRgn");
procedure CloseRgn
  (dstRgn              : in     RgnHandle);
pragma Import (C, CloseRgn, "CloseRgn", "CloseRgn");
function BitMapToRegion
  (region              : in     RgnHandle;
   bMap                : access BitMap)
   return                      OSErr;

pragma Import (C, BitMapToRegion, "BitMapToRegion", "BitMapToRegion");
procedure DisposeRgn
  (rgn                 : in     RgnHandle);

pragma Import (C, DisposeRgn, "DisposeRgn", "DisposeRgn");
procedure CopyRgn
  (srcRgn              : in     RgnHandle;
   dstRgn              : in     RgnHandle);

pragma Import (C, CopyRgn, "CopyRgn", "CopyRgn");
procedure SetEmptyRgn
  (rgn                 : in     RgnHandle);

pragma Import (C, SetEmptyRgn, "SetEmptyRgn", "SetEmptyRgn");
procedure SetRectRgn
  (rgn                 : in     RgnHandle;
   left                : in     Short_Integer;
   top                 : in     Short_Integer;
   right               : in     Short_Integer;
   bottom              : in     Short_Integer);

pragma Import (C, SetRectRgn, "SetRectRgn", "SetRectRgn");
procedure RectRgn
  (rgn                 : in     RgnHandle;
   r                   : access Rect);

pragma Import (C, RectRgn, "RectRgn", "RectRgn");
procedure OffsetRgn
  (rgn                 : in     RgnHandle;
   dh                  : in     Short_Integer;
   dv                  : in     Short_Integer);

pragma Import (C, OffsetRgn, "OffsetRgn", "OffsetRgn");
procedure InsetRgn
  (rgn                 : in     RgnHandle;
   dh                  : in     Short_Integer;
   dv                  : in     Short_Integer);
```

```
pragma Import (C, InsetRgn, "InsetRgn", "InsetRgn");
procedure SectRgn
  (srcRgnA            : in     RgnHandle;
   srcRgnB            : in     RgnHandle;
   dstRgn             : in     RgnHandle);

pragma Import (C, SectRgn, "SectRgn", "SectRgn");
procedure UnionRgn
  (srcRgnA            : in     RgnHandle;
   srcRgnB            : in     RgnHandle;
   dstRgn             : in     RgnHandle);

pragma Import (C, UnionRgn, "UnionRgn", "UnionRgn");
procedure DiffRgn
  (srcRgnA            : in     RgnHandle;
   srcRgnB            : in     RgnHandle;
   dstRgn             : in     RgnHandle);

pragma Import (C, DiffRgn, "DiffRgn", "DiffRgn");
procedure XorRgn
  (srcRgnA            : in     RgnHandle;
   srcRgnB            : in     RgnHandle;
   dstRgn             : in     RgnHandle);

pragma Import (C, XorRgn, "XorRgn", "XorRgn");
function RectInRgn
  (r                  : access Rect;
   rgn                : in     RgnHandle)

   return                      Boolean;
pragma Import (C, RectInRgn, "RectInRgn", "RectInRgn");
function EqualRgn
  (rgnA               : in     RgnHandle;
   rgnB               : in     RgnHandle)
   return                      Boolean;

pragma Import (C, EqualRgn, "EqualRgn", "EqualRgn");
function EmptyRgn
  (rgn                : in     RgnHandle)
   return                      Boolean;

pragma Import (C, EmptyRgn, "EmptyRgn", "EmptyRgn");
procedure FrameRgn
  (rgn                : in     RgnHandle);

pragma Import (C, FrameRgn, "FrameRgn", "FrameRgn");
procedure PaintRgn
  (rgn                : in     RgnHandle);

pragma Import (C, PaintRgn, "PaintRgn", "PaintRgn");
procedure EraseRgn
  (rgn                : in     RgnHandle);

pragma Import (C, EraseRgn, "EraseRgn", "EraseRgn");
procedure InvertRgn
```

```
  (rgn                : in     RgnHandle);

pragma Import (C, InvertRgn, "InvertRgn", "InvertRgn");
procedure FillRgn
  (rgn                : in     RgnHandle;
   pat                : access Pattern);

pragma Import (C, FillRgn, "FillRgn", "FillRgn");
procedure ScrollRect
  (r                  : access Rect;
   dh                 : in     Short_Integer;
   dv                 : in     Short_Integer;
   updateRgn          : in     RgnHandle);

pragma Import (C, ScrollRect, "ScrollRect", "ScrollRect");
procedure CopyBits
  (srcBits            : access BitMap;
   dstBits            : access BitMap;
   srcRect            : access Rect;
   dstRect            : access Rect;
   mode               : in     Short_Integer;
   maskRgn            : in     RgnHandle);

pragma Import (C, CopyBits, "CopyBits", "CopyBits");
procedure SeedFill
  (srcPtr             : in     Void_Ptr;
   dstPtr             : in     Void_Ptr;
   srcRow             : in     Short_Integer;
   dstRow             : in     Short_Integer;
   height             : in     Short_Integer;
   words              : in     Short_Integer;
   seedH              : in     Short_Integer;
   seedV              : in     Short_Integer);

pragma Import (C, SeedFill, "SeedFill", "SeedFill");
procedure CalcMask
  (srcPtr             : in     Void_Ptr;
   dstPtr             : in     Void_Ptr;
   srcRow             : in     Short_Integer;
   dstRow             : in     Short_Integer;
   height             : in     Short_Integer;
   words              : in     Short_Integer);

pragma Import (C, CalcMask, "CalcMask", "CalcMask");
procedure CopyMask
  (srcBits            : access BitMap;
   maskBits           : access BitMap;
   dstBits            : access BitMap;
   srcRect            : access Rect;
   maskRect           : access Rect;
   dstRect            : access Rect);

pragma Import (C, CopyMask, "CopyMask", "CopyMask");
function OpenPicture
  (picFrame           : access Rect)
```

```
   return                       PicHandle;

pragma Import (C, OpenPicture, "OpenPicture", "OpenPicture");
procedure PicComment
  (kind               : in     Short_Integer;
   dataSize           : in     Short_Integer;
   dataHandle         : in     Handle);

pragma Import (C, PicComment, "PicComment", "PicComment");
procedure ClosePicture;

pragma Import (C, ClosePicture, "ClosePicture", "ClosePicture");
procedure DrawPicture
  (myPicture          : in     PicHandle;
   dstRect            : access Rect);

pragma Import (C, DrawPicture, "DrawPicture", "DrawPicture");
procedure KillPicture
  (myPicture          : in     PicHandle);

pragma Import (C, KillPicture, "KillPicture", "KillPicture");
function OpenPoly
   return                       PolyHandle;

pragma Import (C, OpenPoly, "OpenPoly", "OpenPoly");
procedure ClosePoly;

pragma Import (C, ClosePoly, "ClosePoly", "ClosePoly");
procedure KillPoly
  (poly               : in     PolyHandle);

pragma Import (C, KillPoly, "KillPoly", "KillPoly");
procedure OffsetPoly
  (poly               : in     PolyHandle;
   dh                 : in     Short_Integer;
   dv                 : in     Short_Integer);

pragma Import (C, OffsetPoly, "OffsetPoly", "OffsetPoly");
procedure FramePoly
  (poly               : in     PolyHandle);

pragma Import (C, FramePoly, "FramePoly", "FramePoly");
procedure PaintPoly
  (poly               : in     PolyHandle);

pragma Import (C, PaintPoly, "PaintPoly", "PaintPoly");
procedure ErasePoly
  (poly               : in     PolyHandle);

pragma Import (C, ErasePoly, "ErasePoly", "ErasePoly");
procedure InvertPoly
  (poly               : in     PolyHandle);

pragma Import (C, InvertPoly, "InvertPoly", "InvertPoly");
procedure FillPoly
```

```
  (poly                : in     PolyHandle;
   pat                 : access Pattern);

pragma Import (C, FillPoly, "FillPoly", "FillPoly");
procedure SetPt
  (pt                  : access Point;
   h                   : in     Short_Integer;
   v                   : in     Short_Integer);

pragma Import (C, SetPt, "SetPt", "SetPt");
procedure LocalToGlobal
  (pt                  : access Point);

pragma Import (C, LocalToGlobal, "LocalToGlobal", "LocalToGlobal");
procedure GlobalToLocal
  (pt                  : access Point);

pragma Import (C, GlobalToLocal, "GlobalToLocal", "GlobalToLocal");
function Random
   return                      Short_Integer;

pragma Import (C, Random, "Random", "Random");
procedure StuffHex
  (thingPtr            : in     Ptr;
   s                   : access Str255);
pragma Import (C, StuffHex, "StuffHex", "StuffHex");
function GetPixel
  (h                   : in     Short_Integer;
   v                   : in     Short_Integer)
   return                      Boolean;

pragma Import (C, GetPixel, "GetPixel", "GetPixel");
procedure ScalePt
  (pt                  : access Point;
   srcRect             : access Rect;
   dstRect             : access Rect);

pragma Import (C, ScalePt, "ScalePt", "ScalePt");
procedure MapPt
  (pt                  : access Point;
   srcRect             : access Rect;
   dstRect             : access Rect);

pragma Import (C, MapPt, "MapPt", "MapPt");
procedure MapRect
  (r                   : access Rect;
   srcRect             : access Rect;
   dstRect             : access Rect);

pragma Import (C, MapRect, "MapRect", "MapRect");
procedure MapRgn
  (rgn                 : in     RgnHandle;
   srcRect             : access Rect;
   dstRect             : access Rect);
```

```
pragma Import (C, MapRgn, "MapRgn", "MapRgn");
procedure MapPoly
  (poly               : in     PolyHandle;
   srcRect            : access Rect;
   dstRect            : access Rect);

pragma Import (C, MapPoly, "MapPoly", "MapPoly");
procedure SetStdProcs
  (procs              : access QDProcs);

pragma Import (C, SetStdProcs, "SetStdProcs", "SetStdProcs");
procedure StdRect
  (verb               : in     GrafVerb;
   r                  : access Rect);

pragma Import (C, StdRect, "StdRect", "StdRect");
procedure StdRRect
  (verb               : in     GrafVerb;
   r                  : access Rect;
   ovalWidth          : in     Short_Integer;
   ovalHeight         : in     Short_Integer);

pragma Import (C, StdRRect, "StdRRect", "StdRRect");
procedure StdOval
  (verb               : in     GrafVerb;
   r                  : access Rect);

pragma Import (C, StdOval, "StdOval", "StdOval");
procedure StdArc
  (verb               : in     GrafVerb;
   r                  : access Rect;
   startAngle         : in     Short_Integer;
   arcAngle           : in     Short_Integer);

pragma Import (C, StdArc, "StdArc", "StdArc");
procedure StdPoly
  (verb               : in     GrafVerb;
   poly               : in     PolyHandle);

pragma Import (C, StdPoly, "StdPoly", "StdPoly");
procedure StdRgn
  (verb               : in     GrafVerb;
   rgn                : in     RgnHandle);

pragma Import (C, StdRgn, "StdRgn", "StdRgn");
procedure StdBits
  (srcBits            : access BitMap;
   srcRect            : access Rect;
   dstRect            : access Rect;
   mode               : in     Short_Integer;
   maskRgn            : in     RgnHandle);

pragma Import (C, StdBits, "StdBits", "StdBits");
procedure StdComment
  (kind               : in     Short_Integer;
```

```
   dataSize           : in     Short_Integer;
   dataHandle         : in     Handle);

pragma Import (C, StdComment, "StdComment", "StdComment");
procedure StdGetPic
  (dataPtr            : in     Void_Ptr;
   byteCount          : in     Short_Integer);

pragma Import (C, StdGetPic, "StdGetPic", "StdGetPic");
procedure StdPutPic
  (dataPtr            : in     Void_Ptr;
   byteCount          : in     Short_Integer);

pragma Import (C, StdPutPic, "StdPutPic", "StdPutPic");
procedure AddPt
  (src                : in     Point;
   dst                : access Point);

pragma Import (C, AddPt, "AddPt", "AddPt");
function EqualPt
  (pt1                : in     Point;
   pt2                : in     Point)
   return                      Boolean;

pragma Import (C, EqualPt, "EqualPt", "EqualPt");
function PtInRect
  (pt                 : in     Point;
   r                  : access Rect)
   return                      Boolean;

pragma Import (C, PtInRect, "PtInRect", "PtInRect");
procedure Pt2Rect
  (pt1                : in     Point;
   pt2                : in     Point;
   dstRect            : access Rect);

pragma Import (C, Pt2Rect, "Pt2Rect", "Pt2Rect");
procedure PtToAngle
  (r                  : access Rect;
   pt                 : in     Point;
   angle              : access Short_Integer);

pragma Import (C, PtToAngle, "PtToAngle", "PtToAngle");
procedure SubPt
  (src                : in     Point;
   dst                : access Point);

pragma Import (C, SubPt, "SubPt", "SubPt");
function PtInRgn
  (pt                 : in     Point;
   rgn                : in     RgnHandle)
   return                      Boolean;

pragma Import (C, PtInRgn, "PtInRgn", "PtInRgn");
procedure StdLine
```

```
  (newPt              : in     Point);

pragma Import (C, StdLine, "StdLine", "StdLine");
procedure OpenCPort
  (port               : in     CGrafPtr);

pragma Import (C, OpenCPort, "OpenCPort", "OpenCPort");
procedure InitCPort
  (port               : in     CGrafPtr);

pragma Import (C, InitCPort, "InitCPort", "InitCPort");
procedure CloseCPort
  (port               : in     CGrafPtr);

pragma Import (C, CloseCPort, "CloseCPort", "CloseCPort");
function NewPixMap
   return                      PixMapHandle;

pragma Import (C, NewPixMap, "NewPixMap", "NewPixMap");
procedure DisposePixMap
  (pm                 : in     PixMapHandle);

pragma Import (C, DisposePixMap, "DisposePixMap", "DisposePixMap");
procedure CopyPixMap
  (srcPM              : in     PixMapHandle;
   dstPM              : in     PixMapHandle);

pragma Import (C, CopyPixMap, "CopyPixMap", "CopyPixMap");
function NewPixPat
   return                      PixPatHandle;

pragma Import (C, NewPixPat, "NewPixPat", "NewPixPat");
procedure DisposePixPat
  (pp                 : in     PixPatHandle);

pragma Import (C, DisposePixPat, "DisposePixPat", "DisposePixPat");
procedure CopyPixPat
  (srcPP              : in     PixPatHandle;
   dstPP              : in     PixPatHandle);

pragma Import (C, CopyPixPat, "CopyPixPat", "CopyPixPat");
procedure PenPixPat
  (pp                 : in     PixPatHandle);

pragma Import (C, PenPixPat, "PenPixPat", "PenPixPat");
procedure BackPixPat
  (pp                 : in     PixPatHandle);

pragma Import (C, BackPixPat, "BackPixPat", "BackPixPat");
function GetPixPat
  (patID              : in     Short_Integer)
   return                      PixPatHandle;

pragma Import (C, GetPixPat, "GetPixPat", "GetPixPat");
procedure MakeRGBPat
```

```
  (pp                 : in     PixPatHandle;
   myColor            : access RGBColor);

pragma Import (C, MakeRGBPat, "MakeRGBPat", "MakeRGBPat");
procedure FillCRect
  (r                  : access Rect;
   pp                 : in     PixPatHandle);

pragma Import (C, FillCRect, "FillCRect", "FillCRect");
procedure FillCOval
  (r                  : access Rect;
   pp                 : in     PixPatHandle);

pragma Import (C, FillCOval, "FillCOval", "FillCOval");
procedure FillCRoundRect
  (r                  : access Rect;
   ovalWidth          : in     Short_Integer;
   ovalHeight         : in     Short_Integer;
   pp                 : in     PixPatHandle);

pragma Import (C, FillCRoundRect, "FillCRoundRect", "FillCRoundRect");
procedure FillCArc
  (r                  : access Rect;
   startAngle         : in     Short_Integer;
   arcAngle           : in     Short_Integer;
   pp                 : in     PixPatHandle);

pragma Import (C, FillCArc, "FillCArc", "FillCArc");
procedure FillCRgn
  (rgn                : in     RgnHandle;
   pp                 : in     PixPatHandle);

pragma Import (C, FillCRgn, "FillCRgn", "FillCRgn");
procedure FillCPoly
  (poly               : in     PolyHandle;
   pp                 : in     PixPatHandle);

pragma Import (C, FillCPoly, "FillCPoly", "FillCPoly");
procedure RGBForeColor
  (color              : access RGBColor);

pragma Import (C, RGBForeColor, "RGBForeColor", "RGBForeColor");
procedure RGBBackColor
  (color              : access RGBColor);

pragma Import (C, RGBBackColor, "RGBBackColor", "RGBBackColor");
procedure SetCPixel
  (h                  : in     Short_Integer;
   v                  : in     Short_Integer;
   cPix               : access RGBColor);

pragma Import (C, SetCPixel, "SetCPixel", "SetCPixel");
procedure SetPortPix
  (pm                 : in     PixMapHandle);
```

```
pragma Import (C, SetPortPix, "SetPortPix", "SetPortPix");
procedure GetCPixel
  (h                  : in     Short_Integer;
   v                  : in     Short_Integer;
   cPix               : access RGBColor);

pragma Import (C, GetCPixel, "GetCPixel", "GetCPixel");
procedure GetForeColor
  (color              : access RGBColor);

pragma Import (C, GetForeColor, "GetForeColor", "GetForeColor");
procedure GetBackColor
  (color              : access RGBColor);

pragma Import (C, GetBackColor, "GetBackColor", "GetBackColor");
procedure SeedCFill
  (srcBits            : access BitMap;
   dstBits            : access BitMap;
   srcRect            : access Rect;
   dstRect            : access Rect;
   seedH              : in     Short_Integer;
   seedV              : in     Short_Integer;
   matchProc          : in     ColorSearchUPP;
   matchData          : in     Long_Integer);

pragma Import (C, SeedCFill, "SeedCFill", "SeedCFill");
procedure CalcCMask
  (srcBits            : access BitMap;
   dstBits            : access BitMap;
   srcRect            : access Rect;
   dstRect            : access Rect;
   seedRGB            : access RGBColor;
   matchProc          : in     ColorSearchUPP;
   matchData          : in     Long_Integer);

pragma Import (C, CalcCMask, "CalcCMask", "CalcCMask");
function OpenCPicture
  (newHeader          : access OpenCPicParams)
   return                     PicHandle;

pragma Import (C, OpenCPicture, "OpenCPicture", "OpenCPicture");
procedure OpColor
  (color              : access RGBColor);

pragma Import (C, OpColor, "OpColor", "OpColor");
procedure HiliteColor
  (color              : access RGBColor);

pragma Import (C, HiliteColor, "HiliteColor", "HiliteColor");
procedure DisposeCTable
  (cTable             : in     CTabHandle);
pragma Import (C, DisposeCTable, "DisposeCTable", "DisposeCTable");
function GetCTable
  (ctID               : in     Short_Integer)
   return                     CTabHandle;
```

```
pragma Import (C, GetCTable, "GetCTable", "GetCTable");
function GetCCursor
  (crsrID             : in     Short_Integer)
   return                      CCrsrHandle;

pragma Import (C, GetCCursor, "GetCCursor", "GetCCursor");
procedure SetCCursor
  (cCrsr              : in     CCrsrHandle);

pragma Import (C, SetCCursor, "SetCCursor", "SetCCursor");
procedure AllocCursor;

pragma Import (C, AllocCursor, "AllocCursor", "AllocCursor");
procedure DisposeCCursor
  (cCrsr              : in     CCrsrHandle);

pragma Import (C, DisposeCCursor, "DisposeCCursor", "DisposeCCursor");
function GetCIcon
  (iconID             : in     Short_Integer)
   return                      CIconHandle;

pragma Import (C, GetCIcon, "GetCIcon", "GetCIcon");
procedure PlotCIcon
  (theRect            : access Rect;
   theIcon            : in     CIconHandle);

pragma Import (C, PlotCIcon, "PlotCIcon", "PlotCIcon");
procedure DisposeCIcon
  (theIcon            : in     CIconHandle);

pragma Import (C, DisposeCIcon, "DisposeCIcon", "DisposeCIcon");
procedure SetStdCProcs
  (procs              : access CQDProcs);

pragma Import (C, SetStdCProcs, "SetStdCProcs", "SetStdCProcs");
function GetMaxDevice
  (globalRect         : access Rect)
   return                      GDHandle;

pragma Import (C, GetMaxDevice, "GetMaxDevice", "GetMaxDevice");
function GetCTSeed
   return                      Long_Integer;

pragma Import (C, GetCTSeed, "GetCTSeed", "GetCTSeed");
function GetDeviceList
   return                      GDHandle;

pragma Import (C, GetDeviceList, "GetDeviceList", "GetDeviceList");
function GetMainDevice
   return                      GDHandle;

pragma Import (C, GetMainDevice, "GetMainDevice", "GetMainDevice");
function GetNextDevice
  (curDevice          : in     GDHandle)
```

```
      return                      GDHandle;

   pragma Import (C, GetNextDevice, "GetNextDevice", "GetNextDevice");
   function TestDeviceAttribute
     (gdh                : in     GDHandle;
      attribute          : in     Short_Integer)
      return                      Boolean;

   pragma Import (C, TestDeviceAttribute, "TestDeviceAttribute",
"TestDeviceAttribute");
   procedure SetDeviceAttribute
     (gdh                : in     GDHandle;
      attribute          : in     Short_Integer;
      value              : in     Boolean);

   pragma Import (C, SetDeviceAttribute, "SetDeviceAttribute",
"SetDeviceAttribute");
   procedure InitGDevice
     (qdRefNum           : in     Short_Integer;
      mode               : in     Long_Integer;
      gdh                : in     GDHandle);

   pragma Import (C, InitGDevice, "InitGDevice", "InitGDevice");
   function NewGDevice
     (refNum             : in     Short_Integer;
      mode               : in     Long_Integer)
      return                      GDHandle;

   pragma Import (C, NewGDevice, "NewGDevice", "NewGDevice");
   procedure DisposeGDevice
     (gdh                : in     GDHandle);

   pragma Import (C, DisposeGDevice, "DisposeGDevice", "DisposeGDevice");
   procedure SetGDevice
     (gd                 : in     GDHandle);

   pragma Import (C, SetGDevice, "SetGDevice", "SetGDevice");
   function GetGDevice
      return                      GDHandle;

   pragma Import (C, GetGDevice, "GetGDevice", "GetGDevice");
   function Color2Index
     (myColor            : access RGBColor)
      return                      Long_Integer;

   pragma Import (C, Color2Index, "Color2Index", "Color2Index");
   procedure Index2Color
     (index              : in     Long_Integer;
      aColor             : access RGBColor);

   pragma Import (C, Index2Color, "Index2Color", "Index2Color");
   procedure InvertColor
     (myColor            : access RGBColor);

   pragma Import (C, InvertColor, "InvertColor", "InvertColor");
```

```
function RealColor
  (color              : access RGBColor)
   return                      Boolean;

pragma Import (C, RealColor, "RealColor", "RealColor");
procedure GetSubTable
  (myColors           : in     CTabHandle;
   iTabRes            : in     Short_Integer;
   targetTbl          : in     CTabHandle);

pragma Import (C, GetSubTable, "GetSubTable", "GetSubTable");
procedure MakeITable
  (cTabH              : in     CTabHandle;
   iTabH              : in     ITabHandle;
   res                : in     Short_Integer);

pragma Import (C, MakeITable, "MakeITable", "MakeITable");
procedure AddSearch
  (searchProc         : in     ColorSearchUPP);

pragma Import (C, AddSearch, "AddSearch", "AddSearch");
procedure AddComp
  (compProc           : in     ColorComplementUPP);

pragma Import (C, AddComp, "AddComp", "AddComp");
procedure DelSearch
  (searchProc         : in     ColorSearchUPP);

pragma Import (C, DelSearch, "DelSearch", "DelSearch");
procedure DelComp
  (compProc           : in     ColorComplementUPP);

pragma Import (C, DelComp, "DelComp", "DelComp");
procedure SetClientID
  (id                 : in     Short_Integer);

pragma Import (C, SetClientID, "SetClientID", "SetClientID");
procedure ProtectEntry
  (index              : in     Short_Integer;
   protect            : in     Boolean);

pragma Import (C, ProtectEntry, "ProtectEntry", "ProtectEntry");
procedure ReserveEntry
  (index              : in     Short_Integer;
   reserve            : in     Boolean);

pragma Import (C, ReserveEntry, "ReserveEntry", "ReserveEntry");
procedure SetEntries
  (start              : in     Short_Integer;
   count              : in     Short_Integer;
   aTable             : in     CSpecArray);

pragma Import (C, SetEntries, "SetEntries", "SetEntries");
procedure SaveEntries
  (srcTable           : in     CTabHandle;
```

```
      resultTable        : in     CTabHandle;
      selection          : access ReqListRec);

   pragma Import (C, SaveEntries, "SaveEntries", "SaveEntries");
   procedure RestoreEntries
     (srcTable           : in     CTabHandle;
      dstTable           : in     CTabHandle;
      selection          : access ReqListRec);

   pragma Import (C, RestoreEntries, "RestoreEntries", "RestoreEntries");
   function QDError
      return                      Short_Integer;

   pragma Import (C, QDError, "QDError", "QDError");
   procedure CopyDeepMask
     (srcBits            : access BitMap;
      maskBits           : access BitMap;
      dstBits            : access BitMap;
      srcRect            : access Rect;
      maskRect           : access Rect;
      dstRect            : access Rect;
      mode               : in     Short_Integer;
      maskRgn            : in     RgnHandle);

   pragma Import (C, CopyDeepMask, "CopyDeepMask", "CopyDeepMask");
   procedure DeviceLoop
     (drawingRgn         : in     RgnHandle;
      drawingProc        : in     DeviceLoopDrawingUPP;
      userData           : in     Long_Integer;
      flags              : in     DeviceLoopFlags);

   pragma Import (C, DeviceLoop, "DeviceLoop", "DeviceLoop");
   function GetMaskTable
      return                      Ptr;

   pragma Import (C, GetMaskTable, "GetMaskTable", "GetMaskTable");
   procedure StuffHex
     (thingPtr           : in     Ptr;

      s                  : in     String);
end Quickdraw;
```