



Tech Info Library

Macintosh: Determining Video Card Status via Software (2 of 2)

Revised: 12/4/89
Security: Everyone

Macintosh: Determining Video Card Status via Software (2 of 2)

=====

Article Created: 2 November 1989
Article Last Reviewed: 14 July 1992
Article Last Updated:

TOPIC -----

What follows is an MPW Assembly listing that demonstrates, for a given GDevice, how to determine the minimum and maximum depths that the device supports:

```
PRINT      OFF
INCLUDE     "Traps.a"
INCLUDE     "ToolEqu.a"
INCLUDE     "QuickEqu.a"
INCLUDE     "SysEqu.a"
INCLUDE     "PackMacs.a"
INCLUDE     "SlotEqu.a"
INCLUDE     "ROMEQu.a"
INCLUDE     "VideoEqu.a"
PRINT      ON
```

DISCUSSION -----

```
GetScreenMinMax      PROC      EXPORT
;-----
; PROCEDURE GetScreenMinMax(whichScreen : gdHandle;
;                               VAR minDepth, minMode,
;                               maxDepth, maxMode : Integer);
;-----
; This nasty little procedure figures out, for a given GDevice, what the
; minimum and maximum depths that the device supports. It does this by
; using the Slot Manager to traverse the sResources that are in ROM on
; the video interface card.

StackFrame      RECORD      {A6Link}, DECR
```

```
ParamSize      EQU      *-8
whichScreen    DS.L      1
pMinDepth      DS.L      1
pMinMode       DS.L      1
pMaxDepth      DS.L      1
pMaxMode       DS.L      1
Return         DS.L      1
A6Link         DS.L      1
spBlk          DS        SpBlock.spBlockSize
slotModesPtr   DS.L      1
nextMode       DS.W      1
LocalSize      EQU      *
               ENDR
```

```
WITH          StackFrame, SpBlock, vpBlock
```

```
LINK          A6, #LocalSize
```

```
MOVE.L        pMinDepth(A6), A0          ;Get ptr to minDepth VAR
MOVE.W        #$7FFF, (A0)              ;Init to MAXINT
MOVE.L        pMaxDepth(A6), A0          ;Get ptr to maxDepth VAR
CLR.W         (A0)                      ;Init to zero
```

```
; We need to convert the GDevice's refNum to its unit number. Then, we
; can look in the unit table for a handle to a NewDCE block. This
; will tell us in which slot the card for this display is.
```

```
MOVE.L        whichScreen(A6), A0        ;Get the gDevice handle
MOVE.L        (A0), A0                  ;Get a ptr to the gDevice
MOVE.W        gdRefNum(A0), D0           ;Get the device's refNum
NOT.W         D0                        ;Get the unit number
ASL.W         #2, D0                    ;Times 4 (SizeOf UTableEntry)
MOVE.L        UTableBase, A0            ;Get a pointer to the Unit Table
MOVE.L        0(A0, D0.W), A0           ;Get the handle to the NewDCE
MOVE.L        (A0), A0                  ;Get ptr to the NewDCE
```

```
; We only want to deal with sResources on the card that are for Apple-
; style video devices. (We only care about the data format; it really
; doesn't matter who made the hardware.) Set up information about the
; type of sResource that we want.
```

```
MOVE.B        dCtlSlot(A0), spBlk+spSlot(A6) ;Put slot of device into parmBlock
CLR.B         spBlk+spID(A6)              ;Start with first sResource
MOVE.W        #catDisplay, spBlk+spCategory(A6) ;Only want Display sResources
MOVE.W        #typVideo, spBlk+spCType(A6)   ;Only want Video sResources
MOVE.W        #drSwApple, spBlk+spDrvrSW(A6) ;Only want Apple-format sResources
MOVE.B        #1, spBlk+spTBMask(A6)        ;Don't care whose hardware
```

```
; Now go and get the first resource that matches our specs.
```

```
LEA           spBlk(A6), A0              ;Pointer to block in A0
_sNextTypesRsrc ;Get sResource that matches
TST.W         D0                          ;Was one found?
```

```
BNE      BadExit                ;Nope. Oh well.
```

```
; We now have a pointer to the sResource List (in spBlk.spsPointer). This
; sResource List has all of the modes that the card will currently support.
```

```
MOVE.L    spBlk+spsPointer(A6), slotModesPtr(A6) ;Save the result
MOVE.B    #128, nextMode(A6)                ;Start with first video mode
```

REPEAT

```
; For Apple-style video data, the first video mode is 128, and they proceed
; sequentially from there, with no gaps.
```

```
MOVE.B    nextMode(A6), spBlk+spID(A6) ;Want entry for nextMode
MOVE.L    slotModesPtr(A6), spBlk+spsPointer(A6) ;Restore ptr to modes sRsrc
LEA       spBlk(A6), A0                ;Ptr to our parameters
_sFindStruct
TST.W     D0                          ;Was it there?
BNE       NoMoreModes                 ;Nope. We're done.
```

```
; spBlk.spsPointer now contains a pointer to the mode information
; structure we just got.
```

```
MOVE.B    #mVidParams, spBlk+spID(A6) ;We want the video parms data
LEA       spBlk(A6), A0                ;Pointer to param block
_sGetBlock ;Get the video parms data
TST.W     D0                          ;It should always be noErr!
BNE.S     BadExit                     ;It's not. Bail out!
```

```
; spBlk.spResult contains a pointer to the video parms data block. Now
; we check to see if we have a video mode that QuickDraw can deal with.
```

```
MOVE.L    spBlk+spResult(A6), A0        ;Get pointer to video parms
MOVE.W    vpCmpCount(A0), D0            ;How many components/pixel?
CMP.W     #1, D0                       ;Can only handle 1
BNE.S     @1                           ;Don't count this mode
MOVE.W    vpPixelSize(A0), D0          ;How many bits/pixel?
CMP.W     vpCmpSize(A0), D0            ;Does it match component size?
BNE.S     @1                           ;Nope. QD can't handle it.
```

```
; D0 now contains a valid pixel depth, and nextMode(A6) contains the
; mode that has this pixel depth. Update the minDepth, maxDepth, and so
; on variables if needed.
```

```
MOVE.L    pMinDepth(A6), A0            ;Ptr to user's minDepth
CMP.W     (A0), D0                     ;Is this mode less than minDepth?
BGE.S     @2                           ;Nope. Don't update.
```

```
; The pixel size in D0 is less than the pixel size that we have stored in
; minDepth so update minDepth and store this mode into minMode.
```

```
MOVE.W    D0, (A0)                    ;Update minDepth
MOVE.L    pMinMode(A6), A0            ;Get pointer to user's minMode
CLR.W     D1                          ;Start with an empty word
```

```
MOVE.B    nextMode(A6), D1      ;Get this mode
MOVE.W    D1, (A0)              ;And save it to user's minMode
```

@2

```
MOVE.L    pMaxDepth(A6), A0     ;Ptr to user's maxDepth
CMP.W     (A0), D0              ;Is this mode > maxDepth?
BLE.S     @1                    ;Nope. Don't update.
```

```
; The pixel size in D0 is greater than the pixel size that we have
; stored in maxDepth, so update maxDepth and store this mode into
; maxMode.
```

```
MOVE.W    D0, (A0)              ;Update maxDepth
MOVE.L    pMaxMode(A6), A0      ;Get pointer to user's maxMode
CLR.W     D1                    ;Start with an empty word
MOVE.B    nextMode(A6), D1      ;Get this mode
MOVE.W    D1, (A0)              ;And save it to user's maxMode
```

@1

```
; Either QuickDraw couldn't handle this video mode, or we're done
; updating the minDepth and maxDepth variables. Now we have to dispose
; of the video parms block we just got.
```

```
MOVE.L    spBlk+spResult(A6), spBlk+spsPointer(A6) ;The pointer to vidParms
LEA       spBlk(A6), A0         ;Pointer to our param block
_sDisposePtr                                ;Release this block
```

```
ADDI.B    #1, nextMode(A6)      ;Try the next mode
BRA.S     REPEAT
```

BadExit

```
; Something went wrong. Set all of the user's variables to zero and
; return.
```

```
MOVE.L    pMinDepth(A6), A0     ;Ptr to user's minDepth
CLR.W     (A0)                  ;Set to zero
MOVE.L    pMinMode(A6), A0      ;Ptr to user's minMode
CLR.W     (A0)                  ;Set to zero
MOVE.L    pMaxDepth(A6), A0     ;Ptr to user's maxDepth
CLR.W     (A0)                  ;Set to zero
MOVE.L    pMaxMode(A6), A0      ;Ptr to user's maxMode
CLR.W     (A0)                  ;Set to zero
```

```
BRA.S     NoMoreModes           ;Standard clean-up
```

NoMoreModes

```
; When we get here, _sFindStruct couldn't find the mode that we were
; for, so there must not be any more. We've looked through all of the
; modes so we're done.
```

```
UNLK      A6                    ;Release locals
MOVE.L    (SP)+, A0             ;Get return address
ADDA.L    #ParamSize, SP       ;Pop input params off stack
```

```
JMP      (A0)          ;And return to caller
DC.B     "GETSCREE"    ;Name of routine for debuggers
```

```
END
```

Copyright 1989 Apple Computer, Inc.

Tech Info Library Article Number:4726