New Technical Notes

Macintosh

Developer Support

Plotting Small Icons

Imaging

Revised by: James Beninghaus October 1989
Written by: James Beninghaus & Dennis Hescox August 1989

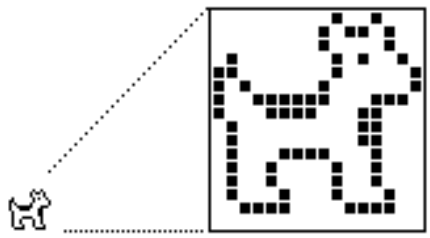
This Technical Note discusses the `'SICN'` resource format and how to plot one in a `GrafPort`.

Changes since August 1989: Corrected errors in the Pascal code and spruced up the rest.

Introduction

Apple first introduced the `'SICN'` resource so that the Script Manager could represent which country specific resources are installed in the system by displaying a small icon in the upper right corner of the menu bar. You can pass a `'SICN'` resource to the Notification Manager or Menu Manager, and they will draw it for you automatically—you should continue to let them do so. However, if you want to draw a small icon in your application's window, then this Note can help.

What does a `'SICN'` look like? Following is a `'SICN'` representation of a dogcow to help answer this question:



`SICN'` **FatBits**

There is reason to believe that this representation is actually a baby dogcow. Due to the protective nature of parent dogcows, young dogcows are rarely seen. This one was spotted during a DTS meeting after it drew attention to itself by crying "moo! woof!". (Note that this dogcow said "moo! woof!" because it was immature; adult dogcows naturally say, "Moof!".)

'SICN' Resource

A 'SICN' resource contains any number of small icon bit images. Each small icon in a 'SICN' list describes a 16 by 16 pixel image and requires 32 bytes of storage. Like an 'ICN#' resource, there is no count of the number of icons stored in a 'SICN'. The following 'SICN' resource, in MPW Rez format, contains two small icons:

```
resource 'SICN' (1984, "clarus") {
      {     /* array: 2 elements */

            $"00 48 00 B4 00 84 40 52 C0 41 A0 81 9F 8E 8F 18"
            $"40 18 40 18 47 88 48 48 48 48 44 44 3C 3C 00 00",

            $"00 48 00 FC 00 FC 40 7E C0 7F E0 FF FF FE FF F8"
            $"7F F8 7F F8 7F F8 78 78 78 78 7C 7C 3C 3C 00 00"
      }
};
```

The Right Tools for the Job

The Macintosh Toolbox interfaces do not describe all the necessary data structures needed to work with 'SICN' resources. As shown in the following example, defining the 'SICN' type as an array of 16 short integers and the handles and pointers to this array type make life much easier.

Pascal

```
TYPE
      SICN        = ARRAY[0 .. 15] of INTEGER;
      SICNList    = ARRAY[0 .. 0] of SICN;
      SICNPtr     = ^SICNList;
      SICNHand    = ^SICNPtr;
```

C

```
typedef    short       SICN[16];
typedef    SICN        *SICNList;
typedef    SICNList    *SICNHand;
```

The Missing Count

The 'SICN' resource does not provide a count to indicate the number of small icons contained within; however, you can easily determine this number by dividing the total size of the resource by the size of a single small icon.

Pascal

```
CONST
      mySICN      = 1984;
VAR
      theSICN     : SICNHand;
      theSize     : LONGINT;
      theCount    : LONGINT;
      theIndex    : LONGINT;

theSICN := SICNHand(GetResource('SICN', mySICN));
IF (theSICN <> NIL) THEN BEGIN
      theSize := GetHandleSize(Handle(theSICN));
      theCount := theSize DIV sizeof(SICN);
END;
```

C

```
#define mySICN         1984

SICNHand   theSICN;
long       theSize;
long       theCount;
long       theIndex;

theSICN = (SICNHand) GetResource('SICN', mySICN);
if (theSICN) {
      theSize = GetHandleSize((Handle)theSICN);
      theCount = theSize / sizeof(SICN);
}
```

The Plot 'SICN's

The example procedure `PlotSICN` draws one small icon of a `'SICN'` resource. It takes the handle from `theSICN` and the position in the list from `theIndex` within the rectangle `theRect` of the current `GrafPort`.

Following is an example call to `PlotSICN` which plots all the small icons in a resource into the same rectangle:

Pascal

```
SetRect(theRect, 0, 0, 16, 16);
FOR theIndex := 0 TO theCount-1 DO
      PlotSICN(theRect, theSICN, theIndex);
```

C

```
SetRect(&theRect, 0, 0, 16, 16);
for (theIndex = 0; theIndex < theCount ; ++theIndex)
      PlotSICN(&theRect, theSICN, theIndex);
```

Because PlotSICN uses `_CopyBits` and `_CopyBits` can move memory, you should lock the handle to the 'SICN' once the resource is loaded. Notice that the `PlotSICN` procedure

dereferences the 'SICN' handle, adds an offset, and copies the resulting value. If the 'SICN' list moves in memory at this time, the bitmap's baseAddr is useless.

To play it safe, PlotSICN saves a copy of the master pointer flags associated with the relocatable block, locks the block with a call to _HLock, and restores the flags after calling _CopyBits. You should **never** examine, set, or clear these flags directly; you should always use the routines which are provided by the Memory Manager and Resource Manager. Note that it is not necessary to check the value of the flag after getting it.

Pascal

```
PROCEDURE PlotSICN(theRect: Rect; theSICN: SICNHand; theIndex : INTEGER);
VAR
      state       : SignedByte;   { we want a chance to restore original state }
      srcBits     : BitMap;       { built up around 'SICN' data so we can
                                        _CopyBits }

BEGIN
      { check the index for a valid value }
      IF (GetHandleSize(Handle(theSICN)) DIV sizeof(SICN)) > theIndex THEN
      BEGIN

          { store the resource's current locked/unlocked condition }
          state := HGetState(Handle(theSICN));

          { lock the resource so it won't move during the _CopyBits call }
          HLock(Handle(theSICN));

          { set up the small icon's bitmap }
          {$PUSH}
          {$R-}                   { turn off range checking }
          srcBits.baseAddr := Ptr(@theSICN^^[theIndex]);
          {$POP}
          srcBits.rowBytes := 2;
          SetRect(srcBits.bounds, 0, 0, 16, 16);

          { draw the small icon in the current grafport }
          CopyBits(srcBits,thePort^.portBits,srcBits.bounds,theRect,srcCopy,NIL);

          { restore the resource's locked/unlocked condition }
          HSetState(Handle(theSICN), state);
      END;
END;
```

C

```
void PlotSICN(Rect *theRect, SICNHand theSICN, long theIndex) {
      auto    char    state;      /* saves original flags of 'SICN' handle */
      auto    BitMap    srcBits;  /* built up around 'SICN' data so we can
                                        _CopyBits */

      /* check the index for a valid value */
      if ((GetHandleSize(Handle(theSICN)) / sizeof(SICN)) > theIndex) {

          /* store the resource's current locked/unlocked condition */
          state = HGetState((Handle)theSICN);
```

```
            /* lock the resource so it won't move during the _CopyBits call */
            HLock((Handle)theSICN);

            /* set up the small icon's bitmap */
            srcBits.baseAddr = (Ptr) (*theSICN)[theIndex];
            srcBits.rowBytes = 2;
            SetRect(&srcBits.bounds, 0, 0, 16, 16);

            /* draw the small icon in the current grafport */
            CopyBits(&srcBits,&(*qd.thePort).portBits,&srcBits.bounds,theRect,srcCopy,nil);

            /* restore the resource's locked/unlocked condition */
            HSetState((Handle) theSICN, state);
        }
}
```

That Was Easy

Now that you've seen it done, it looks pretty easy. With minor modifications, some of the techniques in this Note could also be used to plot a bitmap of any dimension.

Further Reference:

- *Inside Macintosh*, Volume I, QuickDraw
- *Inside Macintosh*, Volume I, Toolbox Utilities
- *Inside Macintosh*, Volume IV, The Memory Manager
- Technical Note M.IM.OffscreenBitMap — Drawing Into an Off-Screen BitMap
- Technical Note M.IM.DrawingIcons — Drawing Icons