New Technical Notes
Macintosh

®

Developer Support

MoreMasters Revisited
Memory M.ME.MoreMasters

Revised by: March 1988
Written by: Jim Friedlander October 1985

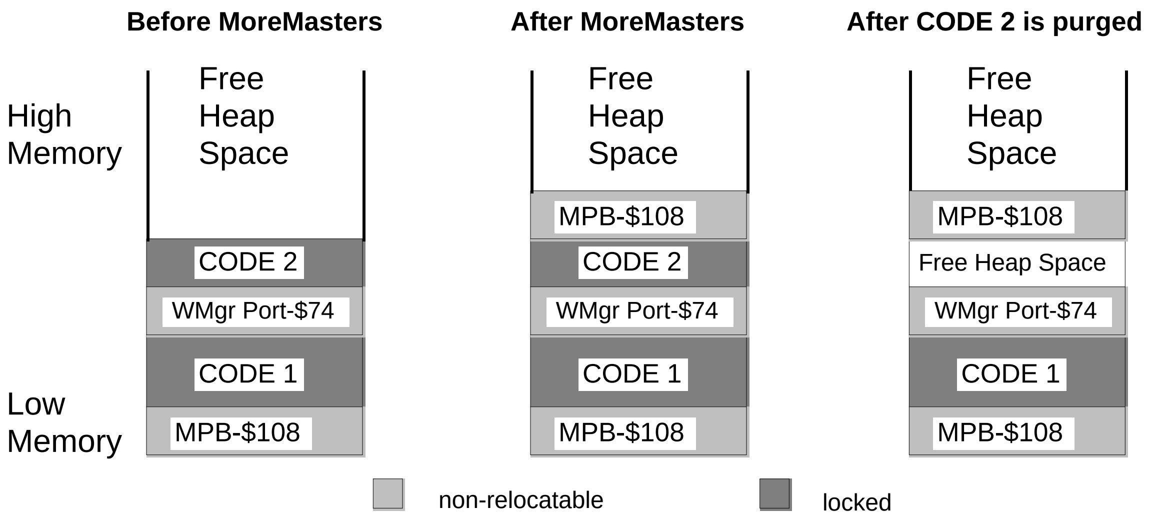
`MoreMasters` should be called from CODE segment 1. The number of master pointers that a program needs can be determined empirically. `MoreMasters` can be tricked into creating the exact number of master pointers desired.

If you ask Macintosh programmers when and how many times `MoreMasters` should be called, you will get a variety of answers, ranging from “four times in the initialization segment” to “once, anywhere.” As you might suspect, the answer is somewhat different from either of these.

`MoreMasters` allocates a block of master pointers in the current heap zone. In the application heap, a block of master pointers consists of 64 master pointers; in the system heap, a block consists of 32 master pointers. Since master pointer blocks are **non-relocatable**, we want to be sure to allocate them early. The system will allocate one master pointer block as your program loads. It’s the first object in the application heap—its size is $108 bytes.

A lot of programmers call `MoreMasters` from an “initialization” segment, but as we shall see, that’s not such a good idea. The problem occurs when we unload our “initialization” segment and it gets purged from memory.

The following diagrams of the application heap illustrate what happens if we call `MoreMasters` from CODE segment 2 (MPB stands for Master Pointer Block):


Before MoreMasters
After MoreMasters
After CODE 2 is purged
High Memory
Low Memory
Free Heap Space
CODE 2
WMgr Port-$74
CODE 1
MPB-$108
Free Heap Space
MPB-$108
CODE 2
WMgr Port-$74
CODE 1
MPB-$108
Free Heap Space
MPB-$108
Free Heap Space
WMgr Port-$74
CODE 1
MPB-$108
non-relocatable
locked

Notice that we now have some heap fragmentation—not serious, but it can be avoided by making all `MoreMasters` calls in CODE segment 1. Because `InitWindows` creates the Window Manager Port (`WMgrPort`), it should also be called from CODE segment 1. Both `MoreMasters` and `InitWindows` should be called before another CODE segment is loaded, or the non-relocatable objects they allocate will be put above the CODE segment and you'll get fragmentation when the CODE segment is purged. If you want to call an initialization segment before calling `MoreMasters` and `InitWindows`, make sure that you unload it before you call either routine.

Now that we know when to call `MoreMasters`, how many times do we call it? The answer depends on your application. If you don't call `MoreMasters` enough times, the system will call it when it needs more master pointers. This can happen at very inconvenient times, causing heap fragmentation. If you call `MoreMasters` too often, you can be wasting valuable memory. This is preferable, however, to allocating too few master pointer blocks!

The number of times you should call `MoreMasters` can be empirically determined. Once your application is almost finished, remove all `MoreMasters` calls. Exercise your application as completely as possible, opening windows, using handles, opening desk accessories, etc. You can then go in with a debugger and see how many times the system called `MoreMasters`. You do that by counting the non-relocatables of size $108. Due to Memory Manager size correction, the master pointer blocks can also have a size of $10C or $110 bytes. You should give yourself about 20% leeway — that is, if the system called `MoreMasters` 10 times for you, you should call it 12 times. If you're more cautious, you might want to call `MoreMasters` 15 times.

Another technique that can save time at initialization is to calculate the number of master pointers you will need, then set the `MoreMast` files of the heap zone header to that number, and then call `MoreMasters` once:

```
PROCEDURE MyMoreMasters(numMastPtrs : INTEGER);

VAR
   oldMoreMast : INTEGER;         {saved value of MoreMast}
   zone        : THz;             {heap zone}

BEGIN
   zone := GetZone;               {get the heap zone}
   WITH zone^ DO BEGIN
      oldMoreMast := MoreMast;    {get the old value of MoreMast}
     MoreMast := numMastPtrs;     {put value we want in zone header}
      MoreMasters;                {allocate the master pointers}
      MoreMast := oldMoreMast;    {restore the old value of MoreMast}
   END;
END;
```

In MPW C:

```
void MyMoreMasters(numMastPtrs)
short numMastPtrs;

{ /* MyMoreMasters */
   short     oldMoreMast;          /* saved value of MoreMast*/
```

```
THz       oZone;                /* heap zone*/

oZone = GetZone();              /* get the heap zone*/
oldMoreMast = oZone->moreMast;  /* get the old value of MoreMast*/
```

```
          oZone->moreMast = numMastPtrs;  /* put the value we want in the
                                             zone header */
          MoreMasters();                  /*allocate the master pointers*/
          oZone->moreMast = oldMoreMast;  /*restore the old value of MoreMast*/
       } /* MyMoreMasters */
```

Further Reference:

- The Memory Manager