**Starting point:**

since the substantial PR #147 by @Elral, I wanted to push his modifications a little bit further to completion.

Thanks to his work, most device objects are now no longer allocated statically, but placed in a common buffer, so (mostly) space is only assigned when required.

This implementation however has still some points that can be improved.

TL;DR: Not all (parts of the) objects are allocated dynamically, plus some space is occupied by the indexing vectors (e.g. encoders[], ledSegments[]…)

Technical details (you can skip this section if not interested in):

* For **inputs**, device objects themselves are fully allocated in the common buffer, but indexing vectors with object pointers still exist. *These vectors must be sized to contain each the maximum allowable number of each device type.*
* For **outputs**, devices are made by the “MF device” object *plus* the actual “worker” object (e.g.: Steppers are made by one “MFStepper” + one “AccelStepper”; LED displays are made by an “MFsegments” plus a “LedControl” etc.).  
  The indexing vectors for output devices contain the *whole* “MF device” object (obviously larger than the pointers alone), while merely the “worker” object is placed in the common buffer.

The improvement consists in removing indexing vectors altogether (using different object management structures) and placing all data in the common buffer.

**The removal of indexing vector allows to use just the exact RAM amount required, without the need for extra provisioning (either of pointers or whole objects).**

**A related perk is that there is no longer a fixed limit for any kind of devices.**

This solution allows a non negligible gain in RAM space (both because of higher efficiency and elimination of unnecessary provisioning), without any penalty in Flash occupation. (Gains can be quantified, but it’s a little hard to

**A sideline story:**

In the meantime, following some comments regarding PR #152, I was trying to help by "de-composing" the individual changes included in the PR with the aim of putting them in the form of individual issues/PR pairs to speed up acceptance.

I started a branch with individual commits targeting following points:

1. Move files to new locations
2. Mobiflight.cpp split to config.\* / commandmessenger.\* / inputHub.\* / outputHub.\* *(see below)*
3. Poll time offsets moved to functions
4. DEBUG constant renamed to DEBUG2MSG
5. MFEEPROM.init() moved into the constructor
6. Removed obsolete Arduino includes for Arduino version < 1.00 *(not included in #152)*
7. Replaced “include guards” in .h files with “#pragma” directives

One thing I did not change yet was the splitting of client functions that use MFxxx device objects (e.g. “Add*Device*()”) and input event handlers / output commands to individual files (e.g. “Encoder.\*” + “MFEncoder.\*”), because that would have been confusing for the sources I was working on.

Instead, I gathered all these functions in two files (pairs), “InputHub.\*” and “OutputHub.\*”. These have become more manageable anyway because the functions contained have become smaller, so I saw no real need to split them anymore.

**Putting it all together:**

After some chats with Ralf, during which I explained the whole situation to him, he kindly agreed that we could possibly proceed this way (pending his final approval):

* replace his PR #152 with a PR from my “detailed” branch;
* waive the file split for edvice client functions;
* on that base, add a new PR for my final modifications described above.