**Rate This Article :**

## PIC32MM bootloader clarification

Jul 21, 2020 · Knowledge

### Article Number
000012093

### Title
PIC32MM bootloader clarification

### Article URL
https://microchipsupport.force.com/s/article/PIC32MM-bootloader-clarification (https://microchipsupport.force.com/s/article/PIC32MM-bootloader-clarification)

### Question
How a bootloader for PIC32MM0256GPM048 can be placed into the Boot Flash memory ?

### Answer
Important things to consider:

- the size of bootloader

- if the bootloader need to program new Config words when updating the application flash

> All PIC32MM products implement 6KB of Boot Flash. While this might sound sufficient for a small bootloader, in practice this is too small and thus poorly suited for such use. Its name is merely a carry over from other PIC32MX/MZ/MK product lines.

>The Debug Executive has a hard claim on addresses [0x9FC00490, 0x9FC00BF0) and hardware has [0x9FC01700, 0x9FC01800) reserved for the Configuration Words + Alternate Configuration Words. This leaves a maximum of [0x9FC00000, 0x9FC00490) and [0x9FC00BF0, 0x9FC01700) available for bootloader code (4000 bytes total), but of this, a handful of fixed hardware reset/nmi and general exception vectors live in the lower region. This makes the true space available fragmented multiple times and difficult to allocate a whole project into, even if the project contains less than 4000 bytes of code.

> Also, if the bootloader needs to ever update the Config words as part of an application update, then the bootloader is further restricted from using addresses [0x9FC01000, 0x9FC01700) as these addresses lie on the same flash erase page as the Config words. This results in only ~1KB of flash left for bootloader code, which is impossibly small for a normal bootloader written in C. A bootloader this small would presumably add a whole extra layer of complexity and need bootstrapping itself (ex: only implement communications code and having the rest of the bootloader obtained from the I2C bus and executed from RAM).

> A simple linker script can't really solve the fragmentation issues because the free-space fragments are so small that even a single function can overflow an available boundary, necessitating manual allocation - a task that is quite project-specific as it depends on the functions that must be implemented or directly/indirectly call in pre-compiled libraries (which themselves have non-fixed sizes due to to optimization levels and other project settings).

### URL Name
PIC32MM-bootloader-clarification