# Making a SAMD21 Bootloader for Hatchling

This document explains what is needed in order to make the Hatchling Bootloader system work. At the end is also an informal list of issues I ran into when adapting the Finch bootloader to Hatchling, and how I got around those issues.

The bootloader system consists of:

1. The SAMD bootloader program
2. The SAMD application program. The application and bootloader programs are merged, and then programmed onto the Hatchling SAMD chip using a JTAG interface at the factory.
3. The micro:bit bootloader, which allows updating of the SAMD application program in the field.

## Make the Bootloader Hex file on the micro:bit for updating the application firmware

I used the repo for V2 bootloaders – [microbit-v2-bootloader](https://github.com/BirdBrainTechnologies/microbit-v2-bootloader), which was initially created for the Finch2 Wiggle Fix code. The main.cpp file is identical, so all I needed to do was update PROG\_DATA.h with the updated application code. Steps:

1. Do a clean and build in HatchlingMain, then copy the HatchlingMain.bin file generated into the “instructions” folder of microbit-v2-bootloader. This file is present in the SAMD HatchlingMain project under debug.

2. Use command tools to run the python script to generate the PROG\_data.h

   py -2 bin2hexHatchling.py

3. Copy and paste the PROG\_data.h file from the instructions folder into the source directory

5. Compile the project with BuildAndCopy.bat – this will generate a MICROBIT.hex file and automatically copy it onto a V2 micro:bit if it is plugged in. It will also remove the build directory and rebuild so as to force a clean and build. The resulting MICROBIT.hex can be renamed and distributed to users who need to update their firmware.

6. Plug the micro:bit into the Hatchling you want to update, then turn it on. It should show ‘E’, followed by ‘P’, followed by a progress bar followed by a check. It will turn off automatically after programming. Plug in a micro:bit V2 with the standard Hatchling MicroBlocks VM program and verify that the updated SAMD firmware is running.

## Make the Combined Factory File that programs the application and bootloader firmware onto the SAMD using JTAG

The SAMD bootloader source code is located in Firmware->HatchlingBootloader->SAMD->HatchlingBootloader

The SAMD application program is located in Firmware->HatchlingMain

To merge the files (so as to update the file that is programmed onto the Hatchling at the factory), do the following:

Take the Hex files from the debug directories of both the bootloader and application files. Use J-Flash tool to merge the both hex files and later store them as a .mot file

​        i. Open J-Flash

​        ii. File -> Open Data File -> pick the bootloader file

​        iii. File->Merge Data File -> pick the application file.

​        iv.  Save the file as .mot file

I made a jlink script, Hatchling\_Script.jlink, that pointed to the location of the Hatchling.mot file – I then copied those into a shorter path directory, C:\Users\TomLauwers\ - because Jlink doesn’t like spaces in paths.

I used a command prompt and went to my SEGGER directory, which was at C:\Program Files\SEGGER\JLink\_V812d on my computer.

I entered the following command, which seemed to program successfully – or at least the Hatchling is behaving normally when not in boot mode

.\JLink -device ATSAMD21G15B -if SWD -speed 2025 -jtagconf -1,-1 -autoconnect 1 -CommanderScript C:\Users\TomLauwers\Hatchling\_Script.jlink

## Make the Bootloader Hex file on the micro:bit for updating the application firmware - issues

I was able to do get the microbit-v2-bootloader repo, but needed to install gcc, cmake, git, and the GNU ARM compiler as per the [install instructions](https://github.com/BirdBrainTechnologies/microbit-v2-bootloader/blob/master/README.md) in the readme. Then I also needed to increase the minimum version of cmake in the file utils/cmake/JSONParser.cmake to version 3.6. Finally, I had to install [ninja.exe](https://github.com/ninja-build/ninja/releases) and put it in a directory on my path. Ninja used to come with CMAKE but no longer does.

I later found that prog\_data.h wasn’t making it into the new hex file because it is static header file. Removing the build directory before compilation fixes this because it forces a clean and build. I updated BuildAndCopy.bat so that it removes this directory, but if someone builds with python build.py they need to delete the build directory manually first.

## Make the Combined Factory File that programs the application and bootloader firmware onto the SAMD using JTAG - issues

1. Issue 1 – the application hex file needs to be set to a higher memory location by adding the following flag to the “miscellaneous” section in the project settings->ARM/GNU Linker area: “-Wl,--section-start=.text=0x800”
2. Issue 2 – the bootloader could not find sam.h and so would not compile. This was caused by changing to a new laptop with a new installation of Microchip studio. I fixed it in ARM/GNU C Compiler->Directories by editing the directory location of the samd21c includes to “../../../../../../../../../../Program Files (x86)/Atmel/Studio/7.0/packs/atmel/SAMD21\_DFP/1.3.395/samd21c/include”
3. Issue 3 – the Hatchling power pin is on pin 14 instead of 19, so I changed this in the code