

Device Specific Instructions for your Application's Linker Script File

STM32CubeIDE - STM32L496x

First thing. Copy the folder "SBLp" from the example application's project folder supplied with the bootloader into your application project folder. You will need to include the .s, .c and .h files it contains into your application build. Add this folder to your tool's include paths for access to the .h file.

If you are not creating and using your own memory regions and sections via the linker script and you have not made edits to your existing project linker script you may use the script file (*.ld) found in the folder (SBLp) in place of the one created by the tools. To do this use the top tool strip of CubeIDE and go to **Project>Properties**. In the properties window left most pane go to **>C/C++ Build > Settings**. Select the **Tool Settings** Tab. Under **MCU GCC Linker** select **General**.

Out to the right of **Linker Script (-T)** click the **Browse** button. Navigate to **your project directory/SBLp** and double click on the file **SBL_App_STM32L496x.ld**. The full file spec (path/filename) should now show in the text window left of the browse button.

While in the settings also set the tool to output a raw binary image. This is required by Flash File Guardian. In the pane in the middle click on **MCU Post build outputs** and of the options to the right check the box **Convert to binary file (-O binary)**. This causes the tool to output a raw binary image of the build as well as the elf file. It is the raw binary image that Flash File Guardian uses to create the secure (.ffg) file.

That's it! Continue to develop your application as normal. It will compile and link as before but located to a higher address (above the bootloader) and include a blank ID Data Block in FLASH memory.

If you are adding the bootloader to an existing project with a linker script that you have made edits to then you may choose 1 of the following 2 options. Edit your existing linker script to include the required memory regions and sections or edit the Driven to Design provided linker script to include your additions. It's up to you.

Here are the required memory regions and code section additions. Text in RED is not a part of the file.

```
/* Memories location and size for a 1MB FLASH device (0x100000) */
MEMORY
{
    RAM        (xrw)  : ORIGIN = 0x20000000, LENGTH = 320K
    VECTAB      (rx)   : ORIGIN = 0x8006800, LENGTH = 0x1B0    < Add this memory region
    ID_DATA     (rx)   : ORIGIN = 0x80069B0, LENGTH = 0x60     < Add this memory region
    FLASH       (rx)   : ORIGIN = 0x8006A10, LENGTH = 0xF95F0   < Edit FLASH region values
}

/* Place sections */
SECTIONS
{
    /* Place the startup code into "FLASH" Rom type memory */
    .isr_vector :
    {
        . = ALIGN(4);
        KEEP(*(.isr_vector)) /* Vector Table */
        . = ALIGN(4);
    } >VECTAB    < Place the .isr_vector section into the new region VECTAB
```

```
/* Place the ID Data block into "FLASH" Rom type memory */  
.iddata :  
{  
    KEEP(*(.iddata))    /* ID Data Block */  
} >ID_DATA             < Place the new section .iddata into ID_DATA
```

The rest of the file should be left as it was. Now your application may be built, secured with Flash File Guardian and uploaded to your board via MCU Flasher.