

Preloader/Bootloader generation

This chapter guides through the tasks which have to be done inside the Intel SoC Embedded Development Suite. As mentioned in page "[Board bring-up overview for TEI0022](#)" this step is for preloader and bootloader generation which should be done in the following three sections:

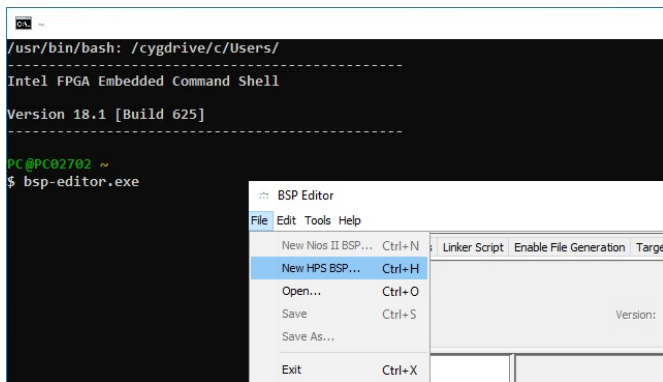
- Preparation
- Preloader/Bootloader generation
- Device Tree generation

The section "Preparation" describes preparing steps which are necessary for the generation of the preloader and the bootloader which is described in section "Preloader/Bootloader generation". After that in section "Device Tree generation" the steps to create the device tree blob is explained.

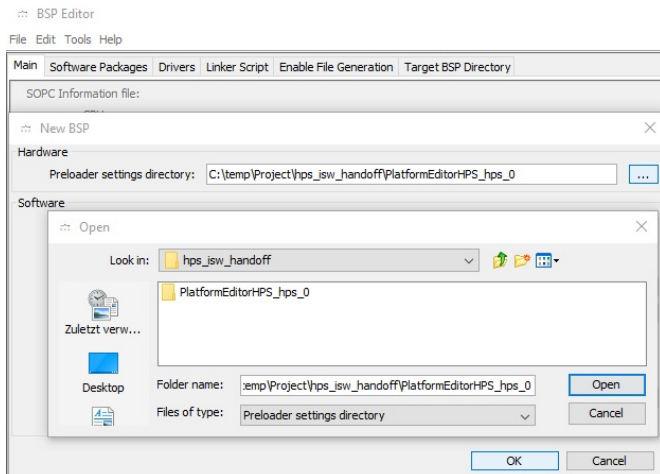
Preparation

While Intel Quartus Prime project compilation, described on page [Intel Quartus Prime Project](#), folder "hps_isw_handoff" is created which is now needed to generate via the bsp-editor further output for preloader and bootloader generation. To do the preparation, follow the following guide:

- Start the SoC EDS Shell as administrator. To do that navigate to **C:\intelFPGA\18.1\embedded**, right click on the file "Embedded_Command_Shell.bat", and select "**Run as administrator**". Click **Yes** in the window "**User Account Control**".
- In the opened shell start the bsp-editor, as visible in the next figure, via: **bsp-editor.exe**

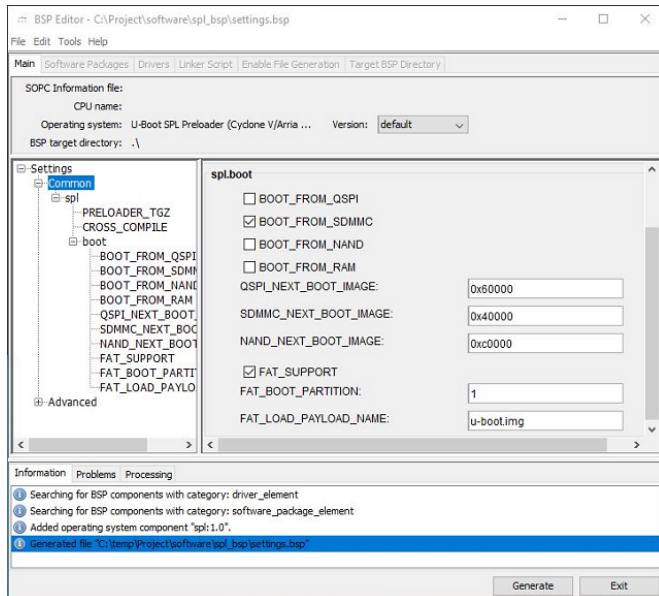


- In the opened bsp-editor select **File New HPS BSP...**
- In the opened **New BSP** dialogue click onto ... and select the **PlatformEditorHPS_hps_0** folder inside the **hps_isw_handoff** folder

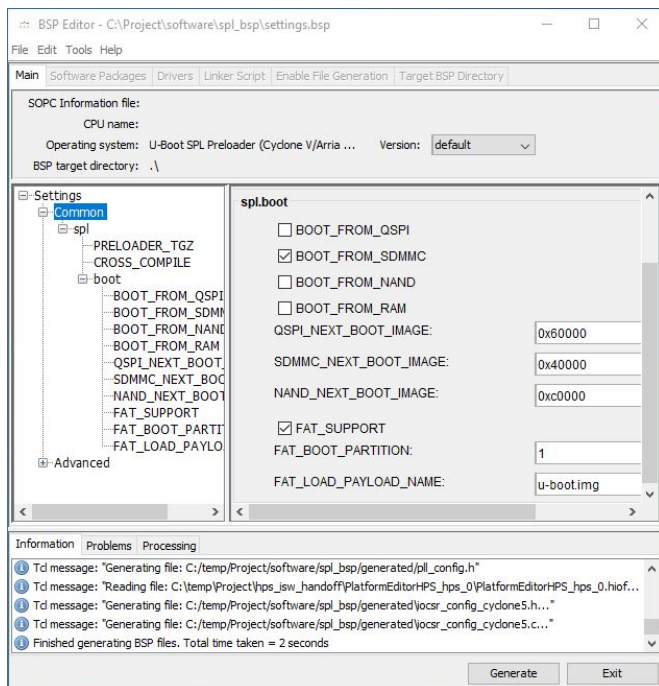


- After that, click **Open** in this dialogue and **OK** in the previous dialogue.

- Now, in the bsp-editor, the preloader should be configured. Select only **BOOT_FROM_SDMMC** as **BOOT_FROM_**-parameter in the right window under the **spl.boot** header.
- Select **FAT_SUPPORT**.
- Select **1** as **FAT_BOOT_PARTITION**.
- Select **u-boot.img** as **FAT_LOAD_PAYLOAD_NAME**.



- Then, generate the output via clicking the **Generate** button.



- After generation, an information like **Finished generation BSP files. Total time taken = ... seconds** is displayed in the information tab. The folder **software** in the project path should now be available.
- Close the bsp-editor.

Preloader/Bootloader generation

After this preparation, it is possible to generate the preloader and the bootloader inside the shell while following the guide:

- Change into folder `.../software/spl_bsp` inside the project folder with the change directory command `cd`. For example: `cd Project/software/spl_bsp`
- Clean the folder via running `/usr/bin/make clean`
- Configure the build process via `/usr/bin/make config` which generates the folder `.../software/spl_bsp/uboot-socfpga`.
- Generate the preloader via `/usr/bin/make` which generates the file `.../software/spl_bsp/preloader-mkpimage.bin`.
- Generate the bootloader via `/usr/bin/make uboot` which generates the image `.../software/spl_bsp/uboot-socfpga/u-boot.img`.
- If the make process ends with an error, try to rerun `/usr/bin/make uboot` until there is no error and the output is generated.

```
PC@PC02702 ~
/usr/bin/bash: /cygdrive/c/Users/
-----
Intel FPGA Embedded Command Shell
Version 18.1 [Build 625]
-----

PC@PC02702 ~
$ bsp-editor.exe

PC@PC02702 ~
$ cd c:

PC@PC02702 ~ /cygdrive/c
$ cd Project/software/spl_bsp/

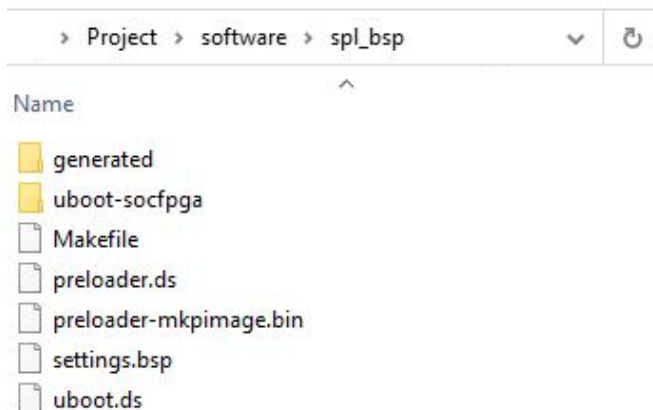
PC@PC02702 ~ /cygdrive/c/Project/software/spl_bsp
$ /usr/bin/make clean
rm -rf preloader-mkpimage.bin uboot-socfpga/.config

PC@PC02702 ~ /cygdrive/c/Project/software/spl_bsp
$ /usr/bin/make config
tar xzf /cygdrive/c/intelFPGA/18.1/embedded/host_tools/altera/preloader/uboot-so
„generated/build.h“ -> „uboot-socfpga/board/altera/socfpga/build.h“

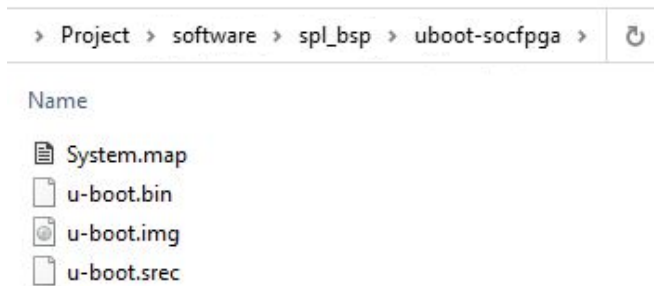
PC@PC02702 ~ /cygdrive/c/Project/software/spl_bsp
$ /usr/bin/make
C:/intelFPGA/18.1/embedded/host_tools/cygwin/bin/make CROSS_COMPILE=arm-altera-e
PGA/18.1/embedded/host_tools/cygwin/bin/cygpath HOSTCC=x86_64-w64-mingw32-gcc HO
boot-socfpga spl/u-boot-spl.bin

PC@PC02702 ~ /cygdrive/c/Project/software/spl_bsp
$ make uboot
C:/intelFPGA/18.1/embedded/host_tools/cygwin/bin/make CROSS_COMPILE=arm-altera-e
in/make CYGPATH=C:/intelFPGA/18.1/embedded/host_tools/cygwin/bin/cygpath HOSTCC=
ingw32-gcc HOSTSTRIP=x86_64-w64-mingw32-strip -C uboot-socfpga spl/u-boot-spl.bi
make[1]: Verzeichnis „/cygdrive/c/temp/Project/software/spl_bsp/uboot-socfpga“ w
. . .
```

After that, the folder `.../software/spl_bsp/` should look like the following figure.



The folder `.../software/spl_bsp/uboot-socfpga` should contain the files shown in the next figure.



Device Tree generation

The device tree generation is a crucial part to tell the linux kernel which hardware has to be handled. To generate the device tree blob follow this guide:

- For device tree generation the Golden **Hardware Reference Design** file **.../intelFPGA/18.1/embedded/examples/hardware/cv_soc_devkit_ghrd/hps_common_board_info.xml** is needed. Therefore, copy this file into the project folder where the **software** folder, the **output_files** folder, ... are. This file contains information regarding the board which can be adapted, if necessary.
- Generate the device tree via the shell command: **sopc2dts --input <Project Name>.sopcinfo --output socfpga.dtb --type dtb --board hps_common_board_info.xml --bridge-removal all --clocks**
- The output in the following listing can be ignored.

Device Tree Generation

```
$ socp2dts.exe --input PlatformEditorHPS.sopcinfo --output DTBsocfpga.dts --type dts --board
hps_common_board_info.xml --bridge-removal all --clocks
MasterIF socp2dts.lib.components.Interface@76fb509a slaveIF null
MasterIF socp2dts.lib.components.Interface@76fb509a slaveIF null
DTAppend: Unable to find parent, null, for #address-cells. Adding to root
DTAppend: Unable to find parent, null, for #size-cells. Adding to root
DTAppend: Unable to find parent, null, for reg. Adding to root
DTAppend: Unable to find parent, null, for spi-max-frequency. Adding to root
DTAppend: Unable to find parent, null, for m25p,fast-read. Adding to root
DTAppend: Unable to find parent, null, for page-size. Adding to root
DTAppend: Unable to find parent, null, for block-size. Adding to root
DTAppend: Unable to find parent, null, for tshsl-ns. Adding to root
DTAppend: Unable to find parent, null, for tsd2d-ns. Adding to root
DTAppend: Unable to find parent, null, for tchsh-ns. Adding to root
DTAppend: Unable to find parent, null, for tslch-ns. Adding to root
DTAppend: Unable to find parent, null, for cdns,page-size. Adding to root
DTAppend: Unable to find parent, null, for cdns,block-size. Adding to root
DTAppend: Unable to find parent, null, for cdns,read-delay. Adding to root
DTAppend: Unable to find parent, null, for cdns,tshsl-ns. Adding to root
DTAppend: Unable to find parent, null, for cdns,tsd2d-ns. Adding to root
DTAppend: Unable to find parent, null, for cdns,tchsh-ns. Adding to root
DTAppend: Unable to find parent, null, for cdns,tslch-ns. Adding to root
```