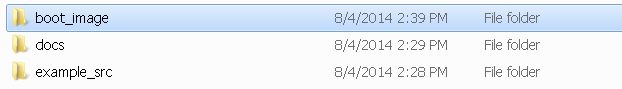
NATIVE SPI BOOT Example on EVMC6678:

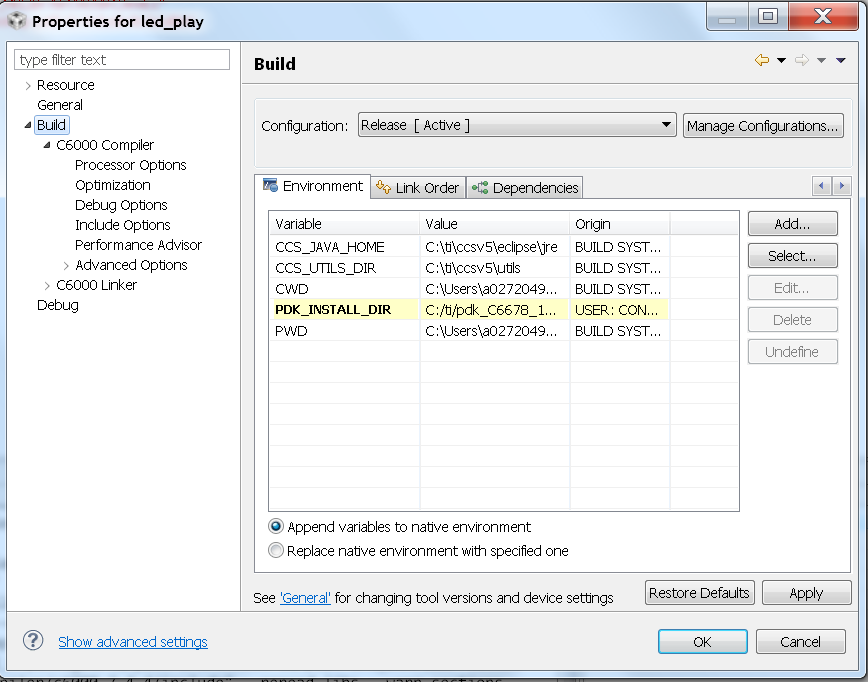
**Directory structure**:



* Example\_src : Contains the CCS project to build led\_play example to blink LEDs on the EVM.
* Boot\_image: Folder contains files required to create the SPI boot example.
  + Pre-built binaries: this contains pre-built boot images and intermediate files that we need for creation of the boot image.
  + SPI\_bootloader: Folder contains bat files that automates the boot image creation. This folder has dependency on the MCSDK.
* Docs: This folder provides documentation for the native SPI boot example for C6678 EVM.

**Building example application binary:**

1. Open CCS Editor and import the led\_play CCS project found under example\_src directory in the workspace.
2. Right click on the project and Open Project properties. Under Build or C/C++ Build (may vary based on version of CCS), set the PDK\_INSTALL\_DIR variable in the Environment.



1. Right click on the project in the Editor view and Rebuild the project in your CCS environment.
2. Load and run the led\_play.out file on the EVMC6678 to ensure that the rebuilt example runs on your EVM.

Note: Use appropriate GEL file to perform device initialization.

1. Copy the led\_play.out binary to the boot\_image/SPI\_bootloader folder.

**Boot Image creation**:

* Open the file spiboot.bat and provide the path to the utils directory in the IBL source under MCSDK installation. The bat file is an automated way to create the boot image and calls into boot utilities in the order provided in the bat file to create the boot image.
* The spiboot.bat file can be run in the Windows terminal. Follow the steps provided below to create the boot image.
  + Set the compiler path in the PATH environment variable by executing:
    - set PATH=C:\ti\ccsv5\tools\compile\c6000\_x.x.x\bin
  + Execute the spiboot.bat file by executing the command
    - spiboot.bat
* Example boot image creation log:

*C:\KeystoneI\_bootloader\_workshop\boot\_image\SPI\_Bootloader>spiboot.bat*

*C:\KeystoneI\_bootloader\_workshop\boot\_image\SPI\_Bootloader>set IBL\_UTIL=C:\ti\mcsdk\_2\_01\_02\_06\tools\boot\_loader\ibl\src\util*

*C:\KeystoneI\_bootloader\_workshop\boot\_image\SPI\_Bootloader>hex6x led\_play.rmd*

*Translating to ASCII-Hex format...*

*"C:\KeystoneI\_bootloader\_workshop\boot\_image\SPI\_Bootloader\led\_p*

*lay.out" ==> .text (BOOT LOAD)*

*"C:\KeystoneI\_bootloader\_workshop\boot\_image\SPI\_Bootloader\led\_p*

*lay.out" ==> .cinit (BOOT LOAD)*

*"C:\KeystoneI\_bootloader\_workshop\boot\_image\SPI\_Bootloader\led\_p*

*lay.out" ==> .const (BOOT LOAD)*

*"C:\KeystoneI\_bootloader\_workshop\boot\_image\SPI\_Bootloader\led\_p*

*lay.out" ==> .switch (BOOT LOAD)*

*C:\KeystoneI\_bootloader\_workshop\boot\_image\SPI\_Bootloader>C:\ti\mcsdk\_2\_01\_02\_06\tools\boot\_loader\ibl\src\util\btoccs\b2i2c led\_play.btbl led\_play*

*.btbl.i2c*

*C:\KeystoneI\_bootloader\_workshop\boot\_image\SPI\_Bootloader>C:\ti\mcsdk\_2\_01\_02\_06\tools\boot\_loader\ibl\src\util\btoccs\b2ccs led\_play.btbl.i2c led\_*

*play.i2c.ccs*

*C:\KeystoneI\_bootloader\_workshop\boot\_image\SPI\_Bootloader>C:\ti\mcsdk\_2\_01\_02\_06\tools\boot\_loader\ibl\src\util\romparse\romparse nysh.spi.map*

*C:\KeystoneI\_bootloader\_workshop\boot\_image\SPI\_Bootloader>copy i2crom.ccs spirom\_le.dat*

*1 file(s) copied.*

*C:\KeystoneI\_bootloader\_workshop\boot\_image\SPI\_Bootloader>byteswapccs i2crom.ccs spirom\_le\_swap.dat*

*C:\KeystoneI\_bootloader\_workshop\boot\_image\SPI\_Bootloader>copy spirom\_le\_swap.dat app.dat*

*1 file(s) copied.*

*C:\KeystoneI\_bootloader\_workshop\boot\_image\SPI\_Bootloader>C:\ti\mcsdk\_2\_01\_02\_06\tools\boot\_loader\ibl\src\util\btoccs\ccs2bin i2crom.ccs spirom\_le.bin*

*C:\KeystoneI\_bootloader\_workshop\boot\_image\SPI\_Bootloader>C:\ti\mcsdk\_2\_01\_02\_06\tools\boot\_loader\ibl\src\util\btoccs\ccs2bin -swap i2crom.ccs spirom\_le.swap.bin*

**Boot Image creation explained:**

1. The RBL expects the image flashed on the SPI flash to be in Boot Table Format. The led\_play example application Code has to be first converted into a Boot Table Format, using the hex6x utility present in CCS installation folder. By default it should be present in the following folder (C:\ti\ccsv5\tools\compiler\c6000\_7.4.2\bin). The hex6x utility expects an rmd file in which you provide path to the application binary and a format in which the boot table is expected. The documentation for hex6x utility is provided in the TMS320C6000 Assembly Language Tools documentation that is part of the compiler documentation. The hex6x utility reads the sections in the application binary and creates a flat binary in boot Table format that allows the ROM to interpret and load the sections of the application binary. The RMD file contains, few of the following information :-

a) The Application.out file that has to be flashed.

b) –a for the output hex format in ASCII

c) –e the entry point for the address, i.e \_c\_init00

d) Output file that contains the application.out in boottable format.

e) Memory sections with the MEM and ROW WIDTH.

Note: All the steps following the boot table creation are post processing steps to either add boot parameter table to change default boot settings or to format the boot image for the specific boot media.

1. From the generated output in previous step which is in the boot table format convert it into the i2c/spi format by passing through the b2i2c. i.e. The byte-aligned boot table is then divided into 0x80 byte blocks and appended with length and checksum to adhere to the format required by the RBL, this is generated by passing through the b2i2c utility. The b2i2c utility is part of the MCSDK installation and present in the following folder. mcsdk\_2\_01\_02\_06\tools\boot\_loader\ibl\src\util\btoccs
2. The output of the previous step, i2c/spi based format of the application table convert it into the CCS acceptable .dat format using b2ccs utility present in the mscdk\tools\boot\_loader\ibl\src\util\btoccs.

Run the utility from the command prompt as below :-

mcsdk\_2\_01\_02\_06\tools\boot\_loader\ibl\src\util\btoccs>b2ccs.exe led\_play.btbl.i2c led\_play.i2c.ccs

1. Using the ROM PARSE utility present in mcsdk/tools/boot\_loader/ibl/src/util/romparse to merge the boot table and boot parameter table. The \*.map file which acts as the input to the romparse utility contains the following information.

a) The Input Application file that has to be flashed into the NOR in CCS acceptable dat format.

b) Boot configurations for SPI boot mode , containing the configurations details. Example in the boot\_image/SPI\_Bootloader folder.

**Important Note: The output i2crom.ccs will be generated using the ROM PARSE utility which hardcodes the I2C address into the parameter table appended to the boot table. In the output i2crom.ccs please change the 51 marked in the below into 00.**

****

5. The output of the above format has to be converted into the big endian format using the byte swap application present part of the document(RBL always works in BIG ENDIAN MODE). The byte swap application has to be used if the end target is little endian and the original application that has to be flashed is compiled in LE mode. Byteswapccs.c has been attached part of the mail and can be used to convert the application from little endian to big endian format.

Create an EXE from byteswapccs.c and run it with the input file as the i2crom.ccs genrated at previous step and output as the app.dat that will be flashed to the NOR.

**Note: If you didn`t change the boot media address in the previous step you can change it in the app.dat. In case you decide to use the bat file to generate the boot image, you will need to ensure that the I2C address has been changed to 00 in your app.dat.**

**Flashing the boot image to EVM SPI NOR flash:**

1. Boot EVM in NO BOOT mode. Hardware switch settings for C6678 EVM have been mentioned on the wiki article below:

[**http://processors.wiki.ti.com/index.php/TMDXEVM6678L\_EVM\_Hardware\_Setup**](http://processors.wiki.ti.com/index.php/TMDXEVM6678L_EVM_Hardware_Setup)

1. The output at this stage is a .dat file which has to flashed into the NOR using the CCS NOR writer. Provided part of the mcsdk folder. mcsdk\_2\_01\_02\_06\tools\writer\nor\evmc6678l

**The EVM uses 24 bit NOR connected on CHIP select 0.** While flashing using the NOR writer use the **.dat as is and don’t convert the .dat into a bin file**. Rebuild then Nor write present in the MCSDK folder , with the following change to the nor\_writer\_input.txt , replace the file name from app.bin to the file name(app.dat) that was generated in step 5 of previous section.

1. Refer to the ReadMe for the Nor writer to flash the app.dat to the EVM SPI flash.
2. Once the NOR is flashed correctly . Boot the EVM in the SPI boot mode with following swtich settings(Refer link provided in Step 1) to see the LEDS on the EVM blink after booting from the flash.