# Booting C6657 directly from SPI NOR flash and enabling DDR initialization using RBL

## Description:

The example aims to demonstrate direct booting of C6657 from the SPI boot instead of using secondary bootloaders like IBL or SBL used in the standard MCSDK/Processor SDK offering for these devices.

The example provided here, demonstrates how a simple application code can be converted into a boot image that can be directly flashed and booted from the NOR flash connected to SPI0 on the EVM. The example also demonstrates appending a boot parameter table to customize peripheral boot speeds and PLL initialization done by the RBL. It also aims to demonstrate configuring the external memory using DDR configuration table

## Software requirements:

* Code composer Studio
* MCSDK/Processor SDK RTOS

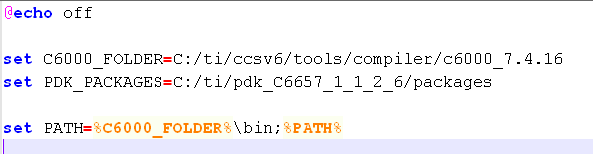
## Folder included in the example:

* src: Folder contains Source files to build the application code.
* build: folder contains files to build the boot image using Windows batch file.
* bin: Folder contains prebuilt binaries for flashing utility and boot image.
* docs: contains documentation for the boot example.

## Steps to reproduce and validate the direct SPI boot on C6657 EVM:

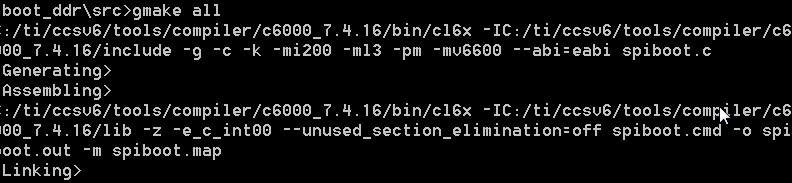
**Step 1**: Setup C6000 compiler path in the file setup.bat and run the bat file from windows command line.

* setup.bat



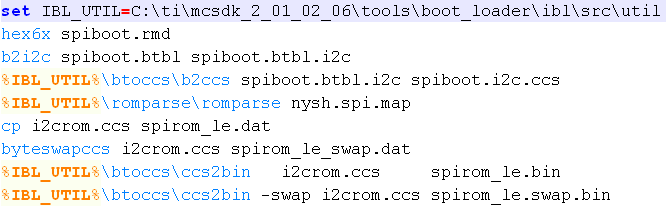
Step2: Change directory to src and build the application binary:

* gmake clean
* gmake all
* **Note:** It requires gmake in the Windows

****

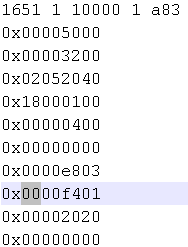
**Step3**: Copy the spiboot.out to build folder

**Step4**: Set IBL\_UTIL path in the spiboot.bat file to the IBL utils folder from the MCSDK/Processor SDK RTOS for C6657.



**Step 5**: Run spiboot.bat file to generate the boot image spirom\_le\_swap.dat.

**Important Note: The romparse utility that is part of the SDK is designed to add boot address 0x51 in the boot parameter table appended to the boot image. Ensure that the value at 0x1F offset in the image is 00 and not 0x51**

****

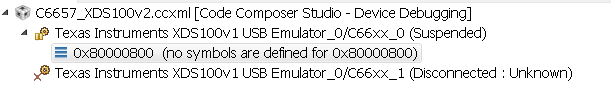
**Step 6**: Flash the boot image to C6657 EVM using norwriter.out in the bin/flashwriter folder using instructions provided in the ReadMe.txt.

**Step 7**: Set boot switches on the EVM to SPI boot

SW3 {pin 0 - pin8) = 10110000 (0-ON, 1-OFF)

SW5(pin0 - pin8) = 00100000

**Step 8**: Power up the EVM and connect to Core0 using CCS, the DDR should be configured by the boot image. To test, connect to GEM core 0 and navigate to address 0x80000000 in the memory browser in CCS.



## Implementation details:

### Creating the boot image with boot parameter table and DDR configuration table:

Following steps are involved in creation of the boot image:

* Create boot table(flat binary) image using hex6x compiler utility
* Appending boot parameter table to the boot table.
* Append DDR configuration table to the boot image.

### Creating boot table image using Hex6x:

The RBL expects the image flashed on the SPI flash to be in Boot Table Format. The spiboot example application Code has to be first converted into a Boot Table Format, using the hex6x utility present in CCS installation folder. (…\ccsv5\tools\compiler\c6000\_7.4.2\bin) (Or a different version of the compiler). 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:-

1. The Application.out file that has to be flashed.
2. –a for the output hex format in ASCII
3. –e the entry point for the address, i.e. \_c\_init00
4. Output file that contains the application.out in boottable format.
5. Memory sections with the MEM and ROW WIDTH
6. Create a new directory c:\temp
7. Copy the out file from the project to the temp directory that you just created. Note, you can copy the out file from the debug directory of the project
8. Copy hex6x from the bin directory (…\ccsv6\tools\compiler\c6000\_7.x.xx\bin) to the temp directory
9. Open a cmd window and cd it to the temp directory
10. Create the rmd file spiboot.rmd using notepad or any other editor as follows:

***spiboot.out***

***-a***

***-boot***

***-e \_c\_int00***

***ROMS***

***{***

***ROM1: org = 0x00000400, length = 0x100000, memwidth = 32, romwidth = 32***

***files = {spiboot.btbl}***

***}***

1. Run hex6x with spiboot.rmd “hex6x spiboot.rmd”
2. The following is a screen shot of the hex6x run:

### Attaching boot parameter table to the boot image:

The SPI boot parameter table for C6657 is defined in the device datasheet or can also be found in the file tiboot\_c6657.h defined in the IBL package of the MCSDK/PRSDK for C6657. The table is used to change the default SPI boot behavior of the device. For Eg: The device boots SPI with PLL in bypass mode so by providing a PLL configuration, you can force the RBL to boot with PLLs configured to full speed.

The romparse utility provided in the IBL utilities in the MCSDK\PRSDK is a TI supported tool for Windows host that can be used to appeand the boot parameter table to the boot table image of the application. The romparse uses a .map file in order to configure boot parameter table that gets appended to the boot image. A sample .map file(nysh.spi.map) is provided in the build folder of the package.

For more details regarding, how each of the boot parameters are configured, the romparse utility is provided in source.

### DDR Configuration tables defined by the ROM bootloader:

The RBL on Keystone I devices like C6657 allows users to initialize DDR using the bootROM by appending a DDR configuration table to the boot image and loading it at a predetermined location in internal memory at the time of Boot. The RBL periodically checks from the DDR configuration table and will initiate the DDR initialization sequence when non-zero values are found in the structure defined in this region.

The DDR configuration table location defined by Boot ROM:

C6678: 0x00873500

C6657: 0x008FFD20

The DDR configuration table for C6678 and C6657 are different due to the difference in ROM implementation. The major difference being the C6657 DDR configuration table includes additional DDR Leveling parameters while C6678 does not. Also, the standard DDR initialization sequence was modified after the ROM bootloader code was finalized hence, we recommend users of C6678 to use a secondary bootloader or application code to initialize the DDR instead of using the RBL.

**DDR configuration table for C6657:**

The ROM code creates a table of DDR configuration parameters in the local L2 at a fixed address (section 6.19.1) and initializes it to zero. While the ROM boot loader is active, the contents of the table are checked after every boot table section is complete. If the boot ROM finds that the enable bitmap is non-zero the DDR is configured. This allows a single boot table record to configure the DDR table, which can be followed by boot sections that reside in DDR.

|  |  |  |
| --- | --- | --- |
| DDR Configuration | | |
| Byte offset | Name | Description |
| 0 | Enable bitmap MSW | Bits 31:0 of the PLL/EMIF enable bitmap. Bit 0 corresponds to the PLL config, Bit 1 to the SDRAM config register. There are 24 valid bits in this field (with the MSB corresponding to Rw/exc thresh) |
| 4 | Enable bitmap SLSW | Bits 31:0 of the chip level register enable bit map. Bit 0 corresponds to chip level config register 0 |
| 8 | Enbale bitmap LSW | Bits 60:32 of the chip level register enable bit map. Bit 0 corresponds to chip level config register 32 |
| 12 | PLL config | See Table 71 |
| 16 | config | SDRAM Config Register |
| 20 | config 2 | SDRAM Config 2 Register |
| 24 | Refresh ctl | SDRAM Refresh Control Register |
| 28 | Timing 1 | SDRAM Timing 1 Register |
| 32 | Timing 2 | SDRAM Timing 2 Register |
| 36 | Timing 3 | SDRAM Timing 3 Register |
| 40 | Nvm timing | LPDDR2-NVM Timing Register |
| 44 | Pwr management | Power Management Control Register |
| 48 | IODFT\_TLGC | IODFT Test Logic Global Control Register |
| 52 | Perf ctl cfg | Performance Counter Config Register |
| 56 | Perf ctl sel | Performance Counter Master Region Select Register |
| 60 | Read idle ctl | Read Idle Control Register |
| 64 | Irq enable | System VBUSM Interrupt Enable Set Register |
| 68 | Zq config | SDRAM Output Impedance Calibration Config Register |
| 72 | Temp alert cfg | Temperature Alert Config Register |
| 76 | Phy ctrl 1 | DDR PHY Control 1 Register |
| 80 | Phy ctrl 2 | DDR PHY Control 2 Register |
| 84 | Pri cos map | Priority to Class of Service Mapping Register |
| 88 | Mst id cos map 1 | Master ID to Class of Service 1 Mapping Register |
| 92 | Mst id cos map 2 | Master ID to Class of Service 2 Mapping Register |
| 96 | Ecc ctrl | ECC Control Register |
| 100 | Ecc addr rng 1 | ECC Address Range 1 Register |
| 104 | Ecc addr rng 2 | ECC Address Range 2 Register |
| 108 | Rw/exc thresh | Read Write Execution Threshold Register |
| 112 | DDR 3 Config 0 | Chip level config register 0 |
| 114 | DDR 3 Config 1 | Chip level config register 1 |
| 120 | DDR 3 Config 2 | Chip level config register 2 |
| 124 | DDR 3 Config 3 | Chip level config register 3 |
| 128 | DDR 3 Config 4 | Chip level config register 4 |
| 132 | DDR 3 Config 5 | Chip level config register 5 |
| 136 | DDR 3 Config 6 | Chip level config register 6 |
| 140 | DDR 3 Config 7 | Chip level config register 7 |
| 144 | DDR 3 Config 8 | Chip level config register 8 |
| 148 | DDR 3 Config 9 | Chip level config register 9 |
| 152 | DDR 3 Config 10 | Chip level config register 10 |
| 156 | DDR 3 Config 11 | Chip level config register 11 |
| 160 | DDR 3 Config 12 | Chip level config register 12 |
| 164 | DDR 3 Config 13 | Chip level config register 13 |
| 168 | DDR 3 Config 14 | Chip level config register 14 |
| 172 | DDR 3 Config 15 | Chip level config register 15 |
| 176 | DDR 3 Config 16 | Chip level config register 16 |
| 180 | DDR 3 Config 17 | Chip level config register 17 |
| 184 | DDR 3 Config 18 | Chip level config register 18 |
| 188 | DDR 3 Config 19 | Chip level config register 19 |
| 192 | DDR 3 Config 20 | Chip level config register 20 |
| 196 | DDR 3 Config 21 | Chip level config register 21 |
| 200 | DDR 3 Config 22 | Chip level config register 22 |
| 204 | DDR 3 Config 23 | Chip level config register 23 |
| 208 | DDR 3 Config 24 | Chip level config register 24 |
| 212 | DDR 3 Config 25 | Chip level config register 25 |
| 216 | DDR 3 Config 26 | Chip level config register 26 |
| 220 | DDR 3 Config 27 | Chip level config register 27 |
| 224 | DDR 3 Config 28 | Chip level config register 28 |
| 228 | DDR 3 Config 29 | Chip level config register 29 |
| 232 | DDR 3 Config 30 | Chip level config register 30 |
| 236 | DDR 3 Config 31 | Chip level config register 31 |
| 240 | DDR 3 Config 32 | Chip level config register 32 |
| 244 | DDR 3 Config 33 | Chip level config register 33 |
| 248 | DDR 3 Config 34 | Chip level config register 34 |
| 252 | DDR 3 Config 35 | Chip level config register 35 |
| 256 | DDR 3 Config 36 | Chip level config register 36 |
| 260 | DDR 3 Config 37 | Chip level config register 37 |
| 264 | DDR 3 Config 38 | Chip level config register 38 |
| 268 | DDR 3 Config 39 | Chip level config register 39 |
| 272 | DDR 3 Config 40 | Chip level config register 40 |
| 276 | DDR 3 Config 41 | Chip level config register 41 |
| 280 | DDR 3 Config 42 | Chip level config register 42 |
| 284 | DDR 3 Config 43 | Chip level config register 43 |
| 288 | DDR 3 Config 44 | Chip level config register 44 |
| 292 | DDR 3 Config 45 | Chip level config register 45 |
| 296 | DDR 3 Config 46 | Chip level config register 46 |
| 300 | DDR 3 Config 47 | Chip level config register 47 |
| 304 | DDR 3 Config 48 | Chip level config register 48 |
| 308 | DDR 3 Config 49 | Chip level config register 49 |
| 312 | DDR 3 Config 50 | Chip level config register 50 |
| 316 | DDR 3 Config 51 | Chip level config register 51 |
| 320 | DDR 3 Config 52 | Chip level config register 52 |
| 324 | DDR 3 Config 53 | Chip level config register 53 |
| 328 | DDR 3 Config 54 | Chip level config register 54 |
| 332 | DDR 3 Config 55 | Chip level config register 55 |
| 336 | DDR 3 Config 56 | Chip level config register 56 |
| 338 | DDR 3 Config 57 | Chip level config register 57 |
| 342 | DDR 3 Config 58 | Chip level config register 58 |
| 346 | DDR 3 Config 59 | Chip level config register 59 |
| 350 | DDR 3 Config 60 | Chip level config register 60 |

**Table 70. DDR Configuration**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| DDR PLL Configuration Bit Fields | | | | | | | | | | |
| 31 | 30 | 29 |  | 16 | 15 |  | 8 | 7 |  | 0 |
| PLL Config Ctl | | PLL Multiplier | | | PLL Pre-Divider | | | PLL Post-divider | | |

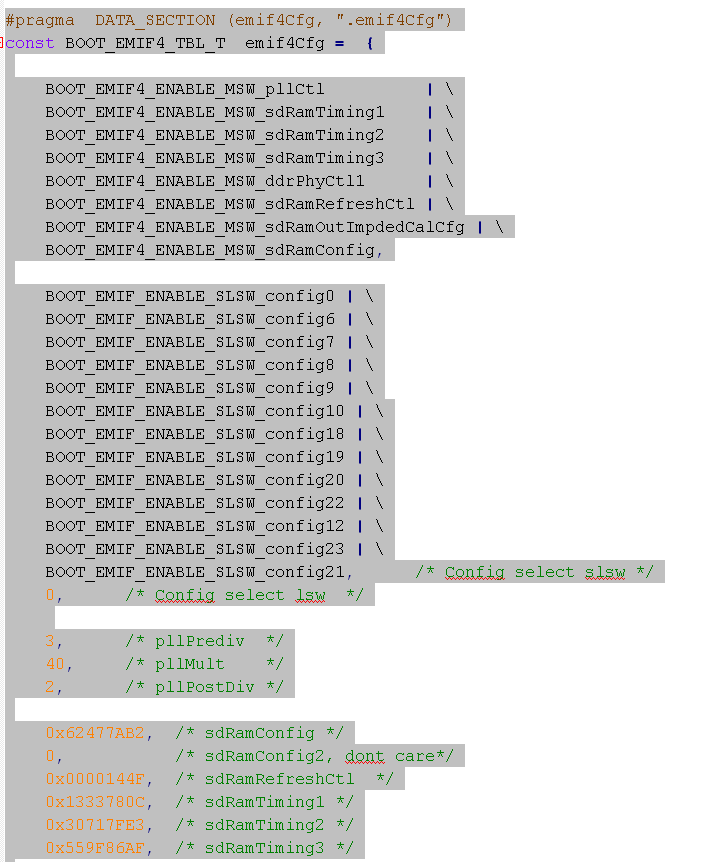
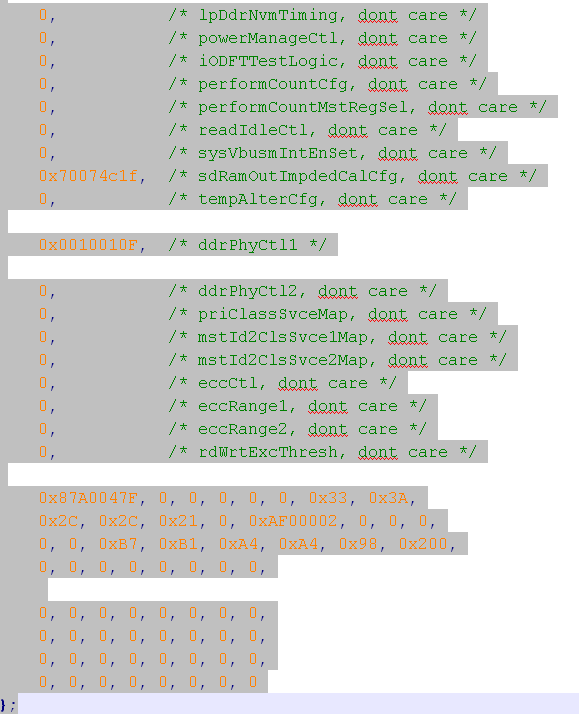
**Table 71. DDR PLL Configuration Bit Fields**

|  |  |  |
| --- | --- | --- |
| DDR PLL Configuration Field Description | | |
| Field | Value | Description |
| Pll Config Ctl | 0b00 | Pll is not configured |
| 0b01 | PLL is configured only if it is currently disabled or in bypass |
| 0b10 | PLL is configured |
| 0b11 | PLL is disabled and put into bypass |
| Pre-divider | 0-255 | Input clock division. The value 0 is treated as pre-divide by 1 |
| Multiplier | 0-16384 | Multiplier. The value is 0 treated as multiply by 1 |
| Post-divider | 0-255 | PLL output division. The value 0 is treated as post divide by 1 |

**Table 72. DDR PLL Configuration Field Descriptions**

### Appending DDR configuration to boot image:

In order to initialize the DDR configuration table the spiboot application image includes a DDR configuration structure in a DATA\_SECTION with a #pragma to put the data section into the memory section .emif4Cfg as shown below:

The .emif4Cfg section is then placed at the DDR configuration table address define by the RBL using the following memory sections defined in the spiboot.cmd file.

