

C6678 SPI Boot Example

Purpose

The purpose of this lab is to demonstrate all the steps that are needed to boot an allocation from SPI flash memory.

Dependencies:

- Code composer Studio v5 or v6
- MCSDK or Processor SDK RTOS software for C6678

Note: Files under mcsdk_2_01_XX_YY\tools\boot_loader have been moved to pdk_c66xx_2_0_0\packages\ti\boot

Task 1: Observe the source file and build the application. Verify that it works correctly

The example application included in the package can be built using gmake or CCS in the Windows environment.

GMAKE based built Procedure:

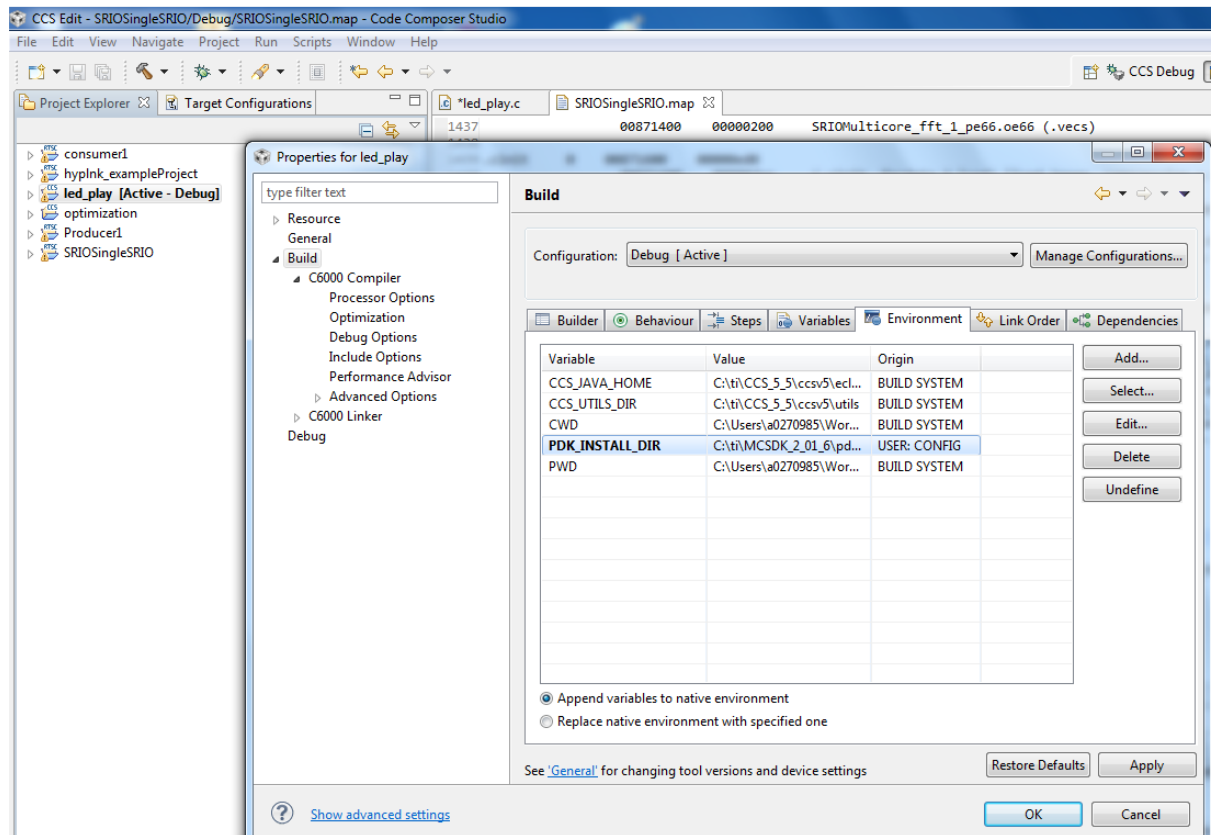
1. Browse to makefile in the led_play folder under src folder
2. In the Makefile set the path to C6000_FOLDER and PDK_PACKAGES. Note that the PDK_PACKAGES must contain path until the packages folder inside the PDK component.
3. In the windows command prompt, add the gmake path to the PATH variable using
Set PATH=%PATH%;<CCS_INSTALL_PATH>\utils\bin
4. From command line you can now build the example using

gmake clean

gmake all

CCS based build Procedure

1. Open CCS Editor and import the led_play project from the src folder.
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.



3. Right click on the project in the Editor view and Rebuild the project in your CCS environment.
4. Load and run the led_play.out file on core 0 of the EVMC6678 to ensure that the rebuilt example runs on your EVM. Observe the LED blinks.

Task 2: Building the btbl file – boot table format

1. Copy the led_play.out file to the build folder
2. In spiboot.bat, modify the path to IBL_UTIL to point to the boot utilities provided in the SDK.
3. From windows command prompt, execute the spiboot.bat file to generate the boot image in the boot table format

4. The details of each step involved in creating the boot image is described in the sections below.

NOTE: Since EVM is designed to have FPGA firmware redirect the DSP core to run IBL, for direct SPI boot don't forget to change the boot address to 0x00 from 0x51 inserted by the romparse utility as described in Section 4

Section 1: Create boot table from application binary using Hex6x Utility

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. (...\\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:-

- 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
1. Create a new directory c:\\temp
 2. 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
 3. Copy hex6x from the bin directory (...\\ccsv6\\tools\\compiler\\c6000_7.x.xx\\bin) to the temp directory
 4. Open a cmd window and cd it to the temp directory
 5. Create the rmd file led_play.rmd using notepad or any other editor as follows:

led_play.out

-a

-boot

-e __c_init00

ROMS

{

ROM1: org = 0x0C000000, length = 0x100000, memwidth = 32, romwidth = 32

```

    files = {led_play.btbl}
}

```

6. Run hex6x with led_play.rmd "hex6x led_play.rmd"
7. The following is a screen shot of the hex6x run:

```

c:\b_temp>
c:\b_temp>
c:\b_temp>
c:\b_temp>hex6x led_play.rmd
Translating to ASCII-Hex format...
"led_play.out" ==> .text <BOOT LOAD>
"led_play.out" ==> .cinit <BOOT LOAD>
"led_play.out" ==> .const <BOOT LOAD>
"led_play.out" ==> .switch <BOOT LOAD>

c:\b_temp>
c:\b_temp>
c:\b_temp>dir
Volume in drive C is OSDisk
Volume Serial Number is 7498-8BE9

Directory of c:\b_temp

08/05/2014  11:28 AM    <DIR>          .
08/05/2014  11:28 AM    <DIR>          ..
05/07/2014  10:35 AM             890,880 hex6x.exe
08/05/2014  11:29 AM             172,772 led_play.btbl
08/05/2014  09:39 AM             543,452 led_play.out
08/05/2014  11:26 AM              155 led_play.rmd
               4 File(s)          1,607,259 bytes
               2 Dir(s)  15,320,596,480 bytes free

```

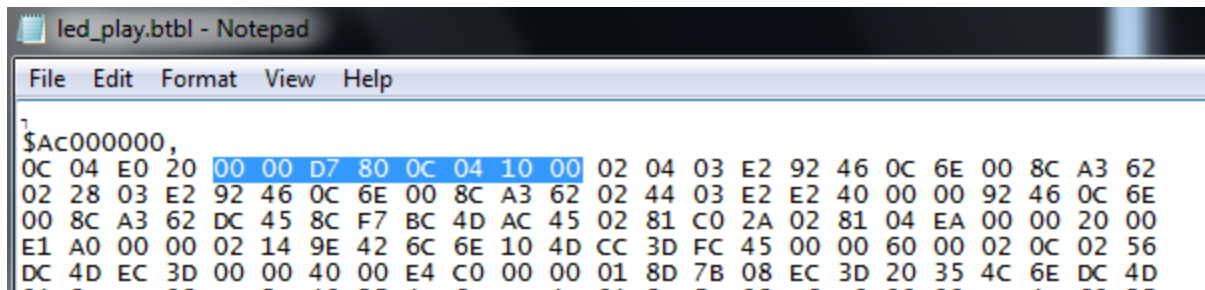
8. Do dir and notice that the file led_play.btbl was generated
9. Note that if you look at the led_play.map map file, and find the text section, you can see the definition of the section and the length in the btbl file. From the map file:

```

48 SECTION ALLOCATION MAP
49
50 output
51 section  page  origin      length      attributes/
52 -----  -
53 .system   0      0c000000  00041000  UNINITIALIZED
54           0c000000  00000008  rts6600_elf.lib : memory.obj (.system)
55           0c000008  00040ff8  --HOLE--
56
57 .text     0      0c041000  0000d780
58           0c041000  00001fe0  ti.platform.evm66781.ae66 : platform.obj (.text)
59

```

And from the file led_play.btbl:



```
$Ac000000,  
0C 04 E0 20 00 00 D7 80 0C 04 10 00 02 04 03 E2 92 46 0C 6E 00 8C A3 62  
02 28 03 E2 92 46 0C 6E 00 8C A3 62 02 44 03 E2 E2 40 00 00 92 46 0C 6E  
00 8C A3 62 DC 45 8C F7 BC 4D AC 45 02 81 C0 2A 02 81 04 EA 00 00 20 00  
E1 A0 00 00 02 14 9E 42 6C 6E 10 4D CC 3D FC 45 00 00 60 00 02 0C 02 56  
DC 4D EC 3D 00 00 40 00 E4 C0 00 00 01 8D 7B 08 EC 3D 20 35 4C 6E DC 4D
```

Section 2: Convert to i2c/SPI format

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.

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_XX_YY\tools\boot_loader\ibl\src\util\btoccs

Note: For Processor SDK RTOS users the utilities can be found under

pd_k_c66xx_2_0_0\packages\ti\boot\ibl\src\util\btoccs

1. Copy the b2i2c.exe utility from the release (directory ...\\MCSDK_2_01_XX\\mcsdk_2_01_XX_YY\\tools\\boot_loader\\ibl\\src\\util\\btoccs into the temp directory
2. Run b2i2c, specify the input and output file name "b2i2c led_play.btbl led_play.btbl.i2c"
3. The screen shot of the run is given below
4. Do dir and see that the i2c format file led_play.btbl.i2c was generated

```

c:\b_temp>
c:\b_temp>
c:\b_temp>
c:\b_temp>b2i2c led_play.btbl led_play.btbl.i2c

c:\b_temp>
c:\b_temp>dir
Volume in drive C is OSDisk
Volume Serial Number is 7498-8BE9

Directory of c:\b_temp

08/05/2014  11:58 AM    <DIR>          .
08/05/2014  11:58 AM    <DIR>          ..
11/19/2012  07:33 PM             24,587 b2i2c.exe
05/07/2014  10:35 AM          890,880 hex6x.exe
08/05/2014  11:29 AM          172,772 led_play.btbl
08/05/2014  11:58 AM          175,933 led_play.btbl.i2c
08/05/2014  09:39 AM          543,452 led_play.out
08/05/2014  11:26 AM           155 led_play.rmd
               6 File(s)          1,807,779 bytes
               2 Dir(s)  15,318,073,344 bytes free

```

Section 3: Convert to CCS downloaded format

Next the i2c formatted file need to be converted into CCS acceptable .dat format using b2ccs utility present in the mscdk\tools\boot_loader\ibl\src\util\btoccs.

1. Copy the b2ccs.exe utility from the release (directory ...\\MSCSDK_2_01_XX\\mscdk_2_01_XX_YY\\tools\\boot_loader\\ibl\\src\\util\\btoccs into the temp directory
2. Run b2ccs, specify the input and output file name "b2ccs led_play.btbl.i2c led_play.i2c.ccs"
3. The screen shot of the run is given below
4. Do dir and see that the i2c format file led_play.i2c.ccs was generated

```

c:\h_temp>
c:\h_temp>
c:\h_temp>
c:\h_temp>b2ccs led_play.btbl.i2c led_play.i2c.ccs

c:\h_temp>
c:\h_temp>dir
Volume in drive C is OSDisk
Volume Serial Number is 7498-8BE9

Directory of c:\h_temp

08/05/2014  01:38 PM    <DIR>          .
08/05/2014  01:38 PM    <DIR>          ..
11/19/2012  07:33 PM             24,008 b2ccs.exe
11/19/2012  07:33 PM             24,587 b2i2c.exe
05/07/2014  10:35 AM            890,880 hex6x.exe
08/05/2014  11:29 AM            172,772 led_play.btbl
08/05/2014  11:58 AM            175,933 led_play.btbl.i2c
08/05/2014  01:38 PM            173,529 led_play.i2c.ccs
08/05/2014  09:39 AM            543,452 led_play.out
08/05/2014  11:26 AM              155 led_play.rnd
               8 File(s)          2,005,316 bytes
               2 Dir(s)    15,307,890,688 bytes free

```

Section 4: Adding Boot parameter Table

An updated boot parameter table is read from the SPI before the actual boot starts. To combine together the boot parameter table and the boot table in the ccs format romparse.exe is used. A *.map file contains the name of the boot table and the values for the boot parameter table. The following shows a standard boot parameter map file:

```

section {

boot_mode = 50

param_index = 0

options = 1

core_freq_mhz = 1000

exe_file = "led_play.i2c.ccs"

next_dev_addr_ext = 0x0

sw_pll_prediv = 5

sw_pll_mult = 32

sw_pll_postdiv = 2

```

```

sw_pll_flags = 1

addr_width = 24

n_pins = 4

csel = 0

mode = 0

c2t_delay = 0

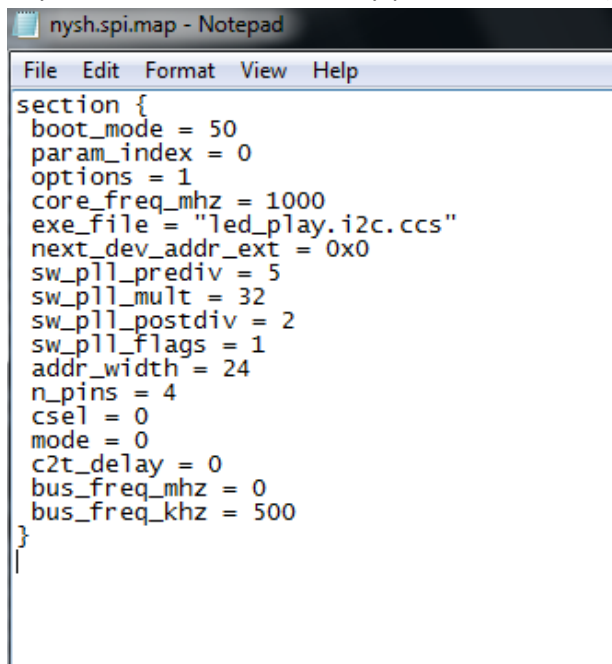
bus_freq_mhz = 0

bus_freq_khz = 500

}

```

1. Copy romparse.exe utility from
\MCSDK_2_XX_YY\mcSDK_2_01_XX_YY\tools\boot_loader\ibl\src\util\romparse to the temp directory
2. Create a map file. You can copy and paste the above file into nysh.spi.map (nysh stands for Keystone I first two families, Nyquist and Shannon). A screen shot is given below



3. Run romparse with the map file as a parameter "romparse nysh.spi.map"
4. Note that the program romparse hardcoded the name of the output file (i2crom.ccs) and the i2c address into the parameter table that is appended to the boot table.
5. The screen shot of the run is given below

6. Do dir and see that the i2c format file i2crom.ccs was generated

```
c:\b_temp>
c:\b_temp>romparse nysh.spi.map

c:\b_temp>
c:\b_temp>dir
Volume in drive C is OSDisk
Volume Serial Number is 7498-8BE9

Directory of c:\b_temp

08/05/2014  02:21 PM    <DIR>          .
08/05/2014  02:21 PM    <DIR>          ..
11/19/2012  07:33 PM             24,008 b2ccs.exe
11/19/2012  07:33 PM             24,587 b2i2c.exe
05/07/2014  10:35 AM            890,880 hex6x.exe
08/05/2014  02:22 PM            176,601 i2crom.ccs
08/05/2014  11:29 AM            172,772 led_play.btbl
08/05/2014  11:58 AM            175,933 led_play.btbl.i2c
08/05/2014  01:38 PM            173,529 led_play.i2c.ccs
08/05/2014  09:39 AM            543,452 led_play.out
08/05/2014  11:26 AM              155 led_play.rmd
08/05/2014  02:21 PM              332 nysh.spi.map
11/19/2012  07:33 PM            66,842 romparse.exe
          11 File(s)          2,249,091 bytes
           2 Dir(s)  15,306,825,728 bytes free
```

7. The program romparse was developed to work with EEPROM connected via i2c. SPI boot protocol is the same as i2c, except that EEPROM is connected to page 0x51 of the i2c while SPI boot starts from 0. The 0x51 is hard-written into the output file i2crom.ccs. The user must change this value into 00. This is done by the following:
 - a. Open the file i2crom.ccs with an editor (I use notepad in the screen shots below)

- c. Save the modified file

Section 5: Big Endian format

The program led_play was built as little endian. The EVM is running as little endian as well, but the RBL always works as big endian. The program byteswapccs swaps the bytes for big endian RBL. The source for byteswapccs.c is given in the Appendix. An executable was built and will be given to the students.

1. Create an EXE from byteswapccs.c (This is already done)
2. Run byteswapccs.exe with the input file as the i2crom.ccs generated at previous step and output as the app.dat that will be flashed to the NOR
3. The screen shot of the run is given below
4. Do dir and see that the big endian file app.dat was generated

```

c:\b_temp>
c:\b_temp>
c:\b_temp>byteswapccs i2crom.ccs app.dat

c:\b_temp>
c:\b_temp>dir
Volume in drive C is OSDisk
Volume Serial Number is 7498-8BE9

Directory of c:\b_temp

08/06/2014  08:24 AM    <DIR>          .
08/06/2014  08:24 AM    <DIR>          ..
08/06/2014  08:24 AM                176,601 app.dat
11/19/2012  07:33 PM                24,008 b2ccs.exe
11/19/2012  07:33 PM                24,587 b2i2c.exe
09/20/2013  11:54 AM                 50,016 byteswapccs.exe
05/07/2014  10:35 AM            890,880 hex6x.exe
08/05/2014  04:14 PM            176,601 i2crom.ccs
08/05/2014  11:29 AM            172,772 led_play.bth1
08/05/2014  11:58 AM            175,933 led_play.bth1.i2c
08/05/2014  01:38 PM            173,529 led_play.i2c.ccs
08/05/2014  09:39 AM            543,452 led_play.out
08/05/2014  11:26 AM                 155 led_play.rmd
08/05/2014  02:21 PM                 332 nysh.spi.map
11/19/2012  07:33 PM             66,842 romparse.exe
               13 File(s)          2,475,708 bytes
               2 Dir(s)    15,636,299,776 bytes free
c:\b_temp>

```

Task3: Flash the EVM SPI flash

1. Configuring the EVM for CCS NOR flashing

Flashing the EVM SPI flash is done using CCS connected to the EVM. The EVM is in no-boot (or sleep) mode. The EVM mode is determined by the setting of four switches on the board, Sw3, SW4, SW5 and SW6. The switches control the following:

- SW3 DSP Boot mode, DSP Configuration
- SW4 DSP boot Configuration
- SW5 DSP boot Configuration
- SW6 DSP boot Configuration, PLL setting, PCIe mode Selection

The following table is taken from

http://processors.wiki.ti.com/index.php/TMDXEVM6678L_EVM_Hardware_Setup describes the various mode setting of the EVM:

Boot Mode Dip Switch Settings

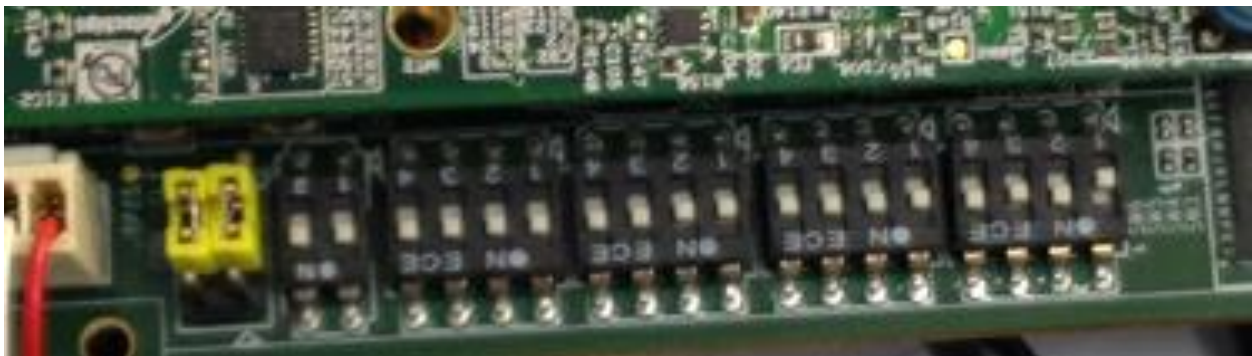
The EVM supports booting image from various devices (EEPROM, NAND or NOR) via IBL (at I2C address 0x51), I2C EEPROM (at I2C address 0x50), and various boot modes (such as Ethernet, SRIO, PCIe, SPI etc.) which address the boot source directly from the ROM code. Below is the table showing the boot mode settings for different boot mode that the EVM supports:

Boot Mode	DIP SW3 (Pin1, 2, 3, 4)	DIP SW4 (Pin1, 2, 3, 4)	DIP SW5 (Pin1, 2, 3, 4)	DIP SW6 (Pin1, 2, 3, 4)
IBL NOR boot on image 0 (default)	(off, off, on, off) ^{1,2}	(on, on, on, on) ³	(on, on, on, off) ⁴	(on, on, on, on)
IBL NOR boot on image 1	(off, off, on, off)	(off, on, on, on)	(on, on, on, off)	(on, on, on, on)
IBL NAND boot on image 0	(off, off, on, off)	(on, off, on, on)	(on, on, on, off)	(on, on, on, on)
IBL NAND boot on image 1	(off, off, on, off)	(off, off, on, on)	(on, on, on, off)	(on, on, on, on)
IBL TFTP boot	(off, off, on, off)	(on, on, off, on)	(on, on, on, off)	(on, on, on, on)
I2C POST boot	(off, off, on, off)	(on, on, on, on)	(on, on, on, on)	(on, on, on, on)
ROM SPI Boot ⁸	(off, on, off, off)	(on, on, on, on)	(on, on, off, on)	(on, on, on, on)
ROM SRIO Boot ⁵	(off, off, on, on)	(on, on, on, off)	(on, off, on, off)	(off, on, on, on)
ROM Ethernet Boot ⁶	(off, on, off, on)	(on, on, on, off)	(on, on, off, off)	(off, on, on, on)
ROM PCIe Boot ⁷	(off, on, on, off)	(on, on, on, on)	(on, on, on, off)	(off, on, on, on)
No boot	(off, on, on, on)	(on, on, on, on)	(on, on, on, on)	(on, on, on, on)

The location of the switched on the EVM for the non-boot case is given by the following pictures.



And a close-up of the switches

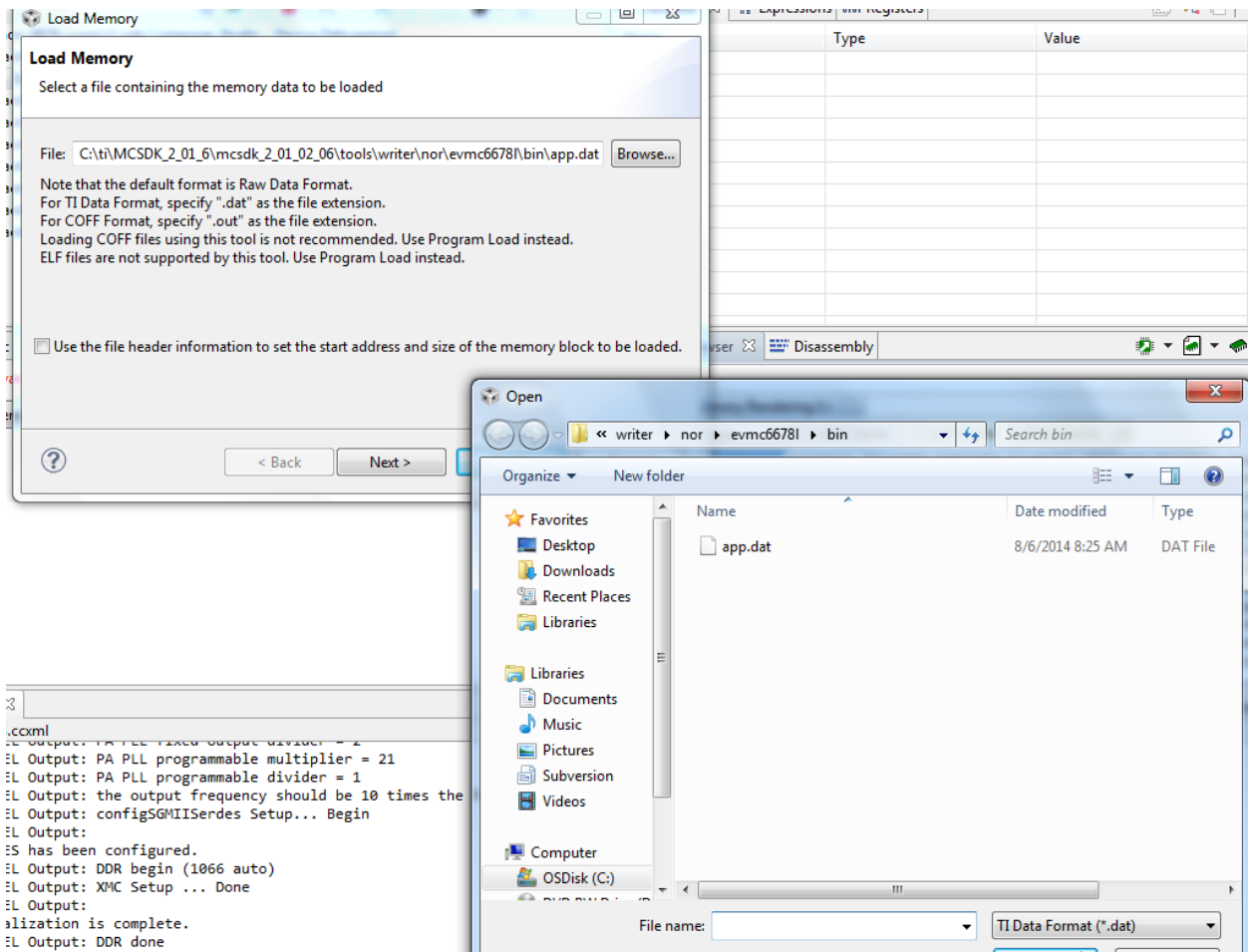


2. The CCS NOR writer is part of the release at location `MCSDK_2_XX_YY\mcsdk_2_01_XX_YY\tools\writer\nor\evmc6678l`. The README.txt file (in the `\nor\docs` directory) gives instructions how to flash the NOR memory. The way the flash

data was developed, some minor modifications to the README.txt file are needed. The following is the updated instructions:

Steps to program the NOR:

1. Be sure to set the boot mode dip switch to no boot/EMIF16 boot mode on the EVM.
2. Copy app.dat file to writer\nor\evmc66xxl\bin directory
3. Change the file_name to app.dat and start_addr to 0 in writer\nor\evmc66xxl\bin\norwriter_input.txt if necessary. See the screen shot below for the norwrite_input.txt file
4. Open CCSv5 and launch the evmc66xx emulator target configuration and connect to core 0.
5. Load the program writer\nor\evmc66xxl\bin\norwriter_evm66xxl.out to CCS, be sure evmc66xxl.gel is used in CCS and DDR is initialized. Ignore the red comment that says that it does not find the main() C source.
6. Open the Memory view (in CCSv5, view->Memory Browser), and view the memory address 0x80000000.
7. Load app.dat to 0x80000000:
 - * In CCSv5, right click mouse in memory window, select "load memory".
 - * Browse and select writer\nor\evmc66xxl\bin\app.dat (TI data format), click "next" . See the following screen shot



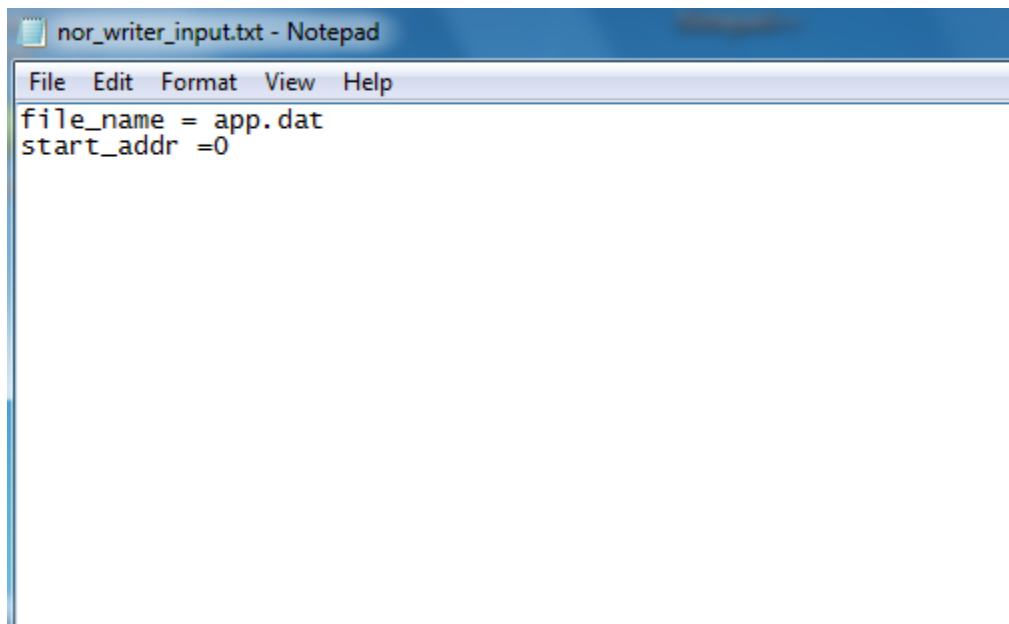
* Set the Start Address to "0x80000000", enter the size of the file. If you check the line "use the file header information to set the start address and size of memory block to be loaded, it will load the file size automatically, see the picture below. Click "finish"

☒ Use the file header information to set the start address and size of the memory block to be loaded.

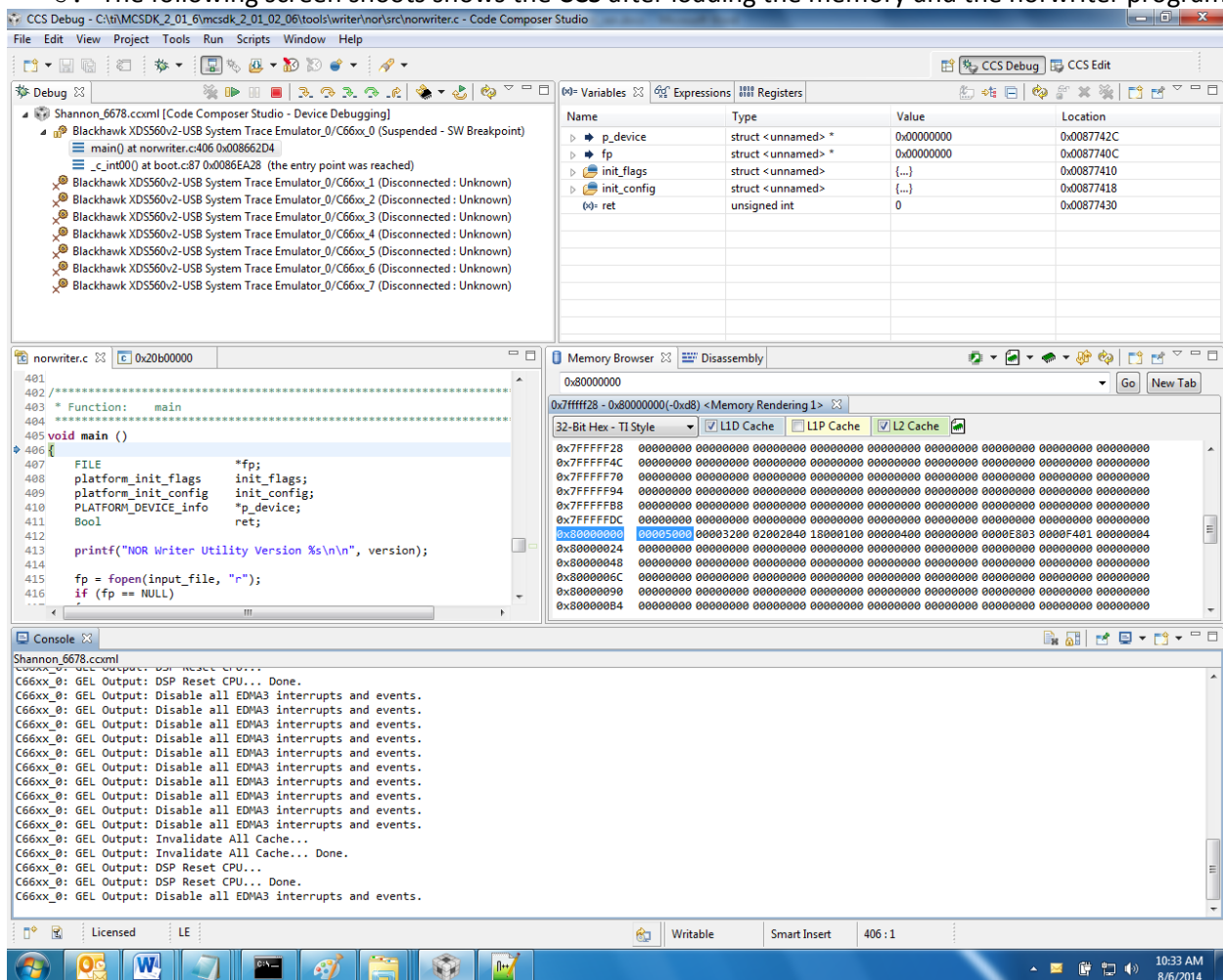
8. After the app.dat file is loaded into the memory, run the program (in CCSv5, press F8), it will start to program the NOR.

9. When programming is completed, the console will print "NOR programming completed successfully", if there is any error, the console will show the error message.

2. The following screen shots shows the nor_writer_input.txt file



3. The following screen shoots shows the CCS after loading the memory and the norwriter program



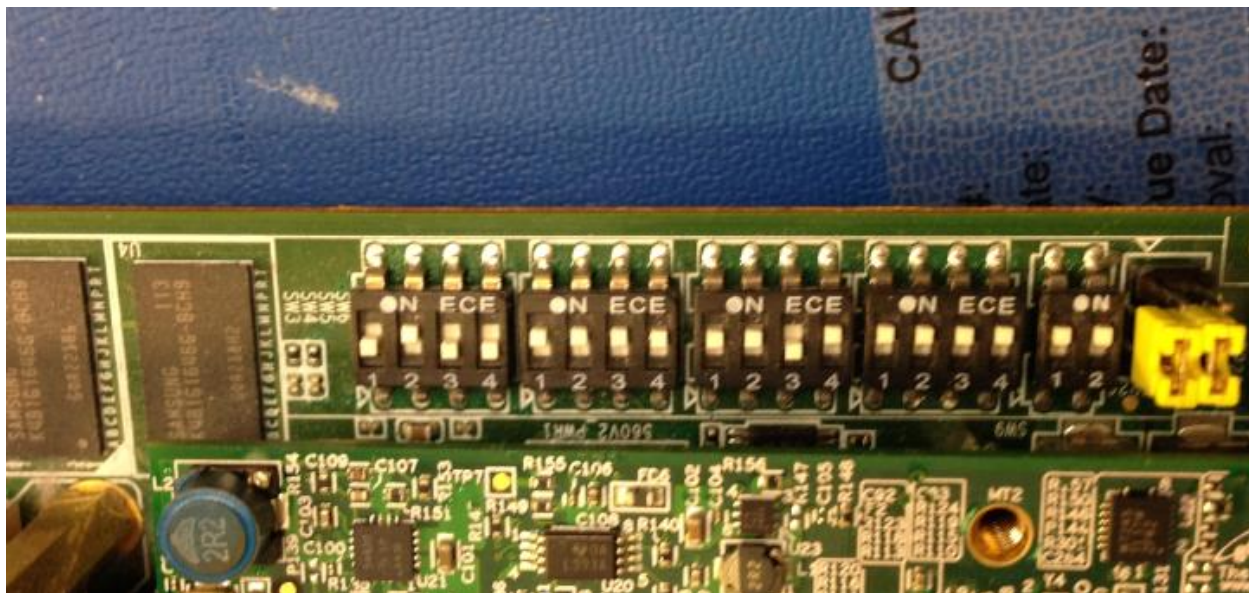
4. The next screen shoot shows the console after running the norwriter program

```
Console X
Shannon_6678.ccxml:CIO
NOR Writer Utility Version 01.00.00.03

Flashing sector 0 (0 bytes of 58860)
Reading and verifying sector 0 (0 bytes of 58860)
NOR programming completed successfully
```

Task 6: Boot from NOR SPI

1. Power off the EVM, change the EVM switched according to the Boot Mode Dip Switch Setting from above - ROM SPI BOOT off, on, off, off on,on,on,on on,on,off,on on,on,on,on . A screen shot of the dip switch is given below



2. Power up the EVM. The LED will blink. The last screen shot shows the blinking LED

