Enabling eMMC support on OMAP-L138/C6748

Software Setup

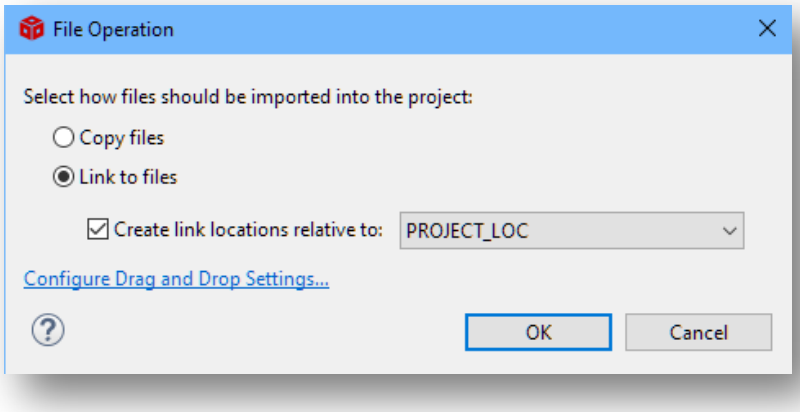
1. Import project **MMCSD_lcdkOMAPL138_DMA_c674xTestProject**

2. As a sanity check, make sure the project builds successfully as-is.

3. **Copy** the following files into your CCS project
   - pdk_omapl138_1_0_11\packages\ti\drv\mmcsd\test\src\main_emmc.c
   - pdk_omapl138_1_0_11\packages\ti\drv\mmcsd\test\src\MMCSD_log.c
   - pdk_omapl138_1_0_11\packages\ti\drv\mmcsd\test\src\MMCSD_log.h

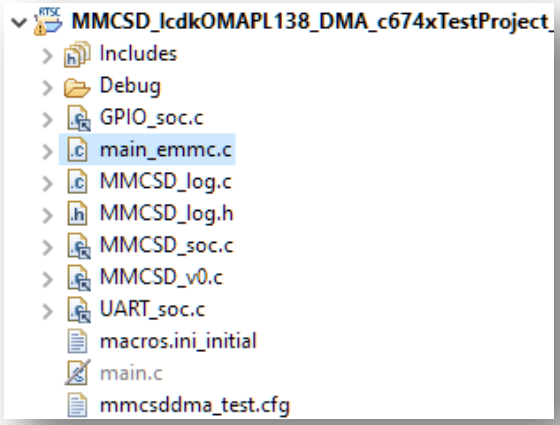
4. *(Recommended)* **Link "**MMCSD_v0.c" to your CCS project so that CCS will include the changes in the binary. If not, the MMCSD driver library will need to be rebuilt via command line gmake in order for the changes to take effect.

   a. **Drag and Drop** pdk_omapl138_1_0_11\packages\ti\drv\mmcsd\src\v0\MMCSD_v0.c into your project.
   b. The following window should pop up. Make sure to link it as shown in the screenshot below.

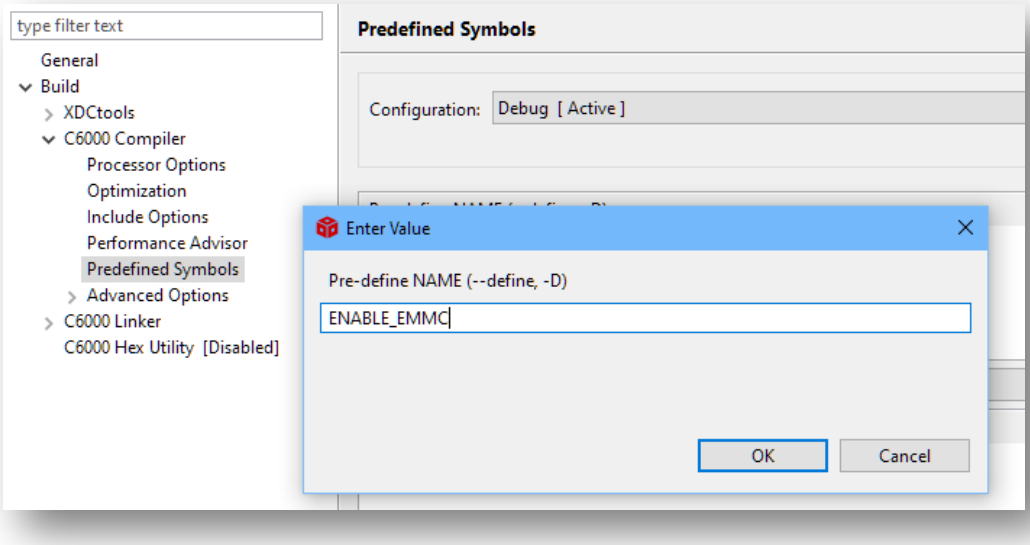


5. Delete or exclude main.c (main_emmc.c will be used instead).

6. At this point, your CCS project should look like the project in the screenshot below.

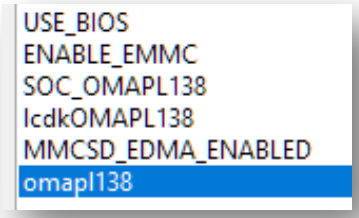


7. Define “ENABLE_EMMC” in project properties as shown in the screenshot below.



8. *(Optional)* If printing logs to the CCS console is preferred over the UART terminal, define “IO_CONSOLE” in project properties.

9. At this point the following symbols should be defined in the CCS project at minimum:



10. Change all instances of “SOC_OMAPL137” to “SOC_OMAPL138” in main_emmc.c.
    a. *Tip*: **CTRL+F to Find/Replace in CCS**

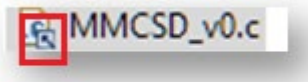
11. Change line 102 to the following to resolve undefined path.

```
#include <ti/drv/mmcsd/test/src/profiling.h> //#include "profiling.h"
```

12. Clean and build the project. It should successfully build after making these changes.

MMCSD driver (MMCSD_v0.c) modifications

Please note since MMCSD_v0.c is linked instead of copied (denoted by the arrow in the icon in the image below) any changes made to it in CCS will persist on your system. Due to this, it is recommended to make a copy of the original as a backup.



13. Update the CMD1 argument to 0xC0FF8080.
    a. Change line 159 to the following:

```
/** \brief MMC Argument Register value for OCR command. */
#define MMCSD_OCR_AGRHL_REG_VAL     (0xC0FF8080) // Originally 0x00FF8000
```

14. Update CMD8 transfer type to read.
    a. Change line 1852 to the following:

```
cmdObj.cmd.xferType   = MMCSD_DIR_READ; // Originally MMCSD_DIR_DONTCARE
```

15. Change line 1857 to the following:

```
transaction.checkStatus = MMCSD_EVENT_EOFCMD | MMCSD_EVENT_READ; // Originally MMCSD_EVENT_EOFCMD | MMCSD_EVENT_ERROR
```

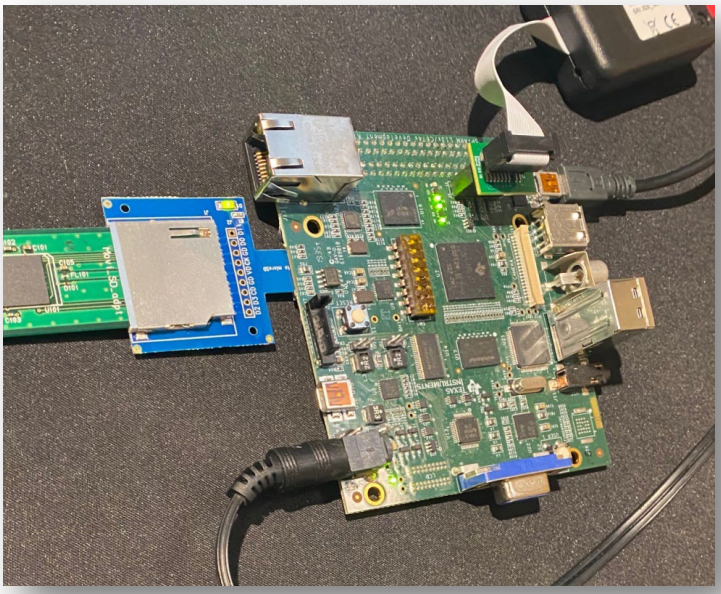
16. Change line 1858 to the following:

```
transaction.flags = MMCSD_CMDRSP_READ | MMCSD_CMDRSP_DATA; // Originally MMCSD_CMDRSP_READ | MMCSD_CMDRSP_WRITE;
```

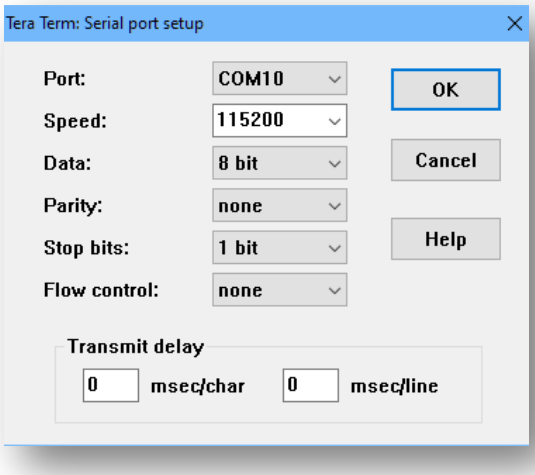
17. Clean and build project.

Hardware Setup

1. The LCDK does not have an onboard eMMC so the microSD slot is used. An eMMC to microSD adapter would be required similar to what is shown below. Soldering the eMMC signals directly to the board is also an option.



2. Connect a mini-USB cable from the PC to the J3 mini-USB port to print logs to the UART terminal. The serial port settings are shown in the screenshot below.



Testing

1. After loading and running the binary on the LCDK, the following output should be shown on the UART terminal or CCS console.

```
------------------------------------------------------
Test Details for test ID      = 9
Test Profile[9]: Description = Default Unit Test (Max speed)
Test Profile[9]: PowerCyle    = Required
Test Profile[9] Device Config : Mode        = SOC Default
Test Profile[9] Device Config : BusVoltage  = SOC Default
Test Profile[9] Device Config : BusWidth    = SOC Default
Test Profile[9] Device Config : Interrupt   = SOC Default
------------------------------------------------------

DMA is enabled
Interrupts are disabled


Performing RAW mode read/write tests ..


MMCSD_Open() completed successfully

SD Read/Write 32 bit ramp
RAW READ/WRITE: Writing test pattern (256 KB) to the EMMC starting at sector 0x4000 in 1 block(s) 256 KB each
RAW READ/WRITE: Reading test pattern (256 KB) from the EMMC starting at sector 0x4000 in 1 block(s) 256 KB each
RAW READ/WRITE: PASS: Read/Write Success for this size (256 KB)

RAW READ/WRITE: Writing test pattern (512 KB) to the EMMC starting at sector 0x4000 in 1 block(s) 512 KB each
RAW READ/WRITE: Reading test pattern (512 KB) from the EMMC starting at sector 0x4000 in 1 block(s) 512 KB each
RAW READ/WRITE: PASS: Read/Write Success for this size (512 KB)

RAW READ/WRITE: Writing test pattern (1024 KB) to the EMMC starting at sector 0x4000 in 1 block(s) 1024 KB each
RAW READ/WRITE: Reading test pattern (1024 KB) from the EMMC starting at sector 0x4000 in 1 block(s) 1024 KB each
RAW READ/WRITE: PASS: Read/Write Success for this size (1024 KB)

RAW READ/WRITE: Writing test pattern (2048 KB) to the EMMC starting at sector 0x4000 in 1 block(s) 2048 KB each
RAW READ/WRITE: Reading test pattern (2048 KB) from the EMMC starting at sector 0x4000 in 1 block(s) 2048 KB each
RAW READ/WRITE: PASS: Read/Write Success for this size (2048 KB)

RAW READ/WRITE: Writing test pattern (4095 KB) to the EMMC starting at sector 0x4000 in 1 block(s) 4095 KB each
RAW READ/WRITE: Reading test pattern (4095 KB) from the EMMC starting at sector 0x4000 in 1 block(s) 4095 KB each
RAW READ/WRITE: PASS: Read/Write Success for this size (4095 KB)

RAW READ/WRITE : Closing the driver..Test Complete for this profile


--------- PLEASE NOTE  ----------

 This test required powercycle.
 Before running the next test, Powercycle/System-reset and run other test

All tests have PASSED
1/1 tests passed
```

Limitations

When using the eMMC driver with DMA enabled, the driver supports a maximum size of 4MB for reads and writes.