Application Note

AM62x Family DDR Debug

Rio Chan, Tushar Thakur

ABSTRACT

This document helps users to know how to debug the DDR TI processor by using TI Sitara AM62x as an example. This document presents a clear guideline on how to debug the DDR. This document focuses on the HS-FS device. If the device is HS-SE, the user must unlock the JTAG on the HS-SE before debugging the DDR.

Three methods are introduced:

1. CCS with JTAG: How to debug and test DDR
2. U-Boot Stage: How to debug and test DDR
3. Kernel Stage: How to debug and test DDR

Navigate here and here for the required HW EVM. Navigate here for the required SW SDK.

Table of Contents

Trademarks

1 Introduction

Use the DDR development flow when using TI processors.

In general, some key points are:

1. Follow the DDR/LPDDR layout guide strictly.
2. Use a single die for design simplicity; this prevents some risks.
3. When the PCB comes, use the CCS JTAG to verify the read /write consistency.
4. When the CCS JTAG DDR testing is complete, the user can use the PCB and is ready on the DDR part.
5. Patch the DDR parameters in U-Boot. Configure the related code.

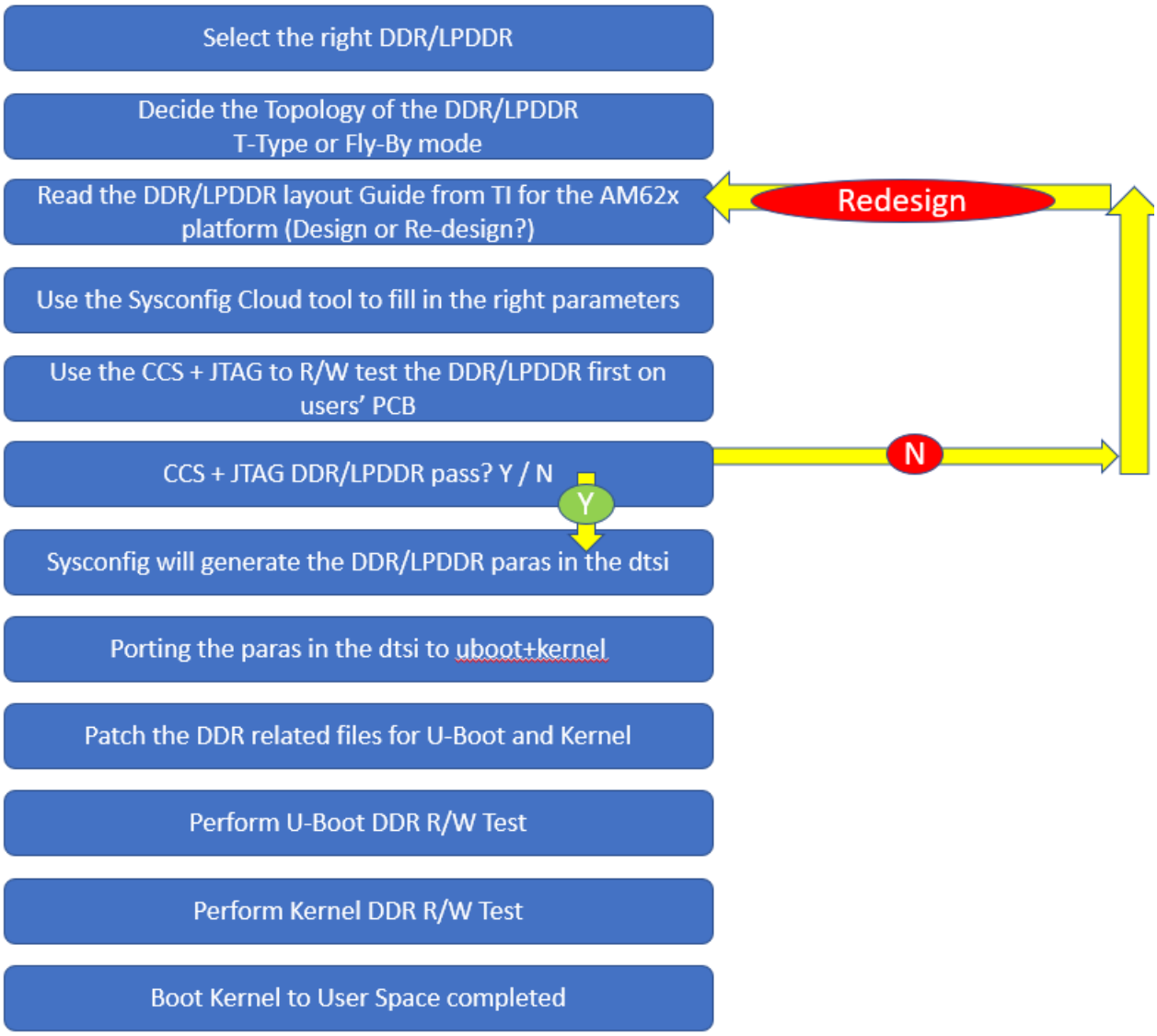

Figure 1-1. TI DDR4/LPDDR4 Developing flow

2 CCS with JTAG: How to Test and Debug DDR

This section shows how to use the TI CCS JTAG to debug and verify the DDR. Use the newest version of the AM62x DDR config tool here. Before using the DDR config tool, use the readme link.

Below is the TI DDR config for the AM62x family. Please see Figure 2-1.

There are five sections:

- Config A: This is an important section; review key factors from the DDR datasheet. Determine memory frequency width, density, and chip select value according to the DDR datasheet.
- DRAM Timing A: Set the CL/CWL (Clock write latency) / CA. These three factors must be matched with the DDR datasheet.
- DRAM Timing B: TI recommends not modifying the parameters here if the user is not familiar with them.
- IO Control A: The parameters inside this section must match the PCB design impedance and voltage.
- IO Control B: The parameters inside this section must match the PCB design impedance and voltage.

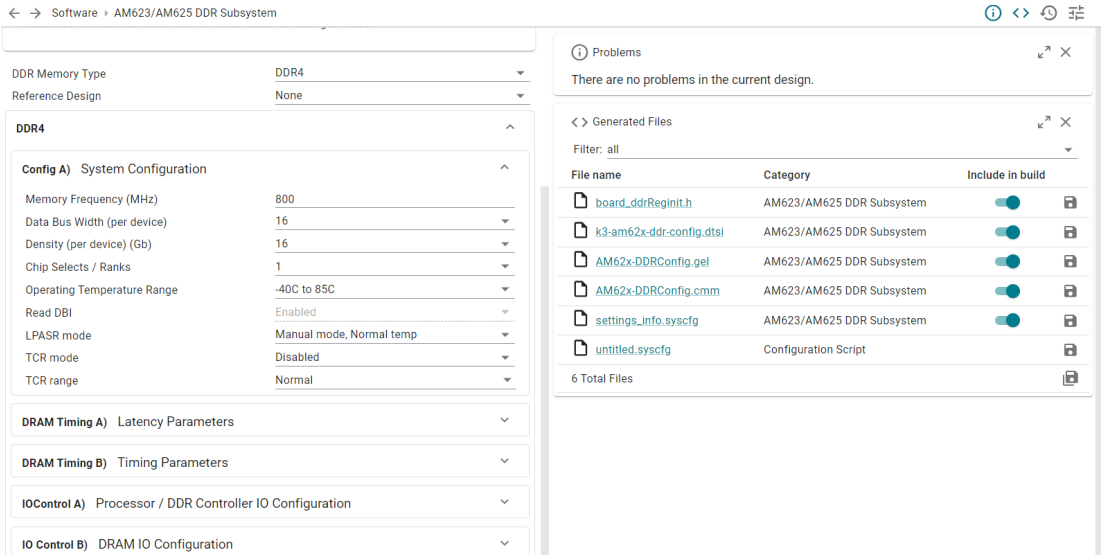

Figure 2-1. AM62X DDR Tool Overview

1. Select the DDR or the LPDDR. If the LPDDR is used, check the system Power IC (PMIC) design, because the key point here is in the sleep / power-save mode for keeping the LPDDR power retention.
2. Decide the topology of the DDR/LPDDR T-type or fly-by mode. Look at the DDR/LPDDR datasheet density and memory topology config. If the user needs 4G Bytes size DDR, then, there are some kinds of those topologies. Case 1: Die inside per chip, and one chip is 1GB, thus the user needs four Chips. Thus, 4x1G = 4G. Case 2: Two dies inside per chip, and one chip is 2GB, thus the user needs two chips. Thus, 2x2G = 4G. Case 3: Four dies inside per chip, and 1 chip is 4GB, as a result the user needs one chip. Thus, 1x4G = 4G. Select Case 3 is simpler to design than cases one and two. Case 3: (Recommended): Users need to consider the T-Topology vs Fly-By-Topology design; consult the memory vendors.
3. Read the DDR/LPDDR layout guide from TI for the AM62x platform. When the HW RD is drawing the DDR/LPDDR layout, please follow the TI AM62x family processor layout guide. Specify the DDR/LPDDR trace length and keep the length is in the reasonable length. Select the topology as simply as possible. Single chip DDR/LPDDR that has the size that fits the user's needs is the best.

4. Use the SysConfig Cloud tool to fill in the right parameters. Use the TI SysConfig Cloud tool (TI recommends using this tool due to live updates and bug fixes). Use this link. A ti.com account is required. Detailed configuration steps are shown in Step 6.
5. Use the CCS and JTAG to R/W test the DDR/LPDDR first on the PCB. Use this video to access the AM62x EVM CCS + JTAG debug demonstration. Detail steps are revealed as the following (applicable only for GP devices)

1. Config the AM62X EVM SW3/SW2 as *No Boot* . For example, JTAG boot.
2. Connect the XDS200 (XDS200 is an example here) USB port to the NB/PC.
3. Create the AM62x CCXML, set this as *Default*, launch the program.
4. Connect to the TIFS core first.
5. Connect to the R5F_0_0 core.
6. Navigate to the menu bar. Navigate to *Scripts*, There is a DDR testing script

Applicable on HSFS devices:

1. Modify the SBL NULL example's *main.c* file to enable only the DDR clock. See Figure 2-2.

```
113     DebugP_log("\r\n");
114     DebugP_log("Starting NULL Bootloader ... \r\n");
115
116     status = SOC_moduleClockEnable(TISCI_DEV_EMIF_DATA_ISO_VD, 1);
117     status += SOC_moduleClockEnable(TISCI_DEV_DDR16SS0, 1);
118     DebugP_assertNoLog(status == SystemP_SUCCESS);
119     loop_forever();
120
121     status = Board_driversOpen();
122     DebugP_assert(status == SystemP_SUCCESS);
```

Figure 2-2. DDR Clock Enable

2. Remove the DDR configuration from the *example.syscfg* file. See Figure 2-3

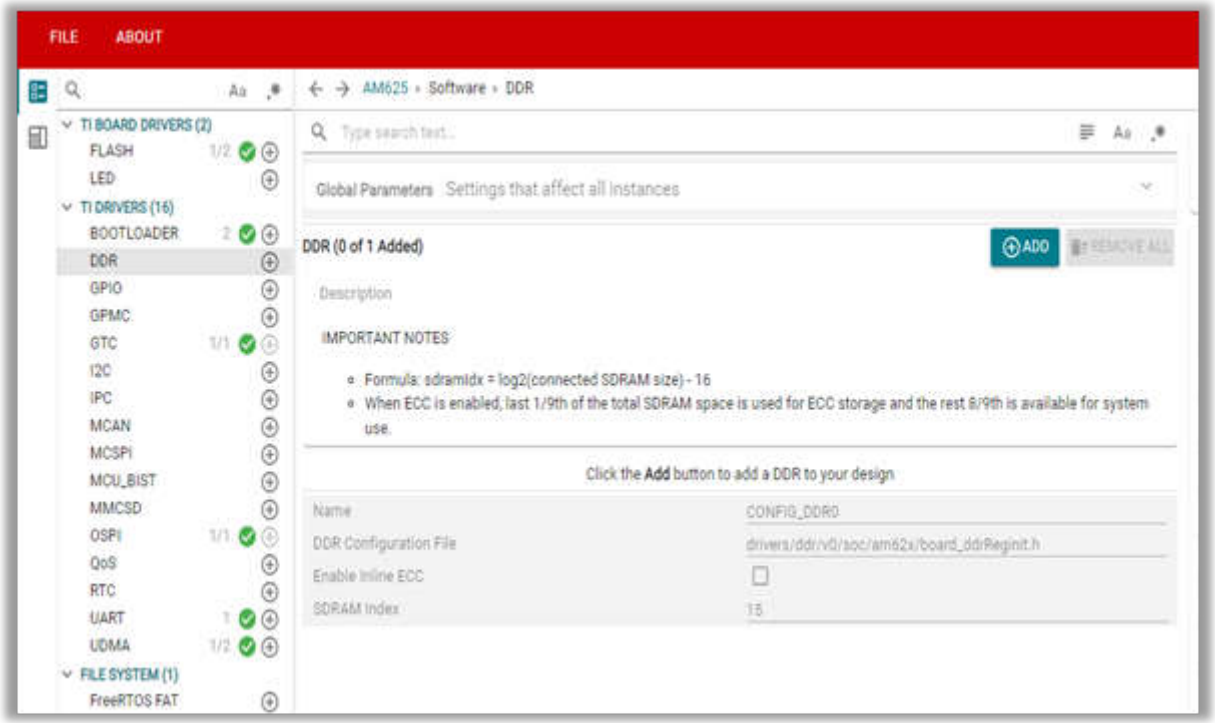

Figure 2-3. Remove DDR Configuration

3. Build the SBL example using the following command.

```
> cd ${MCU+SDK}
> ${gmake|make} -s -C examples\drivers\boot\sbl_null\${device}\r5fss0-0_nortos\ti-arm-clang\
PROFILE=${release|debug}
```

4. Keep the EVM in either UART or DFU boot mode
5. Flash the newly built SBL binary on the EVM either via UART or DFU.
6. Launch the AM62x.ccxml (target configuration) file and connect to R5F0-0 core
7. Navigate to the menu bar. Navigate to *Scripts* and the DDR testing script

6. SysConfig generates the DDR/LPDDR paras in the dtsi. This DTSI is like this file name: *devicetree*.The SysConfig defaults to a generated name if a name is not assigned.

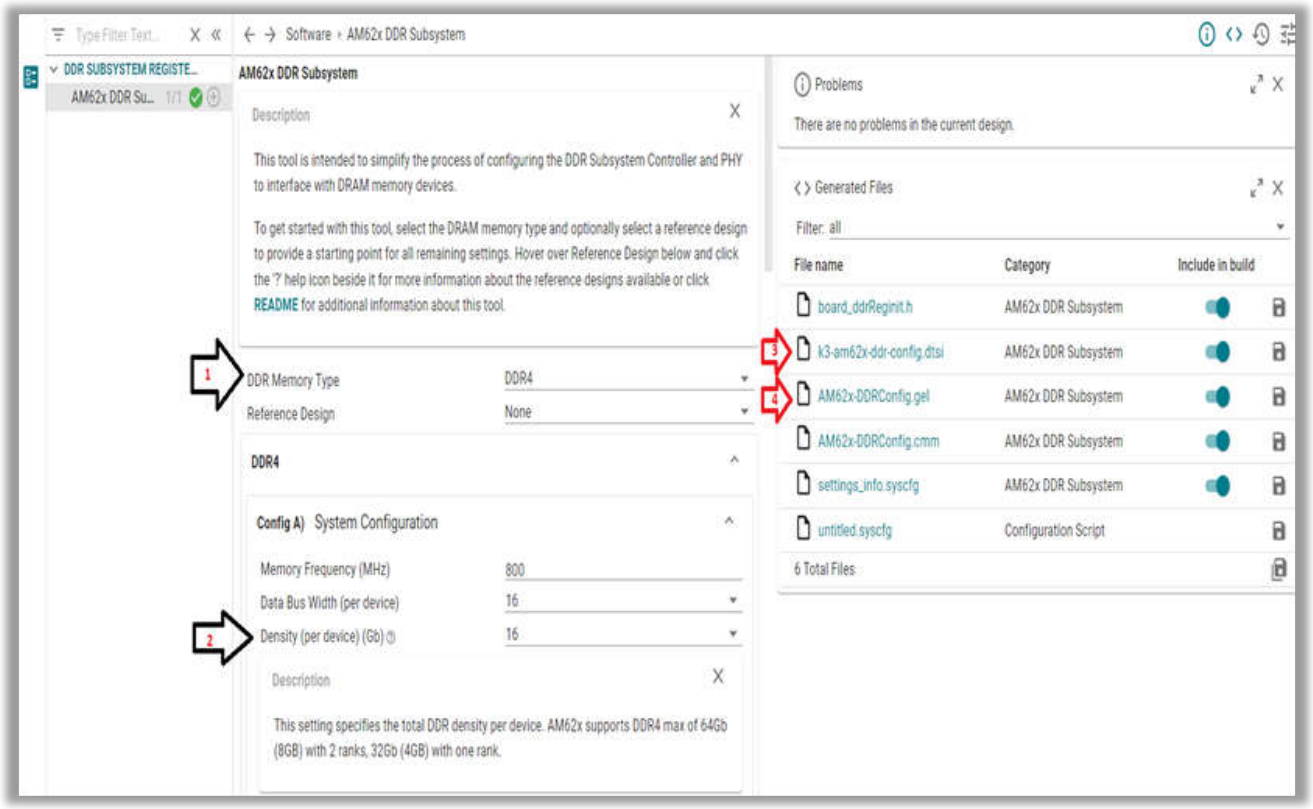

Figure 2-4. AM62X DDR Tool Key Part

7. Porting the paras in the dtsi to U-Boot. Once the SysConfig tool generates the DDR/LPDDR parameters, the user must port the parameters onto this file:

U-Boot file:

sèarch/arm/dts/k3-am62x-sk-ddr4-1600MTs.dtsi

8 Use the DDR related files for U-Boot and Kernel. U-Boot Files: èarch/arm/dts/k3-am625-r5-sk.dts, arch/arm/dts/k3-am625-sk.dts, arch/arm/dts/k3-am62x-sk-ddr4-1600MTs.dtsi, drivers/ram/k3-ddrss/k3-ddrss.c

Kernel Files: k3-am625-sk.dts

9. Perform the U-Boot DDR R/W Test

U-Boot tool is: **Mtest**. Click the link to access TI's U-Boot **Mtest** document.

```
=> md 0x100000 0x10 (This means to display 16 bytes from the 0x100000)
00100000: 0000000f 00000010 00000011 00000012    ................
00100010: 00000013 00000014 00000015 00000016    ................
00100020: 00000017 00000018 00000019 0000001a    ................
00100030: 0000001b 0000001c 0000001d 0000001e    ................
=> mw 0x100000 0xaabbccdd (This means to write the 0xaabbccdd to 0x100000
=> md 0x100000 0x10 (This means to display 16 bytes from the 0x100000)
00100000: aabbccdd 00000010 00000011 00000012    ................
00100010: 00000013 00000014 00000015 00000016    ................
00100020: 00000017 00000018 00000019 0000001a    ................
00100030: 0000001b 0000001c 0000001d 0000001e    ................
=> mw 0x100000 0 6 (This means to write 0 for 6 bytes from the 0x100000.)
=> md 0x100000 0x10 (This means to display 16 bytes from the 0x100000)
00100000: 00000000 00000000 00000000 00000000    ................
00100010: 00000000 00000000 00000015 00000016    ................
00100020: 00000017 00000018 00000019 0000001a    ................
00100030: 0000001b 0000001c 0000001d 0000001e    ................
```

10. Perform the Kernel DDR R/W Test. Kernel tool is: **Memtest** Please make sure the Kernel has this config enabled before using the Memtest: èCONFIG_MEMTEST=y The **Memtest** Example command is like this:

- Virtual Address Memory Test: *memtester 512M 1*. Write *1* with 512M size onto the Virtual Memory (System OS to decide where to write, but the user does not know the physical or actual address.
- Physical Address Memory Test (with -p, *p* means *Physical*.)
- memtester -p 0x880000000 32M 1: This means to write 1 with 32M size onto the Physical Memory address of 0x880000000.

11. Boot Kernel to User Space completed. See Figure 2-5. The Kernel test tool command used below is: *lsmsm + memtester*.

1. This is section of 4G DDR in total. The DDR tested here is 2Gx2 in 1 DDR Chip.
2. This shows the online memory can be used in total 4G.
3. Now, 3G of 4G to do the Memory testing has been tested.

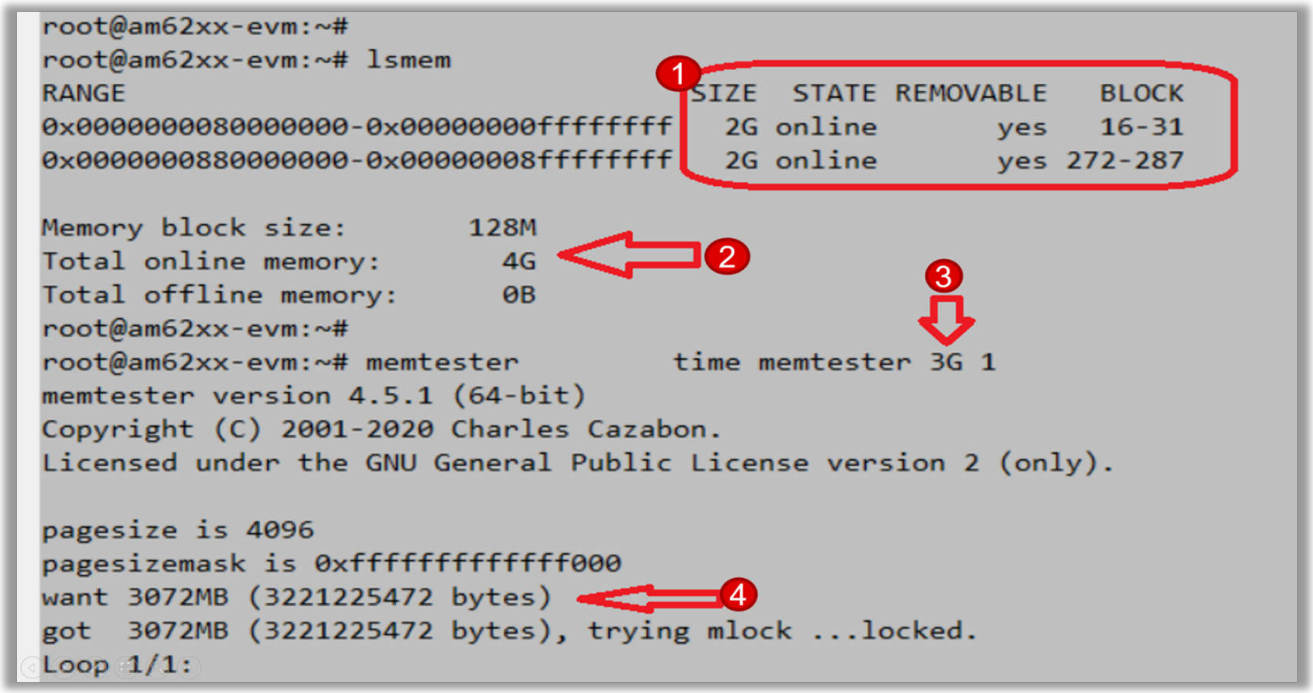

Figure 2-5. Kernel DDR Success Log Example

3 U-Boot Stage: Debug Tips for the DDR

This section describes what abnormal U-Boot logs are related to DDR. Figure 3-1shows the U-Boot stuck in DDR initialization. This means that the CPU tried to load the SPL onto the DDR, but the DDR is not initialized correctly. So, the U-Boot is stuck at that stage. This kind of abnormal log is very common to see if there is DDR that failed to initialize.

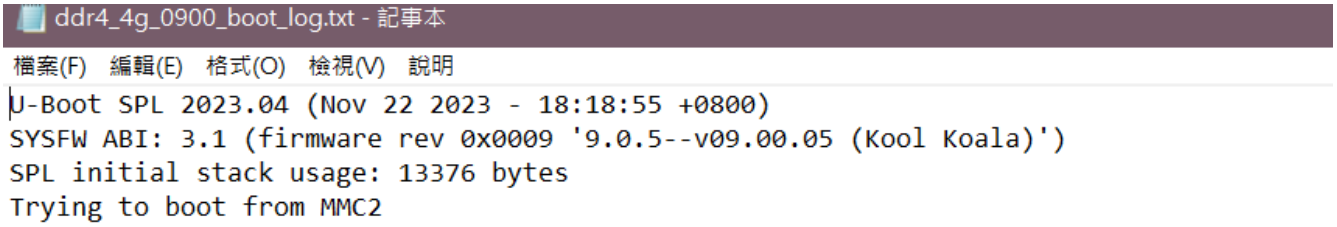

Figure 3-1. Abnormal U-Boot Log Example

Figure 3-2 shows the normal complete boot log on the 4G DDR on the newly designed PCB. Note the two red arrows in the image. Bank 0 and Bank 1 are having the 2GB DDR4 adapted.

The 4GB is effective.

```
U-Boot 2023.04-00001-gf5b119738d-dirty (Nov 22 2023 - 10:18:20 +0800)
initcall: 0000000080822084
U-Boot code: 80800000 -> 808D9738  BSS: -> 808E5010
initcall: 0000000080821eb4
initcall: 0000000080822190
initcall: 0000000080802e48
SoC:   AM62X SR1.0 HS-FS
initcall: 0000000080822a84
Model: Texas Instruments AM625 SK
EEPROM not available at 80, trying to read at 81
Reading on-board EEPROM at 0x51 failed -121
initcall: 0000000080822188
initcall: 0000000080822168
DRAM:  initcall: 0000000080803354
initcall: 0000000080822644
Monitor len: 000E5010
Ram size: 80000000
Ram top: 100000000
initcall: 0000000080821e80
initcall: 0000000080802d98
initcall: 0000000080822120
initcall: 000000008082229c
initcall: 000000008082201c
Reserving 916k for U-Boot at: ff61a000
initcall: 000000008082232c
Reserving 32892k for malloc() at: fd5fb000
initcall: 00000000808222cc
Reserving 152 Bytes for Board Info at: fd5faf60
initcall: 000000008082236c
Reserving 472 Bytes for Global Data at: fd5fad80
initcall: 0000000080821fac
Reserving 63712 Bytes for FDT at: fd5eb4a0
initcall: 00000000808222a4
initcall: 0000000080821e94
initcall: 00000000808222bc
initcall: 00000000808226bc
initcall: 0000000080803358
initcall: 00000000808223b0

RAM Configuration:
Bank #0: 80000000 2 GiB
Bank #1: 880000000 2 GiB

DRAM:  2 GiB (effective 4 GiB)
initcall: 00000000808226d0
initcall: 0000000080821f88
```

Figure 3-2. DDR 4G U-Boot Normal Log.

Cross checking on the other TI Platform which adapted 8G LPDDR4, Figure 3-3 shows 8GB.

```
U-Boot 2023.04-gcb5b1af9a4 (Dec 19 2023 - 11:30:44 +0000)

SoC:   J722S SR1.0 HS-FS
Model: Texas Instruments J722S EVM
Board: J722SX-EVM rev E2
DRAM:  2 GiB (effective 8 GiB)
Core:  68 devices, 29 uclasses, devicetree: separate
Warning: Device tree includes old 'u-boot,dm-' tags: please fix by 2023.07!
MMC:   mmc@fa10000: 0, mmc@fa00000: 1
Loading Environment from nowhere... OK
In:    serial@2800000
Out:   serial@2800000
Err:   serial@2800000
Net:
Warning: ethernet@8000000port@1 (eth0) using random MAC address - 6a:53:69:69:9e:e5
eth0: ethernet@8000000port@1
Hit any key to stop autoboot:  0
switch to partitions #0, OK
mmc1 is current device
SD/MMC found on device 1
Failed to load 'boot.scr'
574 bytes read in 29 ms (18.6 KiB/s)
Loaded env from uEnv.txt
Importing environment from mmc1 ...
21588480 bytes read in 1751 ms (11.8 MiB/s)
68714 bytes read in 36 ms (1.8 MiB/s)
Working FDT set to 88000000
## Flattened Device Tree blob at 88000000
   Booting using the fdt blob at 0x88000000
Working FDT set to 88000000
   Loading Device Tree to 000000008feec000, end 000000008fffffff ... OK
Working FDT set to 8feec000

Starting kernel ...
```

Figure 3-3. DDR 8G U-Boot Normal Log

Using the AM62 platform as an example, what are the DDR configurations and related files in U-Boot?

DDR/LPDDR *config related file* in the U-Boot.

arch/arm/dts/k3-am625-r5-sk.dts b/arch/arm/dts/k3-am625-r5-sk.dts

arch/arm/dts/k3-am625-sk.dts b/arch/arm/dts/k3-am625-sk.dts

k3-am62x-r5-sk-common.dtsi

k3-am62x-sk-ddr4-1600MTs.dtsi

k3-ddrss.c

U-Boot *Debug related* files:

mmc.c

fat.c

spl.c

4 Kernel Stage: Debug Tips for the DDR

Figure 4-1 shows the abnormal kernel logs related to DDR. The Arm register value shows when the kernel is stuck in place, such as *PC / SP (Stack pointer) / x29* (FP, frame pointer registers). The DDR is malfunctioning so that the Arm core cannot access the register with the correct value.

```
2.691280] Hardware name: Texas Instruments AM625 SK (DT)

2.697005] pstate: 80000005 (Nzcv daif -PAN -UAO -TCO BTYPE=--)

2.703248] pc : __alloc_fd+0x68/0x1e0

2.707324] lr : __alloc_fd+0x38/0x1e0

2.711315] sp : ffff80001126bd90

2.714871] x29: ffff80001126bd90 x28: 0000000000000000

2.720421] x27: ffff800010ee04d4 x26: ffff000800100080

2.725972] x25: 0000000000000000 x24: 0000000000000400

2.731522] x23: 0000000000000000 x22: 0000000000000000

2.737073] x21: ffff000800100000 x20: ffff800010f51078

2.742623] x19: 0000000000000000 x18: 000000000000001c
```

Figure 4-1. Abnormal Kernel Log Example

So, what are the DDR configs related files in kernel?

k3-am625-sk.dts

The user can look into this dts and use the section below to modify the DDR size from 2G to 4G.

memory@80000000 {

device_type = "memory";

 # / 2G RAM , comment out these for 2G*/*

 # reg = <0x00000000 0x80000000 0x00000000 0x80000000>;

/ 4G RAM, adding the below for 4G */*

reg = <0x00000000 0x80000000 0x00000000 0x80000000>,

<0x00000008 0x80000000 0x00000000 0x80000000>;

5 Summary

From this application note, users can learn how to debug the DDR by using those tools: JTAG UBoot cmd. Kernel cmd, and how to check the size of the DDR in the whole system.

6 References

1. Texas Instruments, *SK-AM62B: JTAG DDR4G R/W testing cannot be performed on AM62HS-FS*, forum.
2. Texas Instruments, *SK-AM62B: 4G DDR issue*, forum.
3. Texasw Instruments, *AM263: Memory can't be configured to 4GB from device-tree*, forum.
4. Texas Instruments, *SK-AM62B: AM62B EVM JTAG testing is mul-function?*, forum.
5. Texas Instruments, *Unlock JTAG on an HS-SE device*, resource explorer.
6. Texas Instruments, index: ti-u-boot/ti-u-boot, code block.
7. Samsung, *K4ABG165WA-MCTD*, datasheet.
8. Micron, MT40A2G16TBB-062E:F part detail, datasheet.

IMPORTANT NOTICE AND DISCLAIMER

TI PROVIDES TECHNICAL AND RELIABILITY DATA (INCLUDING DATASHEETS), DESIGN RESOURCES (INCLUDING REFERENCE DESIGNS), APPLICATION OR OTHER DESIGN ADVICE, WEB TOOLS, SAFETY INFORMATION, AND OTHER RESOURCES “AS IS” AND WITH ALL FAULTS, AND DISCLAIMS ALL WARRANTIES, EXPRESS AND IMPLIED, INCLUDING WITHOUT LIMITATION ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT OF THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

These resources are intended for skilled developers designing with TI products. You are solely responsible for (1) selecting the appropriate TI products for your application, (2) designing, validating and testing your application, and (3) ensuring your application meets applicable standards, and any other safety, security, regulatory or other requirements.

These resources are subject to change without notice. TI grants you permission to use these resources only for development of an application that uses the TI products described in the resource. Other reproduction and display of these resources is prohibited. No license is granted to any other TI intellectual property right or to any third party intellectual property right. TI disclaims responsibility for, and you fully indemnify TI and its representatives against any claims, damages, costs, losses, and liabilities arising out of your use of these resources.

TI’s products are provided subject to TI’s Terms of Sale, TI’s General Quality Guidelines, or other applicable terms available either on ti.com or provided in conjunction with such TI products. TI’s provision of these resources does not expand or otherwise alter TI’s applicable warranties or warranty disclaimers for TI products. Unless TI explicitly designates a product as custom or customer-specified, TI products are standard, catalog, general purpose devices.

TI objects to and rejects any additional or different terms you may propose.

Last updated 10/2025