Qualcomm Technologies, Inc.

DDR Workflow and Debug Guide

80-P7202-3 C

October 24, 2017

Qualcomm Technologies, Inc.
5775 Morehouse Drive
San Diego, CA 92121
U.S.A.

Revision history

Revision	Date	Description
A	July 2016	Initial release
B	February 2017	Numerous changes were made to this document; it should be read in its entirety.
C	October 2017	Updated SDM660, SDM630 and MSM8998 platform support Numerous changes were made to this document; it should be read in its entirety.

Contents

1 Introduction.. 5

1.1 Purpose.. 5
1.2 Scope.. 5
1.3 Conventions 5
1.4 Technical assistance.. 6

2 DDR setup 7

2.1 DDR configuration 7
2.1.1 DDR parameters customization.. 7
2.1.2 Platform-specific tips... 7
2.1.3 Platform ID 11
2.1.4 Check DSF version number.. 12
2.2 DDR verification.. 12
2.2.1 Supported or unsupported verification 12
2.2.2 PDN, DDR simulation, SI/PI, QDUTT, and QMESA.......................... 14
2.2.3 DDR tests.. 15

3 DDR signature issue 22

3.1 Bit flip(s) (1->0 or 0->1).. 22
3.1.1 1->0 example 22
3.1.2 0->1 example 23
3.1.3 If always bit flip(s) at fixed physical address 23
3.2 Byte or word shifts... 23
3.2.1 Debug suggestion.. 23
3.3 Column corruption.. 23
3.3.1 Debug suggestion.. 24
3.4 DDR lock up 24
3.4.1 Debug suggestion.. 26
3.5 RF desense issue on specific DDR frequency points.................................... 26
3.6 Crash at DDR-related codes... 26
3.6.1 Stuck at DDR training or DDR settings at SBL/XBL 26
3.6.2 Stuck at BIMC Frequencies and BLL configurations of RPM........... 28

4 DDR debug issues.. 30

4.1 DDR frequency.. 30
4.1.1 Check current DDR frequency 30
4.1.2 Check available DDR frequencies.. 31
4.1.3 Limit DDR to a fixed frequency... 37
4.1.4 Adjust specific DDR frequency point.. 39

4.1.5 Change DDR frequency at SBL or XBL 40
4.2 Drive strength 42
4.2.1 Drive strength change 42
4.2.2 Drive strength tuning 65
4.3 DDR voltage 66
4.3.1 LPDDR3 66
4.3.2 LPDDR4 72
4.4 CDC tuning 79
4.4.1 For platforms with DDR boot training 79
4.4.2 For platforms without DDR boot training 84
4.5 Manual CDC margin test 86
4.5.1 Setup 86
4.5.2 Process 87
4.6 Disable hardware self-refresh 88
4.6.1 High-level DDR PHY platforms 88
4.6.2 Mid-level DDR PHY platforms 90
4.7 Disable software self-refresh (Deep sleep) 91
4.7.1 Disable Low-power mode 91
4.7.2 Disable BIMC power collapse and Low-power mode 91
4.7.3 Keep BIMC power domain during Low-power mode 92
4.8 Increase and fix auto refresh rate tREFI 92
4.8.1 High-level DDR PHY platforms 93
4.8.2 Mid-level DDR PHY platforms 93
4.9 Thermal offset changes 93
4.9.1 MSM8994 and MSM8996 94
4.10 Bump up and RBCPR for VddCx/VddMx 94
4.10.1 MSM8926 94
4.10.2 MSM8939 96
4.10.3 MSM8996 102
4.11 Disable periodic training 107
4.11.1 MSM8996 107
4.11.2 MSM8994 108
4.12 DDR decode addressing 108
4.12.1 MSM8974 109
4.12.2 MSM8939 109
4.12.3 MSM8994 109
4.12.4 SDM660 and SDM630 109
4.13 Erase DDR partition 110

A References 111

A.1 Related documents 111
A.2 Acronyms and terms 111

1 Introduction

NOTE: Numerous changes were made to this document; it should be read in its entirety.

1.1 Purpose

The purpose of this document is to identify the following:

- To setup and verify common process for a new DDR chipset
- Describe the crash signature of DDR problems and conclude DDR issues
- Provide general ways to solve DDR issues

1.2 Scope

The document is applicable to new platforms, such as MSM8x10, MSM8x26, MSM8x74, MSM8x16, MSM8x39, MSM8x37, MSM8x17, MSM8x52, MSM8x76, MSM8x53, MSM8x94, and MSM8x96. Since there is no solution to all DDR problems, the document outlines debugging for specific issues of DDR vendors along with ways to verify and debug.

1.3 Conventions

Function declarations, function names, type declarations, and code samples appear in a different font, e.g., `#include`.

Code variables appear in angle brackets, e.g., `<number>`.

Commands to be entered appear in a different font, e.g., **`copy a:*.* b:`**.

Button and key names appear in bold font, e.g., click **Save** or press **Enter**.

If you are viewing this document using a color monitor, or if you print this document to a color printer, red typeface indicates data types, blue typeface indicates attributes, and green typeface indicates system attributes.

Parameter types are indicated by arrows:

→	Depicts an input parameter
←	Depicts an output parameter
↔	Depicts a parameter used for both input and output

1.4 Technical assistance

For assistance or clarification on information in this document, submit a case to Qualcomm Technologies, Inc. (QTI) at https://createpoint.qti.qualcomm.com/.

If you do not have access to the CDMATech Support Service website, register for access or send email to support.cdmatech@qti.qualcomm.com.

2 DDR setup

2.1 DDR configuration

Though Multichip package (MCP) belongs to QTI's verified lists, it is not possible to use it by the default codes (only a JEDEC default CDT XML is released). The customer needs to review and update it based on the DDR specification.

Refer *DDR SDRAM CDT/ECDT User Guide* (80-N1218-1) for more details. The default XML is apt because it supports all JEDEC standard MCPs. Values like tRFC and tXSR need to be reviewed.

2.1.1 DDR parameters customization

2.1.1.1 General rules

- LPDDR2 uses LPDDR2 CDT XML
- LPDDR3 uses LPDDR3 CDT XML
- LPDDR4 uses LPDDR4 CDT XML
- After changing to CDT xml, build new sbl1
- Example steps include the following:

Copy Changed CDT to `boot_images/core/boot/secboot3/scripts,`

```
cd boot_images/core/boot/secboot3/scripts
python cdt_generator.py <CDT>.xml <CDT>.bin
cp boot_cdt_array.c ../../secboot3/hw/msm8xXX/boot_cdt_array.c
```

2.1.2 Platform-specific tips

2.1.2.1 New platforms

New platforms include MSM8930, MSM8960, MSM8x10, MSM8x26, MSM8x74, MSM8x16, MSM8x39, MSM8x37, MSM8x17, MSM8x52, MSM8x76, MSM8x53, MSM8x94, MSM8x96, MSM8x98, MDM9x15, MDM9x25, MDM9x35, and MDM9x45.

1. Refer to *DDR SDRAM CDT/ECDT User Guide* (80-N1218-1) for more details.
2. See Solution CR# 00027431 to modify and generate CDT image and boot_cdt_array.c, change default CDT in sbl1 image (program CDT image is not suggested).
3. Check the following in the vendor specification:
   a. DDR device type

   b. Mode (interleaved or not)

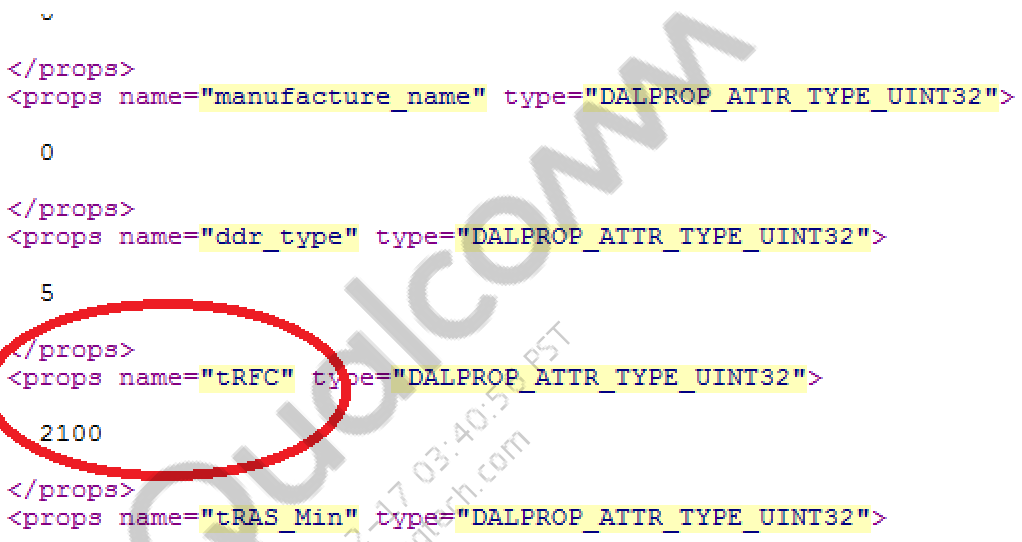
   c. tRFC

   d. tXSR

4. If the specification defines the value as number of clks, the value is num_of_clk * (1/max_frequency) * 10.

An example of tRFC, for > 512MB die, in the DDR specification:

A 210 ns configuration 2100 instead of 1300



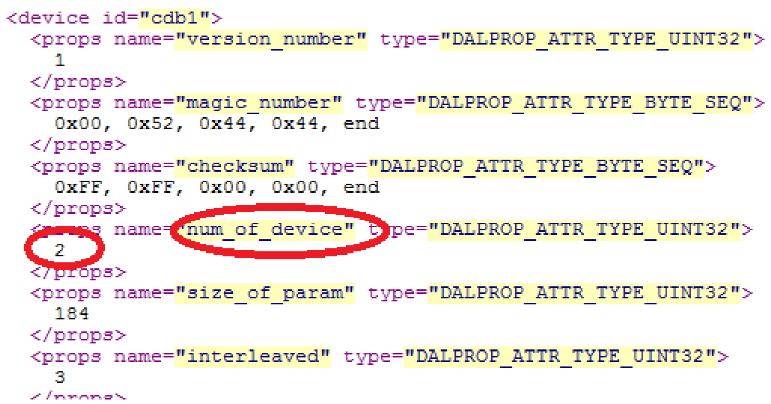
```
  </props>
  <props name="manufacture_name" type="DALPROP_ATTR_TYPE_UINT32">

    0

  </props>
  <props name="ddr_type" type="DALPROP_ATTR_TYPE_UINT32">

    5

  </props>
  <props name="tRFC" type="DALPROP_ATTR_TYPE_UINT32">

    2100

  </props>
  <props name="tRAS_Min" type="DALPROP_ATTR_TYPE_UINT32">
```

2.1.2.2 MSM8x10, MSM8x12, MSM8x26, MSM8x74, and MDM9x25

2.1.2.2.1 CDT XML configuration for multi-DDR

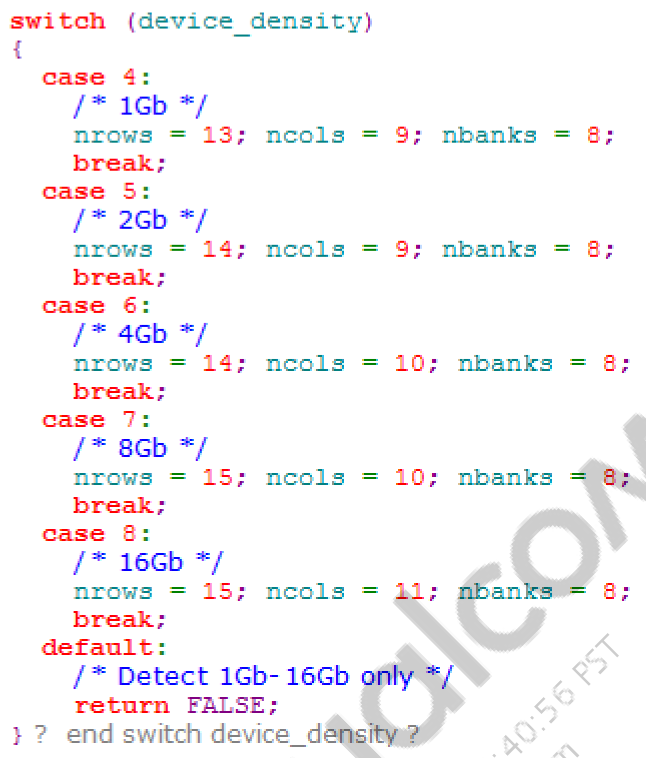
On the B family platforms that are using older BOOT baseline like BOOT.BF.2.1, customers have requirements to use one CDT XML to support different MCPs. In newer BOOT and CDT XML, there is no such requirement as it is suggested to use one CDT XML for all MCPs.

1. Change num_of_device from ‘2’ to ‘4’



```
<device id="cdb1">
  <props name="version_number" type="DALPROP_ATTR_TYPE_UINT32">
    1
  </props>
  <props name="magic_number" type="DALPROP_ATTR_TYPE_BYTE_SEQ">
    0x00, 0x52, 0x44, 0x44, end
  </props>
  <props name="checksum" type="DALPROP_ATTR_TYPE_BYTE_SEQ">
    0xFF, 0xFF, 0x00, 0x00, end
  </props>
  <props name="num_of_device" type="DALPROP_ATTR_TYPE_UINT32">
    2
  </props>
  <props name="size_of_param" type="DALPROP_ATTR_TYPE_UINT32">
    184
  </props>
  <props name="interleaved" type="DALPROP_ATTR_TYPE_UINT32">
    3
  </props>
```

2. Add the cases in HAL_SDRAM_Ram_Size_Detection for the new density of die



```
switch (device_density)
{
  case 4:
    /* 1Gb */
    nrows = 13; ncols = 9; nbanks = 8;
    break;
  case 5:
    /* 2Gb */
    nrows = 14; ncols = 9; nbanks = 8;
    break;
  case 6:
    /* 4Gb */
    nrows = 14; ncols = 10; nbanks = 8;
    break;
  case 7:
    /* 8Gb */
    nrows = 15; ncols = 10; nbanks = 8;
    break;
  case 8:
    /* 16Gb */
    nrows = 15; ncols = 11; nbanks = 8;
    break;
  default:
    /* Detect 1Gb-16Gb only */
    return FALSE;
} ? end switch device_density ?
```

3. Add two more device entry tables in XML; verify whether AC timings, row/column/bank, manufacture IDs are correct

```
The XML structure is:
<device id="cdt_header">
...
</device>
<device id="cdb0">
...
</device>
<device id="cdb1">
<props name="version_number" ...>
... <props name="interleaved" ...>
<props name="device_name" ...> <=== device entry table #1
... manufacture_name

<props name="tDQSCK_max" ...>
<props name="device_name" ...> <=== device entry table #2
...
<props name="tDQSCK_max" ...>
+<props name="device_name" ...> <=== device entry table #3
+... manufacture_name

+<props name="tDQSCK_max" ...>
+<props name="device_name" ...> <=== device entry table #4
+...
```

```
+<props name="tDQSCK_max" ...>
</device>
```

2.1.2.2.2 Specific notes

- About interface_width

Check BIMC_v1.c in SBL1 to confirm if the following line is at the end of HAL_SDRAM_Ram_Size_Detection()

```
        ddr_params->interface_width |= (width << 16);
```

For example, I/O width ID 0x00100020 combined with CS0 I/O width and CS1 I/O width, 16-bits wide CS1 and 32-bits wide CS0, but in CDT XML it is only 32-bits (0x20).

MSM8974, MSM8974SG, MSM8926, MSM8626, MSM8x10, MSM8x12, and MDM9x25 targets using LPDDR3 devices with uneven die interface width, for example, 16-bit wide CS1 and 32-bit wide CS0, need to make appropriate changes in the interface width field in the CDT.xml file.

The hexadecimal representation of the field consists of 8-bits, of which four MSB bits constitute the interface width of CS1 and the rest of the bits represent the interface width of CS0. For the example, the interface width is 0x0010 0020. The decimal equivalent of 0x0010 0020 is 1048608.

Default:

```
<props name="interface_width" type="DALPROP_ATTR_TYPE_UINT32">
  32
</props>

New

<props name="interface_width" type="DALPROP_ATTR_TYPE_UINT32">
  1048608 // this is 0x0010 0020
</props>
```

- Using < 933 MHz DDR MCP on MSM8974

```
some DDR Max freqs is 800 MHz,
Like K3QF6F60MM-QGCE, need remove 933 MHz support too,
    ClockMuxConfigType BIMCClockConfig[] =
    {
    <snip>
    // { 931200000, { HAL_CLK_SOURCE_RAW1, 2, 0, 16, 0 },
CLOCK_VREG_LEVEL_HIGH, 0, CHIPINFO_FAMILY_UNKNOWN, &BIMCPLLConfig[4] }, //
AB & AC
    <snip>
    };
```

- Cross-check the following code and the corresponding WL/RL in DDR specification and verify whether they match with MSM8974 or MSM8926

```
    In rpm/bf/rel/1.2/core/boot/ddr/hw/controller/BIMC_v1.c,
    uint32 lpddr3_rl_max = 14;
    uint32 lpddr3_wl_max = 8;
```

```
Some DDR need use:
uint32 lpddr3_rl_max = 12;
uint32 lpddr3_wl_max = 6;
```

2.1.2.3 Legacy platforms

2.1.2.3.1 MSM60x5

For MSM60x5, refer to solution CR#00006516 Guidelines to configure software for a new NAND/DDR memory for QSC60x5 device.

2.1.2.3.2 MDM6270 and MSM6185

Refer to *Guidelines New NAND_RAM Configuration in QSC6270/QSC6240* (80-VN648-1) for more details.

2.1.2.3.3 MSM6x00 and MSM61x5

- See the dbl_ddr_autodetect.c, Hynix MCP is not autodetect. Hence, disable the FEATURE_AUTODETECT_DDR.
- See the function dbl_get_ddr_on_ebi1(), which branch it runs. It uses DBL_DDR_ON_EBI1_MICRON by default. Hence, the parameter is in the dbl_ddr_micron.c.
- Refer to *Guidelines For New NAND_RAM Configuration in QSC6270/QSC6240* (80-VN648-1) and *Flash Memory Customization Guide* (80-VR646-1) for configuring a new MCP. The parameters need to be modified as per the MCP specification, when a new MCP is used.

2.1.3 Platform ID

See solution # 00027431; "platform_id" accessed by LK boot loader and the Linux kernel and is used to identify the hardware platform and load the correct device tree.

Take MSM8974 as an example, the most important are the first 2 bytes. Verify if it is 0x3, 0x8 if you use MTP reference design.

```
<props name="platform_id" type="DALPROP_ATTR_TYPE_BYTE_SEQ">
0x03,0x08,0x01,0x00,0x00,0x03,0x01,0x01,0x02,0x01,0x03,0x00,end
</props>
 0x3 - Version
0x8 --> MTP
0x1-> hw major version
0x0-> hw minor version
0x0-> subtype.
0x3 means total 3 Key value pairs
KVPs
0x1 PLATFORMINFO_KEY_DDR_FREQ --> 0x1 (933), set to 0x1 will enable DDR
933MHZ
0x2 PLATFORMINFO_KEY_GFX_FREQ --> 0x1 (550), set to 0x1 to enable GPU
550MHZ
```

```
0x3 PLATFORMINFO_KEY_CAMERA_FREQ -> 0x0 , not used yet.
```

2.1.4 Check DSF version number

For later platforms like the MSM8996 or MSM8994 or MSM8953, there is a unified control for DDR-related drivers and softwares in SBL/XBL. Hence, check the latest DSF version number as follows:

NOTE: This is a critical point to confirm whether all the latest updates of QTI DDR changes and CRs are included.

#define **TARGET_DDR_SYSTEM_FIRMWARE_VERSION** xxx

SBL:

```
boot_images/core/boot/ddr/hw/hw_sequence/89xx/target/89xx/settings/target_c
onfig.h
```

For the SDM660, SDM630, MSM8996, and MSM8998 chipsets, the DSF version are divided into two parts: one is XBL and the other is RPM.

XBL:

#define **TARGET_DDR_SYSTEM_FIRMWARE_VERSION**

```
boot_images/QcomPkg/Msm8996Pkg/Library/DSFTargetLib/target/89xx/settings/ta
rget_config.h
```

the directory may change on different platforms and different meta builds, but the macro name should be the same

RPM:

`#define` **TARGET_DDR_SYSTEM_FIRMWARE_RPM_VERSION**

```
rpm_proc\core\boot\ddr\hw\hw_sequence\rpm\target\SDM630_660\settings/target
_config_rpm.h
```

2.2 DDR verification

2.2.1 Supported or unsupported verification

2.2.1.1 General rules

The following queries exist:

- Whether xxxx part MCP on MSMxxxx is supported?
- Whether 3 GB, 4 GB, 6 GB or 8 GB is supported?

All JEDEC standard DDRs are supported, including the large densities like 4 GB on the MSM8916/MSM8939/MSM8952/MSM8976 chipsets and 6 GB, 8 GB on the MSM8996 chipset.

Verify the following cases with QTI:

- The MCP is a new product and has not been used before. This is due to the special die structure of MCP such as SS 12 GB single rank DDR on the MSM8996 chipset; some time is required to add support for it.
- The MCP has special density for all the used MCPs on the platforms such as 3 GB or 4 GB on the MSM8939 chipset and 1.5 GB on the MSM8916 chipset.

For such special requirements from OEMs, verify with QTI. It may take some time to verify and add software changes for it.

2.2.1.2 Specific examples

2.2.1.2.1 MSM8939 3 GB

Hardware solution for adding external buck for DDR power to meet 3 GB DDR (only for SS 3 GB part, 4*768 structure; no need to add external buck for hynix 2G+1G structure) power specification. Refer to *Proposed Solution For 3 GB LPDDR3 in MSM8939* (80-NK808-20) for more details. For example, KMR31000BA-B614 IDD4R out specification (VDDQ+VDD2>600mA) requires external buck.

Software solution for porting external buck driver – PMIC team support, including kernel and sbl1 changes.

3 GB DDR changed system memory mapping. Verify customer baseline including 3 GB supported CRs. It is recommended that the customer upgrades to 3 GB supported baseline that is verified on MTP by the QIPL team.

SS 3 GB(1.5G + 1.5G) DDR address mapping is 0x20000000 – (0xE0000000 – 1)

Hynix 3 GB(2G + 1G) DDR address mapping is 0x40000000 – (0x100000000 – 1) (CS1 1 GB 0x40000000~ 0x7FFFFFFF, CS0 2 GB 0x80000000 – (0x100000000 – 1))

3 GB DDR CRs:

```
794506 SBL1 Built [8939] Support 3GB DDR configuration
794346 SBL1 Built [8939] Support 3GB DDR configuration
794941 APPS Built Add support for 3GB DDR for 8939
799350 MPSS Built [8939] Support 3GB DDR configuration from ICB standpoint
- MBA
794957 TZ Built 3.5 GB DDR support in Trustzone
798072 TZ [8939] Support 3GB DDR configuration from ICB standpoint – TZ
```

2.2.1.2.2 MSM8916 and MSM8939 4 GB

- Only a maximum of 3.75 GB can be used if a 4 GB DDR chipset is used.
- The following CRs are included (SBL, TZ, and Linux):
  - CR: 900928
  - CR: 903470
  - CR: 867493
  - CR: 807479
  - CR: 786714

2.2.1.2.3 MSM8996 6 GB

The MSM8996 6 GB chipset are supported after merging the following patches and DSF versions.

- XBL/RPM - CR: 1005482
- TZ - CR: 969504
- LK - CR: 961081

2.2.2 PDN, DDR simulation, SI/PI, QDUTT, and QMESA

2.2.2.1 PDN

PDN is mandatory for a new product; if PDN is not passed then any random stability issue in any subsystem is expected.

2.2.2.2 DDR simulation

DDR simulation is mandatory for a new product. If the results are not aligned with the default settings, such as MR3 and ROUT values, the following steps are performed:

1. Raise case to QTI to review
2. If it has been agreed to be changed, see Section 4.2 for modification.

2.2.2.3 Fully SI/PI report

SI/PI is suggested for a new product. When DDR-related stability issues are detected (refer to section 3), SI/PI is necessary. Contact the DDR vendor to provide complete SI/PI reports and check with QTI.

2.2.2.4 QDUTT

QDUTT is necessary for any platform, which QDUTT supports, including the MSM8916, MSM8939, MSM8952, MSM8976, MSM8937, and MSM8953 chipsets. For platforms that have no boot trainings, QDUTT results are required to be applied to the default ones. For modifications of CDC delay, see section 4.4. For platforms that have boot trainings, the trained results are followed; QDUTT results are just references. For more details, see Section 2.2.3.

2.2.2.5 QMESA

Stress tests like QMESA or memtester are mandatory for a new product. For details, see Section 2.2.3.

2.2.3 DDR tests

For porting a new DDR, it is mandatory to run a complete memory verification test to help find related problems. Refer to *QRD Hardware Component Verification Schedule* (80-VL976-6) for reference on how to build test cases

- DDR max frequency stress test
- DDR sleep/wake-up stress test

It is recommended to use QDUTT and QMESA to perform the DDR CDC scan and stress test.

2.2.3.1 QDUTT

Purpose: Signal margin test by cdc scan

Supported platforms: MSM8909, MSM8916, MSM8917, MSM8939, MSM8937, MSM8940, MSM8952, MSM8976, MSM8953, SDM630, SDM660, MSM9x07, and MSM9x06

Website address to download: https://createpoint.qti.qualcomm.com/tools/

Use the detailed user guide placed at C:\Program Files (x86)\Qualcomm\QDUTT\doc\userguide.pdf and get CDC scan results. Raise an issue with QTI to review if there are any questions.

A detailed document for VREF SHMOO tests is also provided. This is from the MSM8909 chipset; for MSM8909 it is mandatory and for other platforms, you can refer to it as an option. Refer to *MSM8909/APQ8009 DDR Bitflip Issue* (80-NP408-14) and

MSM8909 DDR Test Instructions (80-NP408-15) for more information.

The following QDUTT chinese docs are used as reference docs:

DCN	Title
80-NR058-2SC	*QDUTT Application Notes For MSM8917/8937/8940/8952/8953/8976/8976pro Simplified Chinese*
80-NR058-3SC	*QDUTT Tool and DDR Test Training Simplified Chinese*
80-NR058-5SC	*QDUTT Application Notes For MSM8909/MSM8916 – Simplified Chinese*
80-PC982-1SC	*SDM660 SDM630 Boot DDR Debug Image User Guide Simplified Chinese*

2.2.3.2 QMESA

Purpose: User space general stress tests

QMESA is a user space tool for smart devices (Android and MP). Raise a case if you have not got the QMESA binary.

Example steps:

```
X:\>adb root
X:\>adb push QMESA_64 /data/local/tmp/QMESA_64
X:\>adb shell chmod 777 /data/local/tmp/QMESA_64
X:\>adb shell
root@msm8996:/ # /data/local/tmp/QMESA_64 > /data/local/tmp/qmlog1.txt &
```

Usually default QMESA parameters are sufficient, use “--help” to check details of parameters.

Also, apply *QMESA: Qualcomm Memory Stress Application User Guide* (80-NH759-1) from your TAMs.

2.2.3.2.1 Notes

- QB/QMESA cannot state decipher cache or DDR issues.

For example, a cache issue like high temperature for failure samples (MSM™); looks same as QB error logs DDR issues

- Distinguish a cache problem

For cache issues, analyze multiple AP crash dumps, such as the value in vmlinux and D.dump. In dmesg buffer or CPU regs, there are bitflips, then it can be a cache issue.

That is out of the DDR scope.

- Example of failure
  - Obvious bit flips

Like the following example, there are some signatures like "(@phys=0xE4C6F2B8), error mask=0x80000000"

Hence, the actual bit flip is found at position DQ31 (bit31 indicated by error mask), and also the physical address is printed.

See Sections 3.1.3 and 4 for further checking.

```
QMESA_32 Version 2.30 Qualcomm MEmory Stress Application
QUALCOMM Proprietary
Copyright (c) 2013-2015, QUALCOMM Incorporated. All rights reserved.
Starting system time is Thu Jan 1 00:11:40 1970
The following configuration is being run:
startSize 2MB
endSize 16MB
errorCheck TRUE
secs 1000000
numThreads 2
MemTest : random generator using Seed=0x61C4DE31
Membench0: random generator using Seed=0x942A2152
Membench1: random generator using Seed=0xC68F6473
Random DDR Test loaded successfully
The following tests are being run:
CP_CLIB
CP_VNUM_BIONICK
CP4I_CLIB
CP4I_ASM
CP4I_VNUM
CP4S_CLIB
CP4S_ASM
CP4S_VNUM
CP4I_COHERENCY
-------------------------STARTING STRESS TESTING------------------
```

```
............................................................................
.....
.
............................................................................
.....
.
.....................................................
2000 Tests: RandDDR=2000, elapsed time=00:10:32, sys time=Thu Jan 1
00:22:12 19
70
............................
............................................................................
.....
.
............................................................................
.....
.
.......................................................................veri
fy_da
ta: QMESA_32 Version 2.30 Qualcomm MEmory Stress Application
verify_data: retry=0, workbuf[0x0], dblk[0x0], bufferid=0x00000000,
calculated C
RC=0x5FFB3E7F, expected CRC=0x222F79DB
verify_data: retry=0, workbuf[0x0] found 0x1 dblks with failing CRC out of
0x1
verify_data: retry=1, workbuf[0x0], dblk[0x0], bufferid=0x00000000,
calculated C
RC=0x5FFB3E7F, expected CRC=0x222F79DB
verify_data: retry=1, workbuf[0x0] found 0x1 dblks with failing CRC out of
0x1
verify_data: retry=2, workbuf[0x0], dblk[0x0], bufferid=0x00000000,
calculated C
RC=0x5FFB3E7F, expected CRC=0x222F79DB
verify_data: retry=2, workbuf[0x0] found 0x1 dblks with failing CRC out of
0x1
verify_data: workbuf[0x0] miscompared all 3 retries

verify_data: Buffer Failure Stats:
verify_data:  workbuf[0x0] failed 0x1 of 0x1 dblks
verify_data:  workbuf[0x1] failed 0x0 of 0x1 dblks

verify_data: Analzing 1st fail in workbuf[0x0], dblk[0x0]
analyze_data: Analyzing CRC32 error: workbuf[0x0], dblk[0x0] contains data
from
…
…
6af0f4, shadow buf @phys=0xc677e0f4
Shadow vs Working BufferID OK,  shadow buf[1]=0x0, work buf[1]=0x0
CRC32 OK:  shadow buf[0]=0x222F79DB,    work buf[0]=0x222F79DB
```

```
CRC32 ERROR:  work buf[0]=0x222F79DB,  work buf recalc=0x5FFB3E7F
CRC32 OK:  shadow buf[0]=0x222F79DB, shadow buf recalc=0x222F79DB


====================== Compare every byte of the work and shadow buffers of
fail
ing block ====================

********************** Printing out error #1 with surrounding data
*************
******************************
    offset=0x00334194, shadow expect=0x5DBA890D (@phys=0xC7663288),
workbuf=
0x5DBA890D (@phys=0xE4C6F288), error mask=0x00000000
…
…

***err**offset=0x003341C4, shadow expect=0xB2E72128 (@phys=0xC76632B8),
workbuf=
0x32E72128 (@phys=0xE4C6F2B8), error mask=0x80000000
    offset=0x003341C8, shadow expect=0x1A3E87A7 (@phys=0xC76632BC),
workbuf=
0x1A3E87A7 (@phys=0xE4C6F2BC), error mask=0x00000000
    offset=0x003341CC, shadow expect=0xCC5ED97A (@phys=0xC76632C0),
workbuf=
0xCC5ED97A (@phys=0xE4C6F2C0), error mask=0x00000000
    offset=0x003341D0, shadow expect=0xD9E6FB94 (@phys=0xC76632C4),
workbuf=


analyze_data: Work/Shadow buffer compare found 1 word errors out of 1775731
word
s total


====================== Running classical IFA-13 Fault test on failing block
in a
ll work buffers ================
    Running IFA-13 Fault tests on workbuf[0x0], dblk[0x0] @phys=0xc96af0f4
f
or 0x6C61CD bytes
    Running IFA-13 Fault tests PASSED
    Running IFA-13 Fault tests on workbuf[0x1], dblk[0x0] @phys=0xc7a33515
f
or 0x6C61CD bytes
    Running IFA-13 Fault tests PASSED

ERROR...elapsed test time=00:39:38, sys time=Thu Jan 1 00:51:18 1970
```

- No obvious bit flip by err mask
  - Cannot find “@phys” or “error mask” in the logs
  - Only some CRC errors might be the reading problem of DDR
  - Refer to Section 4 to adjust MR3. This should be the first option.

```
QMESA_64 Version 2.30 Qualcomm MEmory Stress Application
QUALCOMM Proprietary
Copyright (c) 2013-2015, QUALCOMM Incorporated. All rights reserved.
Starting system time is Thu Jan 1 00:28:38 1970

The following configuration is being run:
startSize 2MB
endSize 32MB
errorCheck TRUE
secs 36000
numThreads 2
MemTest : random generator using Seed=0x3199C4D
Membench0: random generator using Seed=0x357EDF6E
Membench1: random generator using Seed=0x67E4228F
Random DDR Test loaded successfully
The following tests are being run:
CP_CLIB
CP_VNUM_BIONICK
CP_VNUM_A5X
CP4I_CLIB
CP4I_ASM
CP4I_VNUM
CP4S_CLIB
CP4S_ASM
CP4S_VNUM
CP4I_COHERENCY
-------------------------STARTING STRESS TESTING------------------
...........................................................................
......
...........................................................................
......
...........................................................................
......
...........................................................................
......
...........................................................................
......
....................
4000 Tests: RandDDR=4000, elapsed time=00:24:53, sys time=Thu Jan 1
00:53:32 1970
.............................................................
...........................................................................
......
```

```
..............................................................
6000 Tests: RandDDR=6000, elapsed time=00:37:37, sys time=Thu Jan 1
01:06:16 1970
..........
...................................................................
......
.............verify_data: QMESA_64 Version 2.30 Qualcomm MEmory Stress
Application
verify_data: retry=0, workbuf[0x2], dblk[0x2F], bufferid=0x0002002F,
calculated CRC=0x77CE2A7B, expected CRC=0xCC53209C
verify_data: retry=0, workbuf[0x2] found 0x1 dblks with failing CRC out of
0x95
verify_data: retry=1, workbuf[0x2], dblk[0x2F], bufferid=0x0002002F,
calculated CRC=0x77CE2A7B, expected CRC=0xCC53209C
verify_data: retry=1, workbuf[0x2] found 0x1 dblks with failing CRC out of
0x95
verify_data: retry=2, workbuf[0x2], dblk[0x2F], bufferid=0x0002002F,
calculated CRC=0x77CE2A7B, expected CRC=0xCC53209C
verify_data: retry=2, workbuf[0x2] found 0x1 dblks with failing CRC out of
0x95
verify_data: workbuf[0x2] miscompared all 3 retries

verify_data: Buffer Failure Stats:
verify_data:  workbuf[0x0] failed 0x0 of 0x95 dblks
verify_data:  workbuf[0x1] failed 0x0 of 0x95 dblks
verify_data:  workbuf[0x2] failed 0x1 of 0x95 dblks
verify_data:  workbuf[0x3] failed 0x0 of 0x95 dblks

verify_data: Analzing 1st fail in workbuf[0x2], dblk[0x2F]
analyze_data: Analyzing CRC32 error: workbuf[0x2], dblk[0x2F] contains data
from BufferID 0x0002:002F, len=0x5600 bytes
analyze_data: Test name=RNDD_12MB_CP4I_ASM, test_bytes=0xC83800,
dbyte=0x5600, dblk=0x95, dblkstride=0xA4, loop=0x1, adrstride=0x3D4


====================== Recheck CRC on the work and shadow buffers of
failing block ======================
Analyzing with potentially cached data for workbuf @phys=0x14a734ad0,
shadow buf @phys=0x14a734ad0
Shadow vs Working BufferID OK,  shadow buf[1]=0x2002F, work buf[1]=0x2002F
CRC32 OK:  shadow buf[0]=0xCC53209C,    work buf[0]=0xCC53209C
CRC32 ERROR:  work buf[0]=0xCC53209C,  work buf recalc=0x77CE2A7B
CRC32 ERROR: shadow buf[0]=0xCC53209C, shadow buf recalc=0x77CE2A7B

Flushing all caches to insure DDR is being used for analysis, workbuf
@phys=0x14a734ad0, shadow buf @phys=0x14a734ad0
Shadow vs Working BufferID OK,  shadow buf[1]=0x2002F, work buf[1]=0x2002F
CRC32 OK:  shadow buf[0]=0xCC53209C,    work buf[0]=0xCC53209C
CRC32 ERROR:  work buf[0]=0xCC53209C,  work buf recalc=0x77CE2A7B
```

```
CRC32 ERROR: shadow buf[0]=0xCC53209C, shadow buf recalc=0x77CE2A7B

…
…


====================== Compare every byte of the work and shadow buffers of
failing block ====================

analyze_data: Work/Shadow buffer compare found 0 word errors out of 5504
words total

====================== Running classical IFA-13 Fault test on failing block
in all work buffers ================
    Running IFA-13 Fault tests on workbuf[0x0], dblk[0x2F] @phys=0xa96ea840
for 0x5600 bytes
    Running IFA-13 Fault tests PASSED
    Running IFA-13 Fault tests on workbuf[0x1], dblk[0x2F]
 …

    Running IFA-13 Fault tests on workbuf[0x3], dblk[0x2F] @phys=0xa8964c18
for 0x5600 bytes
    Running IFA-13 Fault tests PASSED

ERROR...elapsed test time=01:08:29, sys time=Thu Jan 1 01:37:08 1970
```

2.2.3.3 JTAG DDR test

1. Break at the necessary instance, such as system hang or the test case.

   It should be after DDR init of SBL/XBL.

2. Use the following cmd a TRACE32 (the address range is just an example). Adjust as per the requirements.

   ```
   data.test 0x100000--0x200000 /PRIME /RANDOM /REPEAT 1000
   ```

3 DDR signature issue

When a crash dump is encountered in the kernel or modem area, the AP or modem stability teams are required to triage it first.

There are some error patterns suspected to be linked to a DDR issue. It is not possible to confirm if it is a DDR issue for general memory corruptions without obvious DDR error patterns.

3.1 Bit flip(s) (1->0 or 0->1)

Usually, DDR bit flips only happen on bit positions. If an error pattern is encountered, which is different to the expected values, it is suspected to be linked to the memory corruption of the software and is not a DDR issue.

3.1.1 1->0 example

In page_corruption_summary.txt by "--print-pagealloccorruption"

NOTE: For more details, check with the AP stability team.

```
Single Bit Error at 0x02191ec0
0x02191ec0 0xa8aaaaaa 0xaaaaaaaa 0xaaaaaaaa 0xaaaaaaaa 0xaaaaaaaa 0xaaaaaaaa 0xaaaaaaaa
0xaaaaaaaa
0x02191ee0 0xaaaaaaaa 0xaaaaaaaa 0xaaaaaaaa 0xaaaaaaaa 0xaaaaaaaa 0xaaaaaaaa 0xaaaaaaaa
0xaaaaaaaa

Single Bit Error at 0x02191ec0
0x02191ec0 0xa8aaaaaa 0xaaaaaaaa 0xaaaaaaaa 0xaaaaaaaa 0xaaaaaaaa 0xaaaaaaaa 0xaaaaaaaa
0xaaaaaaaa

Single Bit Error at 0x0397ee38
0x0397ee38 0xaaa8aaaa 0xaaaaaaaa 0xaaaaaaaa 0xaaaaaaaa 0xaaaaaaaa 0xaaaaaaaa 0xaaaaaaaa
0xaaaaaaaa
```

3.1.1.1 Debug suggestion

See debugging steps listed in Section 4.

If all the dumps show only 1->0, it is suspected that this is a DDR retention issue linked to temperatures. See sections 4.6, 4.7, 4.8, and 4.9 for more information.

3.1.2 0->1 example

Page 4288 has 2 bit-flip [0xffffffffffffffff --> 0xfffffffffffffffb] and [0xffffffffffffffff --> 0xffffdfffffffffff]
Page 4160 has 1 bit-flip [0xffffffffffffffff --> 0xffffffffffefffff]
Page 4157 has 1 bit-flip [0xffffffffffffffff --> 0xfffffffeffffffff]

3.1.2.1 Debug suggestion

See debugging steps in Section 4.

3.1.3 If always bit flip(s) at fixed physical address

3.1.3.1 Debug suggestion

Refer to the steps listed in Section 4.12 for more information on decode addressing.

3.2 Byte or word shifts

Bit flip is the most frequent DDR issue; byte/word shifts is a rare problem, such as CR667763 on the MSM8974 chipset.

```
C10134D0| 00000000 00000000 00[48]0000 00CE0000 <<< 3rd byte value 0x48
shifted left by 2 words
C10134E0|>EB[0A]EB49 F000FF00 FA006470 FA006490 <<<< Lock value is
0xEB[0A]EB49, but should be 0xEB[48]EB49
C10134F0| FA006530 0A000AE1 00480000 FFCEFFFF
```

3.2.1 Debug suggestion

See debugging steps listed in Section 4.

3.3 Column corruption

Column corruption is also a rare problem. Example (expected value are all “a”)

- <3>[3980.101828] pagealloc: memory corruption
- <3>[3980.101976] fff20400: 41 aa 41 aa 01 aa 41 aa 40 aa 05 aa 41 aa 41 aa A.A...A.@...A.A.
- <3>[3980.102114] fff20410: 41 aa 45 aa 45 aa 45 aa 51 aa 55 aa 51 aa 41 aa A.E.E.E.Q.U.Q.A.
- <3>[3980.102330] fff20420: 41 aa 41 aa 51 aa 45 aa 44 aa 01 aa 15 aa 41 aa A.A.Q.E.D.....A.
- <3>[3980.102552] fff20430: 44 aa 40 aa 41 aa 51 aa 11 aa 45 aa 01 aa 41 aa D.@.A.Q...E...A.
- <3>[3980.102663] fff20440: 40 aa 45 aa 41 aa 41 aa 51 aa 45 aa 45 aa 41 aa @.E.A.A.Q.E.E.A.
- <3>[3980.102876] fff20450: 41 aa 01 aa 41 aa 51 aa 51 aa 44 aa 41 aa 45 aa A...A.Q.Q.D.A.E.
- <3>[3980.103067] fff20460: 01 aa 41 aa 04 aa 51 aa 41 aa 51 aa 45 aa 41 aa ..A...Q.A.Q.E.A.
- <3>[3980.103254] fff20470: 45 aa 45 aa 41 aa 45 aa 41 aa 41 aa 00 aa 01 aa E.E.A.E.A.A.....
- <3>[3980.103345] fff20480: 41 aa 41 aa 41 aa 15 aa 41 aa fb aa 40 aa 41 aa A.A.A...A...@.A.

3.3.1 Debug suggestion

See the debugging steps listed in Section 4.

3.4 DDR lock up

Resource power manager (RPM) maintains a global variable ddr_register_cache to track the BIMC status registers to check if there are any pending transactions or transactions not seen in the memory dump.

It is possible that it is a lock up by the following patterns:

- An AP sec or non-sec WD bite
- Crashscope/QCAP report for ddr_register_cache, not all are "1" in the SCMO_RCH and SCMO_WCH
- Most DDR lock ups are seen on the MSM8994 platform:

MSM8994 example:

```
BIMC_S_DDR0_SCMO_RCH_STATUS = 0x00011111,
BIMC_S_DDR1_SCMO_RCH_STATUS = 0x1101,
```

- Rare DDR lock ups are seen on other platforms, like the MSM8939 and MSM8952 chipsets

MSM8952 example:

```
ddr_register_cache
========================
SCMO_RCH: 0x00011011 | 69649
SCMO_WCH: 0x00000111 | 273
```

MSM8996 example:

On the MSM8996 platform, for a suspected DDR lockup:

- The two groups registers should be the same
- Should have some abnormal values in BIMC_M_APP_MPORT_STATUS_0A and BIMC_S_DDR0_DPE_MEMC_STATUS_0

```
ddrsns_share_data = 0x000A2000 -> (
version = 2194 = 0x0892, <--------------- DSF 219.4
status_registers = (
(
BIMC_S_DDR0_SCMO_RCH_STATUS = 0x1111, <------------ read pending
BIMC_S_DDR1_SCMO_RCH_STATUS = 0x00011111,
BIMC_S_DDR0_SCMO_WCH_STATUS = 0x0111,
BIMC_S_DDR1_SCMO_WCH_STATUS = 0x0111,
BIMC_S_DDR0_SCMO_CMD_BUF_STATUS = 0x0,
BIMC_S_DDR1_SCMO_CMD_BUF_STATUS = 0x0,
BIMC_S_DDR0_DPE_DRAM_STATUS_0 = 0x03000000,
BIMC_S_DDR1_DPE_DRAM_STATUS_0 = 0x03000000,
BIMC_S_DDR0_DPE_MEMC_STATUS_0 = 0xCCCC, <------------ DQ line pending
BIMC_S_DDR1_DPE_MEMC_STATUS_0 = 0xFFFF,
BIMC_S_SYS0_SWAY_STATUS_0A = 0x7,
BIMC_S_SYS0_SWAY_STATUS_0B = 0x7,
BIMC_S_SYS1_SWAY_STATUS_0A = 0x0,
BIMC_S_SYS1_SWAY_STATUS_0B = 0x0,
BIMC_M_APP_MPORT_STATUS_0A = 0x7, <------------ 7 read commands from APSS
are pending
BIMC_M_APP_MPORT_STATUS_2A = 0x0,
BIMC_M_APP_MPORT_STATUS_2B = 0x0,
BIMC_M_APP_MPORT_STATUS_4A = 0x0,
BIMC_M_APP_MPORT_STATUS_4B = 0x0,
BIMC_M_APP_MPORT_STATUS_6A = 0x0,
BIMC_M_APP_MPORT_STATUS_6B = 0x0,
BIMC_M_GPU_MPORT_STATUS_0A = 0x0,
BIMC_M_GPU_MPORT_STATUS_2A = 0x0,
BIMC_M_GPU_MPORT_STATUS_2B = 0x0,
BIMC_M_GPU_MPORT_STATUS_4A = 0x0,
BIMC_M_GPU_MPORT_STATUS_4B = 0x0,
BIMC_M_GPU_MPORT_STATUS_6A = 0x0,
BIMC_M_GPU_MPORT_STATUS_6B = 0x0,
BIMC_M_SYS_MPORT_STATUS_0A = 0x1,
BIMC_M_SYS_MPORT_STATUS_2A = 0x1,
BIMC_M_SYS_MPORT_STATUS_2B = 0x1,
BIMC_M_SYS_MPORT_STATUS_4A = 0x0,
BIMC_M_SYS_MPORT_STATUS_4B = 0x0,
BIMC_M_SYS_MPORT_STATUS_6A = 0x0,
BIMC_M_SYS_MPORT_STATUS_6B = 0x0,
BIMC_MC_TIMESTAMP_COUNTER_CURRENT = 0x000B8F13),
(
BIMC_S_DDR0_SCMO_RCH_STATUS = 0x1111,
BIMC_S_DDR1_SCMO_RCH_STATUS = 0x00011111,
BIMC_S_DDR0_SCMO_WCH_STATUS = 0x0111,
BIMC_S_DDR1_SCMO_WCH_STATUS = 0x0111,
BIMC_S_DDR0_SCMO_CMD_BUF_STATUS = 0x0,
BIMC_S_DDR1_SCMO_CMD_BUF_STATUS = 0x0,
BIMC_S_DDR0_DPE_DRAM_STATUS_0 = 0x03000000,
BIMC_S_DDR1_DPE_DRAM_STATUS_0 = 0x03000000,
BIMC_S_DDR0_DPE_MEMC_STATUS_0 = 0xCCCC,
BIMC_S_DDR1_DPE_MEMC_STATUS_0 = 0xFFFF,
BIMC_S_SYS0_SWAY_STATUS_0A = 0x7,
BIMC_S_SYS0_SWAY_STATUS_0B = 0x7,
BIMC_S_SYS1_SWAY_STATUS_0A = 0x0,
BIMC_S_SYS1_SWAY_STATUS_0B = 0x0,
BIMC_M_APP_MPORT_STATUS_0A = 0x7,
BIMC_M_APP_MPORT_STATUS_2A = 0x0,
BIMC_M_APP_MPORT_STATUS_2B = 0x0,
BIMC_M_APP_MPORT_STATUS_4A = 0x0,
BIMC_M_APP_MPORT_STATUS_4B = 0x0,
BIMC_M_APP_MPORT_STATUS_6A = 0x0,
BIMC_M_APP_MPORT_STATUS_6B = 0x0,
BIMC_M_GPU_MPORT_STATUS_0A = 0x0,
BIMC_M_GPU_MPORT_STATUS_2A = 0x0,
BIMC_M_GPU_MPORT_STATUS_2B = 0x0,
BIMC_M_GPU_MPORT_STATUS_4A = 0x0,
BIMC_M_GPU_MPORT_STATUS_4B = 0x0,
BIMC_M_GPU_MPORT_STATUS_6A = 0x0,
BIMC_M_GPU_MPORT_STATUS_6B = 0x0,
BIMC_M_SYS_MPORT_STATUS_0A = 0x1,
BIMC_M_SYS_MPORT_STATUS_2A = 0x1,
BIMC_M_SYS_MPORT_STATUS_2B = 0x1,
BIMC_M_SYS_MPORT_STATUS_4A = 0x0,
BIMC_M_SYS_MPORT_STATUS_4B = 0x0,
BIMC_M_SYS_MPORT_STATUS_6A = 0x0,
BIMC_M_SYS_MPORT_STATUS_6B = 0x0,
BIMC_MC_TIMESTAMP_COUNTER_CURRENT = 0x000B905B)),
ddi_buf = (0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0,
0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0
```

3.4.1 Debug suggestion

See the debugging steps in listed Section 4.

3.5 RF desense issue on specific DDR frequency points

RF like GP, WLAN, GSM, and LTE has desense issues on frequency points of DDR frequency, Debug suggestion

See Sections 4.1.3 and 4.1.4.

3.6 Crash at DDR-related codes

3.6.1 Stuck at DDR training or DDR settings at SBL/XBL

- Example of training stuck

```
Format: Log Type - Time(microsec) - Message
Log type: B - since boot(excluding boot rom). D - delta
B - 174460 - SBL1, Start
B - 179706 - scatterload_region && ram_init, Start
D - 30 - scatterload_region && ram_init, Delta
B - 196908 - pm_device_init, Start
D - 19032 - pm_device_init, Delta
B - 216062 - boot_flash_init, Start
D - 30 - boot_flash_init, Delta
B - 219996 - boot_config_data_table_init, Start
D - 79666 - boot_config_data_table_init, Delta
B - 304359 - sbl1_ddr_set_params, Start
D - 0 - sbl1_ddr_set_params, Delta
B - 312320 - pm_driver_init, Start
D - 12474 - pm_driver_init, Delta
```

```
B - 325862 - cpr_init, Start
D - 50233 - cpr_init, Delta
B - 376126 - Pre_DDR_clock_init, Start
D - 61 - Pre_DDR_clock_init, Delta
```
`B - 382317 - do_ddr_training, Start`

- Example of DDR param stuck

```
Log type: B - since boot(excluding boot rom). D - delta
B - 179248 - SBL1, Start
B - 184494 - scatterload_region && ram_init, Start
D - 0 - scatterload_region && ram_init, Delta
B - 201452 - pm_device_init, Start
D - 27328 - pm_device_init, Delta
B - 228872 - boot_flash_init, Start
D - 27663 - boot_flash_init, Delta
B - 256932 - boot_config_data_table_init, Start
D - 46634 - boot_config_data_table_init, Delta
B - 305122 - sbl1_ddr_set_params, Start
```

3.6.1.1 Debug suggestion

- Check hardware connections from hardware side
- Review all CDT XML configurations, see Section 2.1
- See Section 2.2.1; analyze whether if any special requirements are needed to support that, such as new CRs or external power supply buck helper
- Use JTAG to locate the exact stuck place
- Use JTAG to test DDR, see Section 2.2.3
- Use JTAG to check the DDR detection whether the DDR detection density is as expected

Debug the Var “boot_ddr_size_info” or “ddrsns_share_data.ddr_size_info”

If the density is incorrect, raise a case to QTI for co-debugging.

Take MSM8996 as an example:

```
boot_images/QcomPkg/Msm8996Pkg/Library/DDRTargetLib/ddr_target.c
void ddr_remapper(void)
{
<snip>

ddrsns_share_data->ddr_size_info.ddr0_cs0_remapped_addr =
remap_info.dest_addr;
ddrsns_share_data->ddr_size_info.ddr1_cs0_remapped_addr =
remap_info.dest_addr;
ddrsns_share_data->ddr_size_info.ddr0_cs1_remapped_addr =
ddrsns_share_data->ddr_size_info.ddr0_cs1_addr;
ddrsns_share_data->ddr_size_info.ddr1_cs1_remapped_addr =
ddrsns_share_data->ddr_size_info.ddr1_cs1_addr;

} /* ddr_configure_lpae */
```

```
e.g.
ddrsns_share_data.ddr_size_info = (
ddr0_cs0_mb = 0x0300,
ddr0_cs1_mb = 0x0300,
ddr1_cs0_mb = 0x0300,
ddr1_cs1_mb = 0x0300,
ddr0_cs0_addr = 0x0,
ddr0_cs1_addr = 0x80000000,
ddr1_cs0_addr = 0x0,
ddr1_cs1_addr = 0x80000000,
ddr0_cs0_remapped_addr = 0x0000000100000000,
ddr0_cs1_remapped_addr = 0x80000000,
ddr1_cs0_remapped_addr = 0x0000000100000000,
ddr1_cs1_remapped_addr = 0x80000000,
highest_bank_bit = 0x0)
```

- Change DDR init frequencies, refer to Section 4.1.5
- Raise a DDR case to QTI

3.6.2 Stuck at BIMC Frequencies and BLL configurations of RPM

Example:

```
RPM log
95.131244: rpm_message_received (master: "APSS") (message id: 6149)
95.131247: rpm_svs (mode: RPM_SVS_FAST) (reason: imminent processing)
(current_speed: 400000)
95.131254: rpm_svs (mode: RPM_SVS_FAST) (reason: imminent processing)
(current_speed: 400000)
95.131262: rpm_process_request (master: "APSS") (resource type: clk2) (id:
0) (full name: bimc)
95.131264: rpm_xlate_request (resource type: clk2) (resource id: 0) (full
name: bimc)
95.131266: rpm_apply_request (resource type: clk2) (resource id: 0) (full
name: bimc) (bypassed: 0)
95.162566: rpm_err_fatal (lr: 0xfffffff9) (ipsr: 0x00000041) - RPM Wdog
Bark

Call stack
busywait(?)
HAL_clk_GenericPLLEnable(pSource = 0x00095894)
Clock_SourceOn(inline)
Clock_EnableSource(?, pSource = 0x00099250, ?)
Clock_SetClockConfig(pDomain = 0x00098D50, pNewConfig = 0x00091FF0)
Clock_SetClockSpeed(pClockResource = 0x00091094, ?)
Clock_BIMCSetSpeed(inline)
Clock_BIMCSwitchFrequency(?)
Clock_NPANodeBIMCFunc(pResource = 0x00097130, hClient = 0x000971B8, nState
= 105579)
npa_update_resource_state(inline)
npa_invoke_driver_function(inline)
npa_update_resource(inline)
```

```
npa_process_request(inline)
npa_issue_sync_request(?, ?)
rpm_npa_apply(info = 0x00096CA0)
resource_ee_apply(?, settling_time = 18446744069414584320)
do_resource_ee_request(?, ?, ?, ?, ?)
resource_ee_request(r = 0x00095418, ee = 0, set = RPM_ACTIVE_SET, data =
0x00099A10, ?)
Handler::processMessage(this = 0x00099860, header = 0x00099910,
allow_settling = TRUE)
Handler::execute_until(this = 0x00099860, preempt = FALSE, stop_time =
1831438004)
time_service_now(inline)
Task::execute(?, ?, stop_time = 1831438004)
Sched::run(this = 0x00093818)
time_service_now(inline)
main()
end of frame
```

3.6.2.1 Debug suggestion

- If this is always on specific devices, it should be a hardware or MSM chipset issue. It is suggested to do RMA or swap a new MSM
- If this is reproduced on multiple devices easily, raise a case to check with QTI whether there are known CRs for it. See section 1.4
  - Keep some PLL on

Bump up voltage for specific frequencies.

4 DDR debug issues

It is not possible to confirm a DDR issue just by one or two dumps. It depends on the SI/PI report from the vendor. However, there are some debugging changes to verify whether it is helpful for DDR issues when random crashes are encountered and if the error patterns match with those mentioned in Chapter 3.

It is not possible to cover all platforms as the code changes are specific to platforms.

4.1 DDR frequency

4.1.1 Check current DDR frequency

4.1.1.1 QCAP or crashscope report of DUMP

- Find "BIMC" in the path `RPM->NPA->resource`
- The active state is the current DDR frequency; the unit is KHz

Example:

```
: npa_resource (name: "/clk/bimc") (handle: 0x97250) (units: KHz) (resource
max: 465600) (active max: 465600) (active state: 465600) (active headroom:
172388) (request state: 637988)
 :     npa_client (name: MPSS) (handle: 0x9CD50) (resource: 0x97250) (type:
NPA_CLIENT_LIMIT_MAX) (request: 0)
 :     npa_client (name: MPSS) (handle: 0x9CD10) (resource: 0x97250) (type:
NPA_CLIENT_REQUIRED) (request: 0)
 :     npa_client (name: WCSS) (handle: 0x9C2E8) (resource: 0x97250) (type:
NPA_CLIENT_LIMIT_MAX) (request: -1)
 :     npa_client (name: WCSS) (handle: 0x9C2A8) (resource: 0x97250) (type:
NPA_CLIENT_REQUIRED) (request: 41944)
 :     npa_client (name: LPASS) (handle: 0x9B868) (resource: 0x97250)
(type: NPA_CLIENT_LIMIT_MAX) (request: -1)
 :     npa_client (name: LPASS) (handle: 0x9B828) (resource: 0x97250)
(type: NPA_CLIENT_REQUIRED) (request: 27500): npa_client (name: APSS)
(handle: 0x972D8) (resource: 0x97250) (type: NPA_CLIENT_REQUIRED) (request:
637988
```

4.1.1.2 Run-time Linux console

Use `cat /sys/kernel/debug/clk/bimc_clk/measure` to check BMIC frequency.

NOTE: For the MSM8909, MSM8916, MSM8917, MSM8937, MSM8940, and MSM8953 chipsets, the DDR frequency = BIMC frequency

For the MSM8939, MSM8952, and MSM8976 chispets, the DDR frequency = 2*BIMC frequency

For the MSM8992, MSM8994, and MSM8996 chipsets, lower legacy GCC frequency = BIMC frequency

Higher DDRCC frequency = 2*BIMC frequency

4.1.2 Check available DDR frequencies

4.1.2.1 MSM8909, MSM8916, MSM8917, MSM8939, MSM8937, MSM8940, MSM8952, MSM8976, and MSM8953

- File:

```
/rpm_proc/core/systemdrivers/clock/config/MxMxxxx/ClockBSP.c
```

- Variable:

```
ClockMuxConfigType BIMCClockConfig[] =
{
 {  9600000, { HAL_CLK_SOURCE_XO,    4, 1, 0, 0 }, CLOCK_VREG_LEVEL_LOW },
 { 52800000, { HAL_CLK_SOURCE_RAW1, 24, 1, 0, 0 }, CLOCK_VREG_LEVEL_LOW,
0, CHIPINFO_FAMILY_UNKNOWN, &BIMCPLLConfig[0]},
 { 105600000, { HAL_CLK_SOURCE_RAW1, 12, 1, 0, 0 }, CLOCK_VREG_LEVEL_LOW,
0, CHIPINFO_FAMILY_UNKNOWN, &BIMCPLLConfig[0]},
 { 139200000, { HAL_CLK_SOURCE_RAW1,  4, 1, 0, 0 }, CLOCK_VREG_LEVEL_LOW,
0, CHIPINFO_FAMILY_UNKNOWN, &BIMCPLLConfig[1]},
 { 192000000, { HAL_CLK_SOURCE_RAW1,  4, 1, 0, 0 }, CLOCK_VREG_LEVEL_LOW,
0, CHIPINFO_FAMILY_UNKNOWN, &BIMCPLLConfig[2]},
 { 268800000, { HAL_CLK_SOURCE_RAW1,  4, 1, 0, 0 },
CLOCK_VREG_LEVEL_LOW_PLUS,   0, CHIPINFO_FAMILY_UNKNOWN,
&BIMCPLLConfig[3]},
 { 350400000, { HAL_CLK_SOURCE_RAW1,  4, 0, 16, 0 },
CLOCK_VREG_LEVEL_LOW_PLUS,   0, CHIPINFO_FAMILY_UNKNOWN,
&BIMCPLLConfig[4]},
 { 374400000, { HAL_CLK_SOURCE_RAW1,  4, 0, 16, 0 },
CLOCK_VREG_LEVEL_NOMINAL,   0, CHIPINFO_FAMILY_UNKNOWN, &BIMCPLLConfig[5]},
 { 403200000, { HAL_CLK_SOURCE_RAW1,  4, 0, 18, 0 },
CLOCK_VREG_LEVEL_NOMINAL_PLUS, 0, CHIPINFO_FAMILY_UNKNOWN,
&BIMCPLLConfig[6]},
 { 460800000, { HAL_CLK_SOURCE_GPLL5, 4, 0, 20, 0 }, CLOCK_VREG_LEVEL_HIGH,
0, CHIPINFO_FAMILY_UNKNOWN, &BIMCPLLConfig[7]},
 { 0 }
};
```

NOTE: For the MSM8909, MSM8916, MSM8917, MSM8937, MSM8940, and MSM8953 chipsets, the DDR frequency = BIMC frequency

For the MSM8939, MSM8952, and MSM8976 chispets, the DDR frequency = 2*BIMC frequency

The MSM8940 and MSM8937 chipsets have the same code.

4.1.2.2 MSM8992 and MSM8994

- File:

```
boot_images/core/systemdrivers/clock/hw/msm8992/src/ClockSBLBIMC.c
```

- Variable:

```
static RPMClockMuxConfigType DDRClockConfig[] =
{
 /*
===========================================================================
===========================================================================
===============
 ** {freq_Hz,   {source,    DDRCCPLL_idx:DDR_div2x
DDRCC_mode:GPLL23_idx:APSS_div2x,   GFX_div2x:MPSS_div2x,
VDDA_EBI_vote},      VDDCX_vote }
 **
===========================================================================
===========================================================================
==============*/
 { 19200000, { HAL_CLK_SOURCE_XO,  HAL_CLK_BIMC_DIV2X_VAL(0, 2),
HAL_CLK_BIMC_M_VAL(0, 0, 4), HAL_CLK_BIMC_N_VAL( 8, 4),
CLOCK_VREG_LEVEL_LOW   }, CLOCK_VREG_LEVEL_LOW_MINUS },
 { 100000000, { HAL_CLK_SOURCE_GPLL0, HAL_CLK_BIMC_DIV2X_VAL(0, 12),
HAL_CLK_BIMC_M_VAL(0, 0, 24), HAL_CLK_BIMC_N_VAL(32, 24),
CLOCK_VREG_LEVEL_LOW   }, CLOCK_VREG_LEVEL_LOW_MINUS },
 { 150000000, { HAL_CLK_SOURCE_GPLL0, HAL_CLK_BIMC_DIV2X_VAL(0, 8),
HAL_CLK_BIMC_M_VAL(0, 0, 16), HAL_CLK_BIMC_N_VAL(32, 16),
CLOCK_VREG_LEVEL_LOW   }, CLOCK_VREG_LEVEL_LOW_MINUS },
 { 200000000, { HAL_CLK_SOURCE_GPLL0, HAL_CLK_BIMC_DIV2X_VAL(0, 6),
HAL_CLK_BIMC_M_VAL(0, 0, 12), HAL_CLK_BIMC_N_VAL(24, 12),
CLOCK_VREG_LEVEL_LOW   }, CLOCK_VREG_LEVEL_LOW    },
 { 300000000, { HAL_CLK_SOURCE_GPLL0, HAL_CLK_BIMC_DIV2X_VAL(0, 4),
HAL_CLK_BIMC_M_VAL(0, 0, 8), HAL_CLK_BIMC_N_VAL(16, 8),
CLOCK_VREG_LEVEL_LOW   }, CLOCK_VREG_LEVEL_LOW    },
 { 460800000, { HAL_CLK_SOURCE_GPLL2, HAL_CLK_BIMC_DIV2X_VAL(0, 2),
HAL_CLK_BIMC_M_VAL(0, 0, 4), HAL_CLK_BIMC_N_VAL( 8, 4),
CLOCK_VREG_LEVEL_LOW   }, CLOCK_VREG_LEVEL_NOMINAL  },
 { 547200000, { HAL_CLK_SOURCE_GPLL2, HAL_CLK_BIMC_DIV2X_VAL(0, 2),
HAL_CLK_BIMC_M_VAL(0, 1, 4), HAL_CLK_BIMC_N_VAL( 8, 4),
CLOCK_VREG_LEVEL_NOMINAL }, CLOCK_VREG_LEVEL_NOMINAL  },
 // DDRCC mode levels
 { 691200000, { HAL_CLK_SOURCE_GPLL2, HAL_CLK_BIMC_DIV2X_VAL(0, 32),
HAL_CLK_BIMC_M_VAL(1, 2, 2), HAL_CLK_BIMC_N_VAL( 4, 2),
CLOCK_VREG_LEVEL_NOMINAL }, CLOCK_VREG_LEVEL_NOMINAL  },
 { 777600000, { HAL_CLK_SOURCE_GPLL2, HAL_CLK_BIMC_DIV2X_VAL(1, 32),
HAL_CLK_BIMC_M_VAL(1, 3, 2), HAL_CLK_BIMC_N_VAL( 4, 2),
CLOCK_VREG_LEVEL_NOMINAL }, CLOCK_VREG_LEVEL_HIGH   },
 { 931200000, { HAL_CLK_SOURCE_GPLL2, HAL_CLK_BIMC_DIV2X_VAL(2, 32),
HAL_CLK_BIMC_M_VAL(1, 4, 2), HAL_CLK_BIMC_N_VAL( 4, 2),
CLOCK_VREG_LEVEL_HIGH  }, CLOCK_VREG_LEVEL_HIGH   },
 { 0 }
};
```

4.1.2.3 MSM8996 and MSM8998

- File:

```
Boot_images/QcomPkg/Msm899xPkg/Library/ClockTargetLib/ClockBIMC.c
```

- Function:

```
RPMClockMuxConfigType DDRClockConfig_V2[] =
{
 /*
============================================================================
============================================================================
===============
 ** {freq_Hz,   {source,    DDRCCPLL_idx:DDR_div2x
DDRCC_mode:GPLL23_idx:APSS_div2x,   GFX_div2x:MPSS_div2x,
VDDA_EBI_vote},      VDDCX_vote }
 **
============================================================================
============================================================================
==============*/
 { 100000000, { HAL_CLK_SOURCE_GPLL0_DIV2, HAL_CLK_BIMC_DIV2X_VAL(0, 0, 6),
HAL_CLK_BIMC_M_VAL(0, 16, 12), HAL_CLK_BIMC_N_VAL(24, 12),
CLOCK_VREG_LEVEL_LOW_MINUS }, CLOCK_VREG_LEVEL_LOW_MINUS },
 { 150000000, { HAL_CLK_SOURCE_GPLL0_DIV2, HAL_CLK_BIMC_DIV2X_VAL(0, 1, 4),
HAL_CLK_BIMC_M_VAL(0, 12, 8), HAL_CLK_BIMC_N_VAL(16, 8),
CLOCK_VREG_LEVEL_LOW_MINUS }, CLOCK_VREG_LEVEL_LOW_MINUS },
 { 200000000, { HAL_CLK_SOURCE_GPLL2,   HAL_CLK_BIMC_DIV2X_VAL(0, 2, 2),
HAL_CLK_BIMC_M_VAL(0, 6, 4), HAL_CLK_BIMC_N_VAL( 8, 4),
CLOCK_VREG_LEVEL_LOW_MINUS }, CLOCK_VREG_LEVEL_LOW_MINUS },
 { 300000000, { HAL_CLK_SOURCE_GPLL0,   HAL_CLK_BIMC_DIV2X_VAL(0, 3, 4),
HAL_CLK_BIMC_M_VAL(0, 12, 8), HAL_CLK_BIMC_N_VAL(16, 8),
CLOCK_VREG_LEVEL_LOW_MINUS }, CLOCK_VREG_LEVEL_LOW    },

 // DDRCC mode levels, new
 { 412800000, { HAL_CLK_SOURCE_GPLL2,   HAL_CLK_BIMC_DIV2X_VAL(1, 4, 16),
HAL_CLK_BIMC_M_VAL(3, 8, 6), HAL_CLK_BIMC_N_VAL(12, 12),
CLOCK_VREG_LEVEL_LOW    }, CLOCK_VREG_LEVEL_LOW    },
 { 547200000, { HAL_CLK_SOURCE_GPLL2,   HAL_CLK_BIMC_DIV2X_VAL(1, 5, 18),
HAL_CLK_BIMC_M_VAL(3, 8, 6), HAL_CLK_BIMC_N_VAL(12, 12),
CLOCK_VREG_LEVEL_LOW    }, CLOCK_VREG_LEVEL_LOW      },
 { 681600000, { HAL_CLK_SOURCE_GPLL2,   HAL_CLK_BIMC_DIV2X_VAL(1, 6, 20),
HAL_CLK_BIMC_M_VAL(3, 8, 6), HAL_CLK_BIMC_N_VAL(12, 12),
CLOCK_VREG_LEVEL_LOW    }, CLOCK_VREG_LEVEL_LOW    },
 { 768000000, { HAL_CLK_SOURCE_GPLL2,   HAL_CLK_BIMC_DIV2X_VAL(1, 7, 22),
HAL_CLK_BIMC_M_VAL(3, 8, 6), HAL_CLK_BIMC_N_VAL(12, 8),
CLOCK_VREG_LEVEL_NOMINAL  }, CLOCK_VREG_LEVEL_NOMINAL  },
 { 1017600000, { HAL_CLK_SOURCE_GPLL2,   HAL_CLK_BIMC_DIV2X_VAL(1, 8, 24),
HAL_CLK_BIMC_M_VAL(3, 4, 3), HAL_CLK_BIMC_N_VAL( 6, 6),
CLOCK_VREG_LEVEL_NOMINAL  }, CLOCK_VREG_LEVEL_NOMINAL  },
 { 1296000000, { HAL_CLK_SOURCE_GPLL2,   HAL_CLK_BIMC_DIV2X_VAL(1, 9, 26),
HAL_CLK_BIMC_M_VAL(3, 4, 3), HAL_CLK_BIMC_N_VAL( 6, 6),
CLOCK_VREG_LEVEL_NOMINAL  }, CLOCK_VREG_LEVEL_NOMINAL  },
 { 1555200000, { HAL_CLK_SOURCE_GPLL2,   HAL_CLK_BIMC_DIV2X_VAL(1, 10, 28),
HAL_CLK_BIMC_M_VAL(3, 4, 3), HAL_CLK_BIMC_N_VAL( 6, 6),
CLOCK_VREG_LEVEL_NOMINAL  }, CLOCK_VREG_LEVEL_NOMINAL  },
 { 1804800000, { HAL_CLK_SOURCE_GPLL2,   HAL_CLK_BIMC_DIV2X_VAL(1, 11, 30),
HAL_CLK_BIMC_M_VAL(4, 5, 4), HAL_CLK_BIMC_N_VAL( 7, 7),
CLOCK_VREG_LEVEL_HIGH   }, CLOCK_VREG_LEVEL_HIGH   },
 { 0 }
};
```

4.1.2.4 SDM630 and SDM660

- File:

`Boot_images/QcomPkg/SDM660Pkg/Library/ClockTargetLib/ClockBIMC.c`

- Function:

```
RPMClockMuxConfigType DDRClockConfig_V1[] =
{
  /*
============================================================================
============================================================================
================================
  ** {freq_Hz,  {source,                 (ddrcc_mode, ddr_gpll0_sel,
ddr_div2x, cdsp_gpll0_sel, cdsp_div2x),

( bimc_gpll0_sel, bimc_div2x, hmss_gpll0_sel, hmss_div2x)

(gfx_gpll0_sel, gfx_div2x, mpss_gpll0_sel, mpss_div2x),

{gpll23_idx, VDDA_EBI_vote}},               VDDCX_vote }
  **
============================================================================
============================================================================
===============================*/
```

```
  {  100000000, {HAL_CLK_SOURCE_GPLL0_DIV2,  HAL_CLK_BIMC_DIV2X_VAL(0, 0, 
6, 0, 5), HAL_CLK_BIMC_M_VAL(0, 3, 0, 3), HAL_CLK_BIMC_N_VAL(0, 5, 0, 3), 
HAL_CLK_BIMC_N2D_VAL(0, RAILWAY_SVS_LOW)}, RAILWAY_SVS_LOW},
  {  150000000, {HAL_CLK_SOURCE_GPLL0_DIV2,  HAL_CLK_BIMC_DIV2X_VAL(0, 0, 
4, 0, 5), HAL_CLK_BIMC_M_VAL(0, 3, 0, 3), HAL_CLK_BIMC_N_VAL(0, 5, 0, 3), 
HAL_CLK_BIMC_N2D_VAL(0, RAILWAY_SVS_LOW)}, RAILWAY_SVS_LOW},
  {  200000000, {HAL_CLK_SOURCE_GPLL0_DIV2,  HAL_CLK_BIMC_DIV2X_VAL(0, 0, 
3, 0, 5), HAL_CLK_BIMC_M_VAL(0, 3, 0, 3), HAL_CLK_BIMC_N_VAL(0, 5, 0, 3), 
HAL_CLK_BIMC_N2D_VAL(0, RAILWAY_SVS_LOW)}, RAILWAY_SVS_LOW},
  {  300000000, {HAL_CLK_SOURCE_GPLL0,       HAL_CLK_BIMC_DIV2X_VAL(0, 1, 
4, 1, 6), HAL_CLK_BIMC_M_VAL(1, 3, 2, 4), HAL_CLK_BIMC_N_VAL(1, 6, 2, 4), 
HAL_CLK_BIMC_N2D_VAL(0, RAILWAY_SVS)}, RAILWAY_SVS},
  {  412800000, {HAL_CLK_SOURCE_GPLL2,       HAL_CLK_BIMC_DIV2X_VAL(0, 0, 
4, 1, 6), HAL_CLK_BIMC_M_VAL(1, 3, 2, 4), HAL_CLK_BIMC_N_VAL(1, 6, 2, 4), 
HAL_CLK_BIMC_N2D_VAL(0, RAILWAY_SVS)}, RAILWAY_SVS},

  {  547200000, {HAL_CLK_SOURCE_GPLL0,       HAL_CLK_BIMC_DIV2X_VAL(1, 0, 
16, 1, 6), HAL_CLK_BIMC_M_VAL(1, 3, 2, 4), HAL_CLK_BIMC_N_VAL(1, 6, 2, 4), 
HAL_CLK_BIMC_N2D_VAL(0, RAILWAY_SVS)}, RAILWAY_SVS},
  {  681600000, {HAL_CLK_SOURCE_GPLL0,       HAL_CLK_BIMC_DIV2X_VAL(1, 0, 
18, 1, 6), HAL_CLK_BIMC_M_VAL(1, 3, 2, 4), HAL_CLK_BIMC_N_VAL(1, 6, 2, 4), 
HAL_CLK_BIMC_N2D_VAL(0, RAILWAY_SVS)}, RAILWAY_SVS},
  {  768000000, {HAL_CLK_SOURCE_GPLL0,       HAL_CLK_BIMC_DIV2X_VAL(1, 0, 
20, 1, 6), HAL_CLK_BIMC_M_VAL(1, 3, 2, 4), HAL_CLK_BIMC_N_VAL(1, 6, 2, 4), 
HAL_CLK_BIMC_N2D_VAL(0, RAILWAY_SVS)}, RAILWAY_SVS},
  { 1017600000, {HAL_CLK_SOURCE_GPLL2,       HAL_CLK_BIMC_DIV2X_VAL(1, 0, 
22, 0, 4), HAL_CLK_BIMC_M_VAL(0, 2, 1, 2), HAL_CLK_BIMC_N_VAL(0, 4, 0, 3), 
HAL_CLK_BIMC_N2D_VAL(1, RAILWAY_SVS_HIGH)}, RAILWAY_SVS_HIGH},
  { 1296000000, {HAL_CLK_SOURCE_GPLL2,       HAL_CLK_BIMC_DIV2X_VAL(1, 0, 
24, 2, 4), HAL_CLK_BIMC_M_VAL(2, 2, 0, 4), HAL_CLK_BIMC_N_VAL(2, 4, 1, 2), 
HAL_CLK_BIMC_N2D_VAL(2, RAILWAY_NOMINAL)}, RAILWAY_NOMINAL},
  { 1353600000, {HAL_CLK_SOURCE_GPLL2,       HAL_CLK_BIMC_DIV2X_VAL(1, 0, 
26, 2, 4), HAL_CLK_BIMC_M_VAL(2, 2, 0, 4), HAL_CLK_BIMC_N_VAL(2, 4, 1, 2), 
HAL_CLK_BIMC_N2D_VAL(2, RAILWAY_NOMINAL)}, RAILWAY_NOMINAL},
  { 1555200000, {HAL_CLK_SOURCE_GPLL2,       HAL_CLK_BIMC_DIV2X_VAL(1, 0, 
28, 2, 4), HAL_CLK_BIMC_M_VAL(2, 2, 0, 4), HAL_CLK_BIMC_N_VAL(2, 4, 1, 2), 
HAL_CLK_BIMC_N2D_VAL(2, RAILWAY_NOMINAL)}, RAILWAY_NOMINAL},
  { 1804800000, {HAL_CLK_SOURCE_GPLL2,       HAL_CLK_BIMC_DIV2X_VAL(1, 0, 
30, 0, 8), HAL_CLK_BIMC_M_VAL(0, 4, 4, 4), HAL_CLK_BIMC_N_VAL(0, 8, 1, 2), 
HAL_CLK_BIMC_N2D_VAL(3, RAILWAY_TURBO)}, RAILWAY_TURBO},
  {0},
};
```

4.1.2.5 SDM630

- File:

`Boot_images/QcomPkg/SDM660Pkg/Library/ClockTargetLib/ClockBIMC.c`

- Function:

```
/* Starhawks DDR clock plan */
```

```
RPMClockMuxConfigType DDRClockConfig_SDM630[] =
{
  /*
==============================================================================
==============================================================================
================================
  ** {freq_Hz,  {source,                 (ddrcc_mode, ddr_gpll0_sel,
ddr_div2x, cdsp_gpll0_sel, cdsp_div2x),

( bimc_gpll0_sel, bimc_div2x, hmss_gpll0_sel, hmss_div2x)

(gfx_gpll0_sel, gfx_div2x, mpss_gpll0_sel, mpss_div2x),

{gpll23_idx, VDDA_EBI_vote}},            VDDCX_vote }
  **
==============================================================================
==============================================================================
===============================*/
  {  100000000, {HAL_CLK_SOURCE_GPLL0,       HAL_CLK_BIMC_DIV2X_VAL(0,
0,12, 0,10), HAL_CLK_BIMC_M_VAL(0, 6, 0, 6), HAL_CLK_BIMC_N_VAL(0,10, 0,
6), HAL_CLK_BIMC_N2D_VAL(0, RAILWAY_SVS_LOW)}, RAILWAY_SVS_LOW},
  {  150000000, {HAL_CLK_SOURCE_GPLL0,       HAL_CLK_BIMC_DIV2X_VAL(0, 0,
8, 0,10), HAL_CLK_BIMC_M_VAL(0, 6, 0, 6), HAL_CLK_BIMC_N_VAL(0,10, 0, 6),
HAL_CLK_BIMC_N2D_VAL(0, RAILWAY_SVS_LOW)}, RAILWAY_SVS_LOW},
  {  200000000, {HAL_CLK_SOURCE_GPLL0,       HAL_CLK_BIMC_DIV2X_VAL(0, 0,
6, 0,10), HAL_CLK_BIMC_M_VAL(0, 6, 0, 6), HAL_CLK_BIMC_N_VAL(0,10, 0, 6),
HAL_CLK_BIMC_N2D_VAL(0, RAILWAY_SVS_LOW)}, RAILWAY_SVS_LOW},
  {  300000000, {HAL_CLK_SOURCE_GPLL0,       HAL_CLK_BIMC_DIV2X_VAL(0, 1,
4, 1, 6), HAL_CLK_BIMC_M_VAL(2, 4, 2, 4), HAL_CLK_BIMC_N_VAL(1, 6, 2, 4),
HAL_CLK_BIMC_N2D_VAL(0, RAILWAY_SVS)}, RAILWAY_SVS},
  {  412800000, {HAL_CLK_SOURCE_GPLL2,       HAL_CLK_BIMC_DIV2X_VAL(0, 0,
4, 1, 6), HAL_CLK_BIMC_M_VAL(2, 4, 2, 4), HAL_CLK_BIMC_N_VAL(1, 6, 2, 4),
HAL_CLK_BIMC_N2D_VAL(0, RAILWAY_SVS)}, RAILWAY_SVS},


  {  547200000, {HAL_CLK_SOURCE_GPLL0,       HAL_CLK_BIMC_DIV2X_VAL(1, 0,
16, 1, 6), HAL_CLK_BIMC_M_VAL(2, 4, 2, 4), HAL_CLK_BIMC_N_VAL(1, 6, 2, 4),
HAL_CLK_BIMC_N2D_VAL(0, RAILWAY_SVS)}, RAILWAY_SVS},
  {  681600000, {HAL_CLK_SOURCE_GPLL0,       HAL_CLK_BIMC_DIV2X_VAL(1, 0,
18, 1, 6), HAL_CLK_BIMC_M_VAL(2, 4, 2, 4), HAL_CLK_BIMC_N_VAL(1, 6, 2, 4),
HAL_CLK_BIMC_N2D_VAL(0, RAILWAY_SVS)}, RAILWAY_SVS},
  {  768000000, {HAL_CLK_SOURCE_GPLL0,       HAL_CLK_BIMC_DIV2X_VAL(1, 0,
20, 1, 6), HAL_CLK_BIMC_M_VAL(2, 4, 2, 4), HAL_CLK_BIMC_N_VAL(1, 6, 2, 4),
HAL_CLK_BIMC_N2D_VAL(0, RAILWAY_SVS)}, RAILWAY_SVS},
  { 1017600000, {HAL_CLK_SOURCE_GPLL2,       HAL_CLK_BIMC_DIV2X_VAL(1, 0,
22, 0, 4), HAL_CLK_BIMC_M_VAL(0, 2, 1, 2), HAL_CLK_BIMC_N_VAL(0, 4, 0, 3),
HAL_CLK_BIMC_N2D_VAL(1, RAILWAY_SVS_HIGH)}, RAILWAY_SVS_HIGH},
  { 1296000000, {HAL_CLK_SOURCE_GPLL2,       HAL_CLK_BIMC_DIV2X_VAL(1, 0,
24, 2, 4), HAL_CLK_BIMC_M_VAL(2, 2, 0, 4), HAL_CLK_BIMC_N_VAL(3, 6, 1, 2),
HAL_CLK_BIMC_N2D_VAL(2, RAILWAY_NOMINAL)}, RAILWAY_NOMINAL},
    {0},
  };
```

4.1.3 Limit DDR to a fixed frequency

4.1.3.1 Fix DDR frequency in RPM

This is specific to all platforms that have the RPM subsystems like the MSM8x10, MSM8x26, MSM8x74, MSM8x16, MSM8x39, MSM8x37, MSM8x17, MSM8x52, MSM8x76, MSM8x94, MSM8x96, MSM9x15, MSM9x25, MSM8x40, MSM8953, MSM9x35, MSM8994, MSM8996, MSM8998, SDM630, and SDM660 chipsets.

The modification is in work while RPM is running

```
In rpm_proc
static npa_resource_state Clock_NPANodeBIMCFunc
{
…
 /* Do not scale clock if DCVS is disable. Return current speed */
 pDrvCtxt = Clock_GetDrvCtxt();
 if ( !pDrvCtxt->bDCVSEnabled )
 {
  return pClockRsc->pClock->pDomain->pBSPConfig[pClockRsc-
>nCurLevel].nFreqHz / 1000;
 }
 +if ( nState != 0 )
 +{
  +nState = 460000; //460000 is just an example, you can change to other
values but make sure this vlalue is in the BIMCClockConfig of ClockBSP.c,
the unit is khz

 +}
```

4.1.3.2 Fix DDR frequency in SBL1 or XBL

The modification is work in the two situations to avoid some problem caused by DDR high frequency.

1. Before RPM boots up
2. After DDR training is finished and before the system boots up
3. Before DDR training

First two scenarios can use the following changes and scenario 3 can use the changes mentioned in section 4.1.5.1

Example: MSM8953

```
[SBL1]
boot_images\core\boot\ddr\hw\msm8953\ddr_target.c
boolean ddr_do_phy_training( void )
{
```

```
<snip>
HAL_DDR_Boot_Training(ddrsns_share_data, DDR_CH_BOTH, DDR_CS_BOTH,
DDR_TRAINING_MODE_INIT);

- ddr_pre_clock_switch(0, ddrsns_share_data-
>misc.clock_plan[ddrsns_share_data->misc.ddr_num_clock_levels -
1].clk_freq_in_khz, SDRAM_INTERFACE_0);
- boot_clock_set_bimcspeed(ddrsns_share_data-
>misc.clock_plan[ddrsns_share_data->misc.ddr_num_clock_levels -
1].clk_freq_in_khz);
- ddr_post_clock_switch(0, ddrsns_share_data-
>misc.clock_plan[ddrsns_share_data->misc.ddr_num_clock_levels -
1].clk_freq_in_khz, SDRAM_INTERFACE_0);
+ ddr_pre_clock_switch(0, 844800, SDRAM_INTERFACE_0);
+ boot_clock_set_bimcspeed(844800);
+ ddr_post_clock_switch(0, 844800, SDRAM_INTERFACE_0);
ddr_printf (DDR_NORMAL, "DDR: End of HAL DDR Boot Training");
}
#if ONE_TIME_TRAINING
else {
- ddr_pre_clock_switch(0, ddrsns_share_data-
>misc.clock_plan[ddrsns_share_data->misc.ddr_num_clock_levels -
1].clk_freq_in_khz, SDRAM_INTERFACE_0);
- boot_clock_set_bimcspeed(ddrsns_share_data-
>misc.clock_plan[ddrsns_share_data->misc.ddr_num_clock_levels -
1].clk_freq_in_khz);
- ddr_post_clock_switch(0, ddrsns_share_data-
>misc.clock_plan[ddrsns_share_data->misc.ddr_num_clock_levels -
1].clk_freq_in_khz, SDRAM_INTERFACE_0);
+ ddr_pre_clock_switch(0, 844800, SDRAM_INTERFACE_0);
+ boot_clock_set_bimcspeed(844800);
+ ddr_post_clock_switch(0, 844800, SDRAM_INTERFACE_0);
return FALSE;
}
#endif
serial_number = BOOT_HWIO_IN(QFPROM_RAW_SERIAL_NUM, 0);
<snip>
}
```

4.1.3.3 For legacy platform

ebi1, ebi2, and cpu clk are highly connected and cannot adjust one separately.

Take SDM660, for example:

```
.global_perf =
{
[CLKRGM_GLOBAL_PERF_LEVEL_0] = CLKRGM_GLOBAL_SPEED_XO,
[CLKRGM_GLOBAL_PERF_LEVEL_1] = CLKRGM_GLOBAL_SPEED_48_MHZ, // MCLK 48MHz,
HCLK 24MHz, EBI2 12MHz
[CLKRGM_GLOBAL_PERF_LEVEL_2] = CLKRGM_GLOBAL_SPEED_96_MHZ, // MCLK 96MHz,
HCLK 48MHz, EBI2 24MHz
[CLKRGM_GLOBAL_PERF_LEVEL_3] = CLKRGM_GLOBAL_SPEED_144_MHZ, // MCLK 144MHz,
HCLK 72MHz, EBI2 36MHz
[CLKRGM_GLOBAL_PERF_LEVEL_4] = CLKRGM_GLOBAL_SPEED_288_MHZ, // MCLK 288MHz,
HCLK 96MHz, EBI2 48MHz
[CLKRGM_GLOBAL_PERF_LEVEL_5] = CLKRGM_GLOBAL_SPEED_384_MHZ, // MCLK 384MHz,
HCLK 128MHz, EBI2 64MHz
[CLKRGM_GLOBAL_PERF_LEVEL_6] = CLKRGM_GLOBAL_SPEED_480_MHZ, // MCLK 480MHz,
HCLK 160MHz, EBI2 80MHz

},
```

For different performance levels, there are different clk plans - ebi2 is for Flash.

To fix ebi2 to a frequency (only for test purposes), set min/max to the same level.

```
 .global_perf_cfg =
  {
   .min = CLKRGM_GLOBAL_PERF_LEVEL_6,
   .max = CLKRGM_GLOBAL_PERF_LEVEL_6, //fixed at highest level
```

4.1.4 Adjust specific DDR frequency point

This is used to avoid RF issues. Adjust the DDR frequency configurationss by L, M, and N values, once the issue is found by the fixed DDR frequency (see 4.1.3 for more information).

NOTE: Confirm with QTI for formal usage.

4.1.4.1 MSM8926

Most typical platform for desense issue.

For backup frequency plans details, refer to *MSM8X26/MSM8X28+WTR2605 DDR Desense Issue* (80-NC832-17) and *MSM8926/MSM8928 Chipset DDR Desense Issue Application Note* (80-NE925-17).

4.1.4.2 MSM8916 and MSM8909

- Example: Replace 533 MHz with 518.4 MHz
- Common rules: BIMCPLL init is done in SBL1. RPM changes DDR frequency according to the frequency table, BIMCClockConfig[].
- To set your own frequency:

1. Modify SBL1 first.

   The calculation is 19.2*(L+(M/N))/2

2. Modify BIMCClockConfig[] for your frequency entry in ClockBSP.c in RPM.

```
boot_images/core/systemdrivers/clock/hw/msm8916/src/ClockSBLConfig.c

/* BIMCPLL @ 1036.8MHz. */
.BIMCPLL_Cfg =
{
```

```
.nPLLModeAddr = HWIO_ADDR(GCC_BIMC_PLL_MODE),
.nVoteAddr = HWIO_ADDR(GCC_RPM_GPLL_ENA_VOTE),
.nVoteMask = HWIO_FMSK(GCC_RPM_GPLL_ENA_VOTE, BIMC_PLL),
.nVCO = 0,
.nPreDiv = 1,
.nPostDiv = 1,
.nL = 54, // change here
.nM = 0, // change here
.nN = 1, // change here
.nConfigCtl = 0x00031000 },
rpm_proc/core/systemdrivers/clock/config/msm8916/ClockBSP.c
/*----------------------------------------------------------------------*/
/* GPLL3 is retained here to avoid adding a new enum for BIMC_PLL */ /*----
----------------------------------------------------------------*/
{
/* .eSource = */ HAL_CLK_SOURCE_GPLL3,
/* .HALConfig */ {
/* .HALConfig.eSource = */ HAL_CLK_SOURCE_XO,
/* .HALConfig.eVCO = */ HAL_CLK_PLL_VCO1,
/* .HALConfig.nPreDiv = */ 1,
/* .HALConfig.nPostDiv = */ 1,
/* .HALConfig.nL = */ 54, // change here
/* .HALConfig.nM = */ 0, // change here
/* .HALConfig.nN = */ 1, // change here },
/* .nConfigMask = */ CLOCK_CONFIG_PLL_FSM_MODE_ENABLE,
/* .nFreqHz = */ 1036800 * 1000, // // change here
/* .eVRegLevel = */ CLOCK_VREG_LEVEL_LOW, },
const ClockMuxConfigType BIMCClockConfig[] = { { 9600000,
{ HAL_CLK_SOURCE_XO, 1, 1, 1, 1 }, CLOCK_VREG_LEVEL_LOW }, { 50000000,
{ HAL_CLK_SOURCE_GPLL0, 16, 1, 1, 1 }, CLOCK_VREG_LEVEL_LOW }, { 100000000,
{ HAL_CLK_SOURCE_GPLL0, 8, 1, 1, 1 }, CLOCK_VREG_LEVEL_LOW }, { 200000000,
{ HAL_CLK_SOURCE_GPLL0, 4, 1, 1, 1 }, CLOCK_VREG_LEVEL_LOW }, { 400000000,
{ HAL_CLK_SOURCE_GPLL0, 2, 1, 1, 1 }, CLOCK_VREG_LEVEL_NOMINAL },
- { 533000000, { HAL_CLK_SOURCE_GPLL3, 2, 1, 1, 1 },
CLOCK_VREG_LEVEL_HIGH },
+ { 518400000, { HAL_CLK_SOURCE_GPLL3, 2, 1, 1, 1 },
CLOCK_VREG_LEVEL_HIGH },
{ 0 }
```

4.1.5 Change DDR frequency at SBL or XBL

4.1.5.1 Change DDR initialization frequency

In the initializtion before DDR training, configure DDR frequency and PLL

Take MDM9x40 as an example, find the “ddr_speed_khz” in XBL/SBL codes.

```
-ddr_speed_khz = 404200;
+ddr_speed_khz = 202100;
boolean Clock_PreDDRInit( uint32 ddr_type)
{
  #ifndef FEATURE_RUMI_BOOT
  uint32 mask;
  /* A mux config for BIMC clock to 808.4 MHz */
  const ClockConfigMuxType  clkCfg =
  {
    .nCMDCGRAddr = HWIO_ADDR(GCC_BIMC_DDR_CMD_RCGR),
    .eMux = MUX_GCC,
    .eSource = SRC_GPLL2,
    .nDiv2x = 2, // default 2 is 404.2mhz, change to 4 for 202.1mhz, just
for a test
    .nM = 0,
    .nN = 0,
    .n2D = 0
  };
```

4.1.5.2 Change DDR frequency during XBL/SBL

This is only for test purpose; usually for debugging DDR training problems.

Use the function as follows in XBL/SBL. The unit is KHZ.

```
Clock_SetBIMCSpeed()
```

4.1.5.3 Limit max DDR frequency during training

This is only for test purpose; usually for debugging DDR training problems.

Take MSM8996, for example:

```
FILE : \boot_images\QcomPkg\Msm8996Pkg\Library\ClockTargetLib\ClockBimc.c
+void Clock_limit_MaxDDRFreq(uint32 nFreqKHz)
+{
+ uint32 nIdx;

+ Clock_SetBIMCSpeed(nFreqKHz);

+ for ( nIdx = 0; (DDRClockConfigTbl->table[nIdx].nFreqHz != 0); nIdx++ )
+ {
+ if(DDRClockConfigTbl->table[nIdx].nFreqHz > (nFreqKHz * 1000)) {
+ DDRClockConfigTbl->table[nIdx].nChipVersion = BSP_NOT_SUPPORTED;
+ }
+ }
+}
An example to limit Max DDR freqs just after training
File : \boot_images\QcomPkg\Msm8996Pkg\Library\XBLLoaderLib\sbl1_hw.c
+extern void Clock_limit_MaxDDRFreq(uint32 nFreqKHz);
void sbl1_ddr_init(bl_shared_data_type *bl_shar`ed_data)
<SNIP>
#ifdef FEATURE_DEVICEPROGRAMMER_IMAGE
/* Skip DDR training for Device Programmer, and lower clock speed */
boot_Clock_SetBIMCSpeed(200000);
#else
```

```
if(!boot_dload_is_dload_mode_set())
{
boot_enable_led(DDR_TRAINING_LED, TRUE);
if (boot_ddr_do_ddr_training())
{
/* Do DDR training */
boot_log_message("do_ddr_training, Start");
boot_log_start_timer();
ddr_training_entry();
ddr_save_training_required = boot_ddr_post_training();
boot_log_stop_timer("do_ddr_training, Delta");
}
boot_enable_led(DDR_TRAINING_LED, FALSE);
+ Clock_limit_MaxDDRFreq(1555200);
```

4.2 Drive strength

It is possible to tune the drive strength for both DDR side (reading) and MSM side (writing) to improve the stability of the DDR. The ways to change drive strength are specific to different platforms. In this document, only the most frequently used are covered.

NOTE: Sync-up with QTI and the corresponding vendor for more information.

There are following kinds of drive strength:

- DDR side– This is linked to DDR reading; refer to the DDR specification for the driver strength table.
- MSM side– This is linked to DDR writing; see section 4.2.1 for details.

4.2.1 Drive strength change

4.2.1.1 MDM9x15

- For EBI pad DS, check DIM_DQ_PAD_CFGx 4:6 bits ROUT. Refer to *MDM8215(M), MDM8615M, MDM8110M, MDM9X15(M) Software Interface* (80-N5423-2) for more details.

```
void ddr_target_init()
{
 if (HWIO_INF(QFPROM_RAW_FEAT_CONFIG_ROW5_MSB, STACKED_MEMORY_ID))
 {
  /* rout=5 for stacked DDR */
  ddr_boot_config.DIM_CA_PAD_CFG0 = 0x00222250;
  ddr_boot_config.DIM_CA_PAD_CFG1 = 0x00220450;
  ddr_boot_config.DIM_DQ_PAD_CFG0 = 0x00222250;
  ddr_boot_config.DIM_DQ_PAD_CFG1 = 0x00222250;
 }
 else
 {
  /* rout=7 for non-stacked DDR */
  ddr_boot_config.DIM_CA_PAD_CFG0 = 0x00222270;
  ddr_boot_config.DIM_CA_PAD_CFG1 = 0x00220470;
  ddr_boot_config.DIM_DQ_PAD_CFG0 = 0x00222270;
  ddr_boot_config.DIM_DQ_PAD_CFG1 = 0x00222270;
 }

} /* ddr_target_init */
```

- For DDR2 device DS, write the MR3 register of DDR in DDR init. For example:

```
 //MR3
//0000B: Reserved
//0001B: 34.3-Ω typical
//0010B: 40-Ω typical (default)
//0011B: 48-Ω typical
//0100B: 60-Ω typical
//0101B: Reserved for 68.6-Ω typical
//0110B: 80-Ω typical
//0111B: 120-Ω typical
In end of
HAL_SDRAM_LPDDR2_Init
{
…
HAL_SDRAM_Write_MR(interface, chip_select, 0x3, xxx);
}
3 For DDR1 device
HAL_SDRAM_LPDDR1_Update_MR_Settings()
{
 /*
   Extended Mode Register

   BIT[7:5] - Drive Strength
   BIT[4:3] - TCSR (optional)
   BIT[2:0] - PASR (optional)
  */
  HAL_SDRAM_Write_MR(SDRAM_INTERFACE_0, chip_select, 0x1, 0x0);
}
```

4.2.1.2 MSM8974

- From DDR device to MSM: MR3 is for input drive strength from the MSM perspective.
- MR3is programmed after MR1 and MR2 in order to change the reference output driver impedance in the DRAM device.
- MR3 is not programmed. To program MR3, add the following sequence after MR2 programming.

```
\boot_images\core\boot\ddr\hw\controller\BIMC_SHKE_v1.c // * Please check
Sconscript first for correct file. *
```

```
void HAL_SDRAM_SHKE_Device_Init(SDRAM_INTERFACE interface, SDRAM_CHIPSELECT
chip_select, uint32 clk_speed)
{
<snip>
/* Program MR2 */
HAL_SDRAM_Write_MR(interface, chip_select, 0x2, data);
/*start of MR3 setting*/
/* Program MR3 */
data = 0x1; // 34.3 Ω for example : This is an example. i.e. Drive Strength
of MR3 was set to 0x1. Please refer to datasheet for other values.
HAL_SDRAM_Write_MR(interface, chip_select, 0x3, data);
/*end of MR3 settings*/
<snip>
}
```

- From MSM to DDR device: ROUT is for output drive strength from the MSM perspective.

It is possible to decrease the drive strength by decreasing the bit (XX_ROUT) value; but resistance is not linear, that is, to set a higher output drive strength than the default value, set ROUT to a higher value than the default value of QTI.

NOTE: There is a positive correlation between ROUT and the output drive strength.

```
 boot_images\core\boot\ddr\hw\msm8974\ddr_config.c // * Please check
Sconscript first for correct file. *
uint32 ddr_caphy_config_static[][2] =
{u
<snip>
{BOOT_HWIO_ADDR(PHY_CA_ADDR(DDRPHY_CA_PAD_CFG0)), 0x200AF150},
//CA_ROUT[6:4] - default value might be different with you
{BOOT_HWIO_ADDR(PHY_CA_ADDR(DDRPHY_CA_PAD_CFG1)), 0x200A0FD0},
//CK_ROUT[6:4] - default value might be different with you
<snip>
}
<snip>
uint32 ddr_dqphy_config_static[][2] =
{
<snip>
{BOOT_HWIO_ADDR(PHY_DQ_ADDR(DDRPHY_DQ_PAD_CFG0)), 0x200AF150},
//DQ_ROUT[6:4] - default value might be different with you
{BOOT_HWIO_ADDR(PHY_DQ_ADDR(DDRPHY_DQ_PAD_CFG1)), 0x200A4FD0},
//DQS_ROUT[6:4] - default value might be different with you
<snip>
}
```

4.2.1.3 MSM8x10

- From DDR device to MSM: MR3 is for input drive strength from the MSM perspective.
- MR3 is programmed after MR1 and MR2 in order to change the reference output driver impedance in the DRAM device.
- MR3 is not programmed. To program MR3, add the following sequence after MR2 programming.

```
 \boot_images\core\boot\ddr\hw\controller\BIMC_SHKE_v1.c
void HAL_SDRAM_SHKE_Device_Init(SDRAM_INTERFACE interface, SDRAM_CHIPSELECT
chip_select, uint32 clk_speed)
{
<snip>
/* Program MR2 */
HAL_SDRAM_Write_MR(interface, chip_select, 0x2, data);

/*start of MR3 setting*/
/* Program MR3 */
data = 0x1; // 34.3 Ω for example : This is an example. i.e. Drive Strength
of MR3 was set to 0x1. Please refer to datasheet for other values.
HAL_SDRAM_Write_MR(interface, chip_select, 0x3, data);
/*end of MR3 settings*/
<snip>
}
```

- From MSM to DDR device: ROUT is for output drive strength from the MSM perspective.

  It is possible to increase or decrease the drive strength by changing the bit (XX_ROUT) value; but resistance is not linear, that is, to set a higher output drive strength than the default value of QTI, set ROUT to a higher value than the default value.

NOTE: There is a positive correlation between ROUT and the output drive strength.

```
 boot_images\core\boot\ddr\hw\msm8x10\ddr_config.c
uint32 ddr_caphy_config_MSM8x10[][2] =
{
<snip>
{HWIO_ADDR(PHY_CA_ADDR(PAD_CFG0)), 0x20222230}, // CA_ROUT[6:4] !!!Please
check source code and SWI!!!
{HWIO_ADDR(PHY_CA_ADDR(PAD_CFG1)), 0x20220430}, // CK_ROUT[6:4] !!!Please
check source code and SWI!!!
<snip>
}
<snip>
uint32 ddr_dqphy_config_MSM8x10[][2] =
{
<snip>
{HWIO_ADDR(PHY_DQ_ADDR(PAD_CFG0)), 0xA0222230}, // DQ_ROUT[6:4] !!!Please
check source code and SWI!!!
{HWIO_ADDR(PHY_DQ_ADDR(PAD_CFG1)), 0xA0222230}, // DQS_ROUT[6:4] !!!Please
check source code and SWI!!!
<snip>
}
```

4.2.1.4 MSM8926

- From DDR device to MSM: MR3 is for input drive strength from the MSM perspective.

```
boot_images\core\boot\ddr\hw\controller\BIMC_SHKE_v1.c
void HAL_SDRAM_SHKE_Device_Init(SDRAM_INTERFACE interface, SDRAM_CHIPSELECT
chip_select, uint32 clk_speed)
{
<snip>
/* Program MR2 */
<snip>
HAL_SDRAM_Write_MR(interface, chip_select, 0x2, data);
/* @#@ here */
/* Program MR3 */
data = 0x1; // 34.3 Ω for example. Please refer to your datasheet for the
impedance value range
HAL_SDRAM_Write_MR(interface, chip_select, 0x3, data);
/* @#@ here */

/* Mark ddr as initialized */
if(chip_select & SDRAM_CS0)
<snip>
}
```

- From MSM to DDR device: ROUT is for output drive strength from the MSM perspective during the write operation.

  It is possible to decrease the drive strength by decreasing the bit (XX_ROUT) value; but resistance is not linear, that is, to set a higher output drive strength than the default value of QTI, set ROUT to a higher value than the default value.

NOTE: There is a positive correlation between ROUT and the output drive strength.

```
boot_images\core\boot\ddr\hw\msm8x26\ddr_config.c
uint32 ddr_caphy_config_MSM8x26[][2] =
{
<snip>
{HWIO_ADDR(PHY_CA_ADDR(PAD_CFG0)), 0x20222250}, // CA_ROUT[6:4]
{HWIO_ADDR(PHY_CA_ADDR(PAD_CFG1)), 0x20220450}, // CK_ROUT[6:4]
<snip>
}
<snip>
uint32 ddr_dqphy_config_MSM8x26[][2] =
{
```

```
<snip>
{HWIO_ADDR(PHY_DQ_ADDR(PAD_CFG0)), 0xA0222250}, // DQ_ROUT[6:4]
{HWIO_ADDR(PHY_DQ_ADDR(PAD_CFG1)), 0xA0222250}, // DQS_ROUT[6:4]
<snip>
}
```

4.2.1.5 MSM8916

- For DDR side DS:

Change cas_latency’s high 16-bits in jedec_lpddr3_single_channel.xml or change the following codes in SBL1:

```
HAL_SDRAM_SHKE_Device_Init()
{
 ..
data = ((ddr_params->cas_latency) >> 0x10) & 0xF;
 if (data != 0)
 {
  HAL_SDRAM_Write_MR(base, interface, chip_select, 0x3, data);
 }
 else
 {
  /* Configure MR3 for optimal drive strength */
  HAL_SDRAM_Write_MR(base, interface, chip_select, 0x3, 0x1);
 }
+ HAL_SDRAM_Write_MR(base, interface, chip_select, 0x3, xxxx); //set the
value you want
..
}
```

- For MSM side DS:

From MSM to DDR device, ROUT is for output drive strength from the MSM perspective. Refer *MSM8916 Hardware Register Description Document for OEMS* (80-NK807-2x) for details.

It is possible to decrease or increase the drive strength by decreasing or increasing the ROUT values.

The registers are:

```
- DIM_C00_DIM_CA_PAD_CFG0(default 0xE0222240), CA_ROUT
- DIM_C00_DIM_CA_PAD_CFG1(default 0xE0220440), CK_ROUT
- DIM_D0x_DIM_DQ_PAD_CFG0(default 0xE0222240), DQ_ROUT (x:0~3)
- DIM_D0x_DIM_DQ_PAD_CFG1(default 0xE0222240), DQS_ROUT (x:0~3)
- DIM_D0x_DIM_DQ_PAD_CFG4(default 0xE0222240), RCW_ROUT (x:0~3)
```

It is possible to enter settings in ddr_config.c in MSM8x16_LPDDR2[][2] by appending the entries. This applies to LPDDR2 and LPDDR3.

Change bit-4, bit-5 and bit-6 about PHY_CA_ADDR(PAD_CFG0/1) and PHY_DQ_ADDR(PAD_CFG0/1/4)

```
 {HWIO_ADDR(PHY_CA_ADDR(PAD_CFG0)), 0x20222250},
 {HWIO_ADDR(PHY_CA_ADDR(PAD_CFG1)), 0x20220450},

 {HWIO_ADDR(PHY_DQ_ADDR(PAD_CFG0)), 0xA0222250},
 {HWIO_ADDR(PHY_DQ_ADDR(PAD_CFG1)), 0xA0222250},
 {HWIO_ADDR(PHY_DQ_ADDR(PAD_CFG4)), 0x20222250},
```

4.2.1.6 MSM8952

- Change MR3 to use Max D/S MR3='1 '

```
Write MR3 function to
'${BUILDPATH}/core/boot/ddr/hw/hw_sequence/BIMC/v2.2/bimc_mc_shke.c', /
ABHN_SHKE_Device_Init_LPDDR3()
MR3 change from default 2 to 1(40Ω to 34.3 Ω)
==================================================
Write MR3 function to
'${BUILDPATH}/core/boot/ddr/hw/hw_sequence/BIMC/v2.2/bimc_mc_shke.c', /
ABHN_SHKE_Device_Init_LPDDR3()
void ABHN_SHKE_Device_Init_LPDDR3( uint32 _inst_, uint32 chip_select,
uint32 clk_freq_in_khz, uint32 MR1_value, uint32 MR2_value )
{
…
 ABHN_SHKE_Write_MR(_inst_,
   chip_select,
   0x2,
   MR2_value);
 /// ------------------
 /// mode register 1 (moving after MR2 setting is done)
 /// ------------------
 tmp = HWIO_INX (_inst_, ABHN_SHKE_DRAM_MANUAL_1 );
 tmp = (tmp & ~(HWIO_ABHN_SHKE_DRAM_MANUAL_1_WAIT_TIMER_DOMAIN_BMSK)) |
((HWIO_ABHN_SHKE_DRAM_MANUAL_1_WAIT_TIMER_DOMAIN_XO_CLOCK_FVAL) <<
HWIO_ABHN_SHKE_DRAM_MANUAL_1_WAIT_TIMER_DOMAIN_SHFT ) ;
 tmp = (tmp &
~(HWIO_ABHN_SHKE_DRAM_MANUAL_1_WAIT_TIMER_BEFORE_HW_CLEAR_BMSK)) | ((0x5)
<< HWIO_ABHN_SHKE_DRAM_MANUAL_1_WAIT_TIMER_BEFORE_HW_CLEAR_SHFT ) ;
 HWIO_OUTX (_inst_, ABHN_SHKE_DRAM_MANUAL_1, tmp );

 ABHN_SHKE_Write_MR(_inst_,
   chip_select,
   0x1,
   MR1_value);
 /// ------------------
 + /// mode register 3 (DEBUG for Samsung, set ddr output driver
strength(MR3) to 0x1--34.3Ω)
 + /// ------------------
 + tmp = HWIO_INX (_inst_, ABHN_SHKE_DRAM_MANUAL_1 );
 + tmp = (tmp & ~(HWIO_ABHN_SHKE_DRAM_MANUAL_1_WAIT_TIMER_DOMAIN_BMSK)) |
((HWIO_ABHN_SHKE_DRAM_MANUAL_1_WAIT_TIMER_DOMAIN_XO_CLOCK_FVAL) <<
HWIO_ABHN_SHKE_DRAM_MANUAL_1_WAIT_TIMER_DOMAIN_SHFT ) ;
 + tmp = (tmp &
~(HWIO_ABHN_SHKE_DRAM_MANUAL_1_WAIT_TIMER_BEFORE_HW_CLEAR_BMSK)) | ((0x5)
<< HWIO_ABHN_SHKE_DRAM_MANUAL_1_WAIT_TIMER_BEFORE_HW_CLEAR_SHFT ) ;
 + HWIO_OUTX (_inst_, ABHN_SHKE_DRAM_MANUAL_1, tmp );
 +
 + ABHN_SHKE_Write_MR(_inst_,
 + chip_select,
 + 0x3,
 + 0x1); //parameter 0x1 is 0x1--34.3Ω
 + /// ------------------
  /// clean up
  /// ------------------
 HWIO_OUTXF (_inst_, ABHN_SHKE_DRAM_MANUAL_1, WAIT_TIMER_BEFORE_HW_CLEAR,
0);
…
```

- Change Rout to ‘6’

```
// Write DQ/DQS Rout = 0x6, SLEW=0x3
boot_images/core/boot/ddr/hw/hw_sequence/PHY/v2.5/ddr_phy_config_8952.h
-#define DDRPHY_DQ_PAD_CFG0_RECVAL 0x270FF140
+#define DDRPHY_DQ_PAD_CFG0_RECVAL 0x270FF160
-#define  DDRPHY_DQ_PAD_CFG1_RECVAL           0x170F00C0
+#define  DDRPHY_DQ_PAD_CFG1_RECVAL           0x170F00E0 //please only
change bits 6:4
// CA/CS/CK ROUT=0x6, SLEW=0x3
```

boot_images/core/boot/ddr/hw/hw_sequence/PHY/v2.5/ddr_phy_config_8952.h

```
- #define DDRPHY_CA_PAD_CFG0_RECVAL 0x200FF140
+ #define DDRPHY_CA_PAD_CFG0_RECVAL 0x200FF160
-#define  DDRPHY_CA_PAD_CFG1_RECVAL           0x200F00C0
+#define  DDRPHY_CA_PAD_CFG1_RECVAL           0x200F00E0 //please only
change bits 6:4
```

4.2.1.7 MSM8937 and MSM8940

From MSM to DDR device: ROUT is for output driver strength from the MSM perspective.

```
boot_images\core\boot\ddr\hw\hw_sequence\PHY\v2.5\ddr_phy_config_8937.h
<snip>
#define DDRPHY_CA_PAD_CFG0_RECVAL 0x200FF140 // Bit 6:4 has CA_ROUT.
#define DDRPHY_CA_PAD_CFG1_RECVAL 0x200F00C0 // Bit 6:4 has CK_ROUT
<snip>
#define DDRPHY_DQ_PAD_CFG0_RECVAL 0x270FF140 // Bit 6:4 has DQ_ROUT
#define DDRPHY_DQ_PAD_CFG1_RECVAL 0x170F00C0 // Bit 6:4 has DQS_ROUT
```

NOTE: 0x5 – 30 Ω, 0x6 – 28 Ω, 0x7 – 24 Ω and others use 240 Ω / (ROUT + 3).

It is possible to decrease the driver strength by decreasing the bit (XX_ROUT) value; but resistance is not linear, that is, to set a higher output driver strength than the default value, set ROUT to a higher value than the default QTI value.

NOTE: There is a positive correlation between ROUT and the output driver strength.

NOTE: Current ROUT has 0x4.

- From MSM to DDR device: MR3 is for input driver strength from the MSM perspective.

  MR3 is not programmed. To program it, add the following sequence. Also, MR3 is write-only register. Contact your memory vendor for the default MR3 value.

```
 boot_images\core\boot\ddr\hw\hw_sequence\BIMC\v2.2\bimc_mc_shke.c
__attribute__((section("ddr_boot_funcs")))
void ABHN_SHKE_Device_Init_LPDDR3( uint32 _inst_, uint32 chip_select,
uint32 clk_freq_in_khz, uint32 MR1_value, uint32 MR2_value )
{
#if ENABLE_LPDDR3
<snip>
/// ------------------
/// mode register 1 (moving after MR2 setting is done)
/// ------------------
tmp = HWIO_INX (_inst_, ABHN_SHKE_DRAM_MANUAL_1 );
tmp = (tmp & ~(HWIO_ABHN_SHKE_DRAM_MANUAL_1_WAIT_TIMER_DOMAIN_BMSK)) |
((HWIO_ABHN_SHKE_DRAM_MANUAL_1_WAIT_TIMER_DOMAIN_XO_CLOCK_FVAL) <<
HWIO_ABHN_SHKE_DRAM_MANUAL_1_WAIT_TIMER_DOMAIN_SHFT ) ;
tmp = (tmp &
~(HWIO_ABHN_SHKE_DRAM_MANUAL_1_WAIT_TIMER_BEFORE_HW_CLEAR_BMSK)) | ((0x5)
<< HWIO_ABHN_SHKE_DRAM_MANUAL_1_WAIT_TIMER_BEFORE_HW_CLEAR_SHFT ) ;
HWIO_OUTX (_inst_, ABHN_SHKE_DRAM_MANUAL_1, tmp );

ABHN_SHKE_Write_MR(_inst_,
chip_select,
0x1,
MR1_value);
+
+ /// ------------------
+ /// mode register 3
+ /// ------------------
+ tmp = HWIO_INX (_inst_, ABHN_SHKE_DRAM_MANUAL_1 );
+ tmp = (tmp & ~(HWIO_ABHN_SHKE_DRAM_MANUAL_1_WAIT_TIMER_DOMAIN_BMSK)) |
((HWIO_ABHN_SHKE_DRAM_MANUAL_1_WAIT_TIMER_DOMAIN_XO_CLOCK_FVAL) <<
HWIO_ABHN_SHKE_DRAM_MANUAL_1_WAIT_TIMER_DOMAIN_SHFT ) ;
+ tmp = (tmp &
~(HWIO_ABHN_SHKE_DRAM_MANUAL_1_WAIT_TIMER_BEFORE_HW_CLEAR_BMSK)) | ((0x5)
<< HWIO_ABHN_SHKE_DRAM_MANUAL_1_WAIT_TIMER_BEFORE_HW_CLEAR_SHFT ) ;
+ HWIO_OUTX (_inst_, ABHN_SHKE_DRAM_MANUAL_1, tmp );
+
+ ABHN_SHKE_Write_MR(_inst_,
+ chip_select,
+ 0x3,
+ <MR3 value>); // You can replace <MR3 value> with 1, 2, 3 or else based
on DDR datasheet.
/// ------------------
/// clean up
/// ------------------
```

4.2.1.8 MSM8917

- From MSM to DDR device: ROUT is for the output driver strength from the MSM perspective.

```
/BOOT.BF.3.3/boot_images/core/boot/ddr/hw/msm8917/ddr_config.c

<snip>
 {HWIO_ADDR(PHY_CA_ADDR(PAD_CFG0)), 0x00222260},// Bit 6:4 has CA_ROUT.
 {HWIO_ADDR(PHY_CA_ADDR(PAD_CFG1)), 0x00220460},// Bit 6:4 has CK_ROUT
<snip>
 {HWIO_ADDR(PHY_DQ_ADDR(PAD_CFG0)), 0x80222250},// Bit 6:4 has DQ_ROUT
 {HWIO_ADDR(PHY_DQ_ADDR(PAD_CFG1)), 0x80222250},// Bit 6:4 has DQS_ROUT
<snip>
```

NOTE: 0x5 – 30 Ω, 0x6 – 28 Ω, 0x7 – 24 Ω and others use 240 Ω (ROUT + 3).

You can decrease the driver strength by decreasing the bit (XX_ROUT) value; but resistance is not linear, that is, if you want to set a higher output driver strength than the default value of QTI, set ROUT to a higher value than the default value.

NOTE: There is a positive correlation between ROUT and the output driver strength.

NOTE: Current ROUT has 0x5 and 0x6.

- From DDR device to MSM: MR3 is for the input driver strength from the MSM perspective.
- MR3 is not programmed. It can be done by adding the following sequence. Also, MR3 is write-only register. Hence, contact your memory vendor for the default MR3 value.

boot_images\core\boot\ddr\hw\hw_sequence\BIMC\v2.2\bimc_mc_shke.c

```
__attribute__((section("ddr_boot_funcs")))
void ABHN_SHKE_Device_Init_LPDDR3( uint32 _inst_, uint32 chip_select,
uint32 clk_freq_in_khz, uint32 MR1_value, uint32 MR2_value )
{
#if ENABLE_LPDDR3
<snip>
/// ------------------
/// mode register 1 (moving after MR2 setting is done)
/// ------------------
```

```
tmp = HWIO_INX (_inst_, ABHN_SHKE_DRAM_MANUAL_1 );
tmp = (tmp & ~(HWIO_ABHN_SHKE_DRAM_MANUAL_1_WAIT_TIMER_DOMAIN_BMSK)) |
((HWIO_ABHN_SHKE_DRAM_MANUAL_1_WAIT_TIMER_DOMAIN_XO_CLOCK_FVAL) <<
HWIO_ABHN_SHKE_DRAM_MANUAL_1_WAIT_TIMER_DOMAIN_SHFT ) ;
tmp = (tmp &
~(HWIO_ABHN_SHKE_DRAM_MANUAL_1_WAIT_TIMER_BEFORE_HW_CLEAR_BMSK)) | ((0x5)
<< HWIO_ABHN_SHKE_DRAM_MANUAL_1_WAIT_TIMER_BEFORE_HW_CLEAR_SHFT ) ;
HWIO_OUTX (_inst_, ABHN_SHKE_DRAM_MANUAL_1, tmp );

ABHN_SHKE_Write_MR(_inst_,
chip_select,
0x1,
MR1_value);
+
+ /// ------------------
+ /// mode register 3
+ /// ------------------
+ tmp = HWIO_INX (_inst_, ABHN_SHKE_DRAM_MANUAL_1 );
+ tmp = (tmp & ~(HWIO_ABHN_SHKE_DRAM_MANUAL_1_WAIT_TIMER_DOMAIN_BMSK)) |
((HWIO_ABHN_SHKE_DRAM_MANUAL_1_WAIT_TIMER_DOMAIN_XO_CLOCK_FVAL) <<
HWIO_ABHN_SHKE_DRAM_MANUAL_1_WAIT_TIMER_DOMAIN_SHFT ) ;
+ tmp = (tmp &
~(HWIO_ABHN_SHKE_DRAM_MANUAL_1_WAIT_TIMER_BEFORE_HW_CLEAR_BMSK)) | ((0x5)
<< HWIO_ABHN_SHKE_DRAM_MANUAL_1_WAIT_TIMER_BEFORE_HW_CLEAR_SHFT ) ;
+ HWIO_OUTX (_inst_, ABHN_SHKE_DRAM_MANUAL_1, tmp );
+
+ ABHN_SHKE_Write_MR(_inst_,
+ chip_select,
+ 0x3,
+ <MR3 value>); // You can replace <MR3 value> with 1, 2, 3 or else based
on DDR datasheet.
/// ------------------
/// clean up
/// ------------------
```

4.2.1.9 MSM8994

- From DDR device to MSM: MR3 is for the input drive strength from the MSM perspective.

```
boot_images\core\boot\ddr\hw\hw_sequence\8994\ddrss\bimc\mc230\src\bimc_ini
t.c
void BIMC_Memory_Device_Init (DDR_STRUCT *ddr, DDR_CHANNEL channel,
DDR_CHIPSELECT chip_select,
uint32 clk_freq_khz)
{
uint8 ch = 0x0;
uint32 reg_offset_shke = 0;
BIMC_Memory_Device_Init_Lpddr (ddr, channel, chip_select);
BIMC_ZQ_Calibration (ddr, channel, chip_select);
// RL and WL MR write
BIMC_MR_Write (channel, chip_select, JEDEC_MR_2, BIMC_RL_WL_Table_Sel(ddr,
MR2_WR_VAL, clk_freq_khz));
+ //Set MR3
+ BIMC_MR_Write (channel, chip_select, JEDEC_MR_3, <MR_data_for_MR3>);
<snip>
}
```

ROUT settings are adjusted through Frequency Power Manager (FPM).

FRM registers:

Registers:

CA/CK:

CHx_CAy_DDR_PHY_DDRPHY_FPM_PRFS_z_PWRS_1_LO_CFG

DQ/DQS

CHx_DQy_DDR_PHY_DDRPHY_FPM_PRFS_z_PWRS_1_LO_CFG where,

x: channel

y: CA or DQ index

z: performance band

Performance band:

#define F_RANGE_0 200000 // PRFS band 0

#define F_RANGE_1 558000 // PRFS band 1

#define F_RANGE_2 868000 // PRFS band 2

#define F_RANGE_3 1116000 //PRFS band 3

Bit information:

DQ/DQS:

bit [14-12]: pull-up DQS

bit [11-9]: pull-up DQ

bit [8-6]: pull-down DQS

bit [5-3]: pull-down DQ

CK/CA:

bit [14-12]: pull-up CK

bit [11-9]: pull-up CA

bit [8-6]: pull-down CK

bit [5-3]: pull-down CA

Codes to change:

```
file :
\boot_images\core\boot\ddr\hw\hw_sequence\8994\target\8994\settings\ddr_phy
_config.c
```

DQ/DQS:

uint32 ddr_phy_dq_config[][2]

CK/CA:

uint32 ddr_phy_ca0_config[][2]

uint32 ddr_phy_ca1_config[][2]

The resistance value equal to 240Ω / (code + 1).

```
   {HWIO_ADDRX(0, DDR_PHY_DDRPHY_FPM_PRFS_0_PWRS_1_LO_CFG), 0x00002492}, //
ROUT 2 for Band0
  {HWIO_ADDRX(0, DDR_PHY_DDRPHY_FPM_PRFS_1_PWRS_1_LO_CFG), 0x28A036D8}, //
ROUT 3 for Band1
  {HWIO_ADDRX(0, DDR_PHY_DDRPHY_FPM_PRFS_2_PWRS_1_LO_CFG), 0x28A04920}, //
ROUT 4 for Band2
  {HWIO_ADDRX(0, DDR_PHY_DDRPHY_FPM_PRFS_3_PWRS_1_LO_CFG), 0x28A04920}, //
ROUT 4 for Band3
```

4.2.1.10 MSM8953

NOTE: For drive strength; only for debugging purpose. It is not suggested to change the default settings.

- From DDR device to MSM: MR3 is for the input driver strength from the MSM perspective.
- MR3 is not programmed. To program it, add the following sequence. Also, MR3 is a write-only register. Hence, contact your memory vendor for the default MR3 value.

MR3:

```
boot_images\core\boot\ddr\hw\hw_sequence\8953\ddrss\bimc\mc230\src\bimc_ini
t.c
void BIMC_Memory_Device_Init (DDR_STRUCT *ddr, DDR_CHANNEL channel,
DDR_CHIPSELECT chip_select,
uint32 clk_freq_khz)
{
uint8 ch = 0x0;
uint32 reg_offset_shke = 0;
BIMC_Memory_Device_Init_Lpddr (ddr, channel, chip_select);
BIMC_ZQ_Calibration (ddr, channel, chip_select);
// RL and WL MR write
BIMC_MR_Write (channel, chip_select, JEDEC_MR_2, BIMC_RL_WL_Table_Sel(ddr,
MR2_WR_VAL, clk_freq_khz));
+ //Set MR3
+ BIMC_MR_Write (channel, chip_select, JEDEC_MR_3, <MR_data_for_MR3>);
<snip>
}
```

- From MSM to DDR device:

ROUT settings are adjusted through FPM (Frequency Power Manager).

FRM registers:

Registers:

CA/CK:

CHx_CAy_DDR_PHY_DDRPHY_FPM_PRFS_z_PWRS_1_LO_CFG

DQ/DQS

CHx_DQy_DDR_PHY_DDRPHY_FPM_PRFS_z_PWRS_1_LO_CFG where,

x: channel

y: CA or DQ index

z: performance band

Performance band:

boot_images/core/boot/ddr/hw/hw_sequence/msm8953/target/msm8953/header/ddr_phy_technology.h

```
 #define F_RANGE_0          250000
#define F_RANGE_1          450000
#define F_RANGE_2          790000
#define F_RANGE_3          900000
#define F_RANGE_4          1066000
#define F_RANGE_5          1066000
#define F_RANGE_6          1066000
#define F_RANGE_7          1066000
```

Bit information:

DQ/DQS:

bit [14-12]: pull-up DQS

bit [11-9]: pull-up DQ

bit [8-6]: pull-down DQS

bit [5-3]: pull-down DQ

CK/CA

bit [14-12]: pull-up CK

bit [11-9]: pull-up CA

bit [8-6]: pull-down CK

bit [5-3]: pull-down CA

Codes to change:

File:

```
boot_images/core/boot/ddr/hw/hw_sequence/msm8953/target/msm8953/settings/ddr_p
hy_config.c
```

DQ/DQS:

uint32 ddr_phy_dq_config[][2]

CK/CA:

uint32 ddr_phy_ca0_config[][2]

uint32 ddr_phy_ca1_config[][2]

The resistance value equal to 240Ω / (code + 1).

4.2.1.11 MSM8996

- MSM <- DDR device: MR3 is for the input driver strength from the MSM perspective.

```
File #1:
boot_images/QcomPkg/Msm8996Pkg/Library/DSFTargetLib/ddrss/bimc/mc230/src/bi
mc_lpddr4.c

#if TARGET_SILICON
// Enable DBI-WR, DBI-RD/PDDS/PU-CAL/WR_PST are set to default
//FSP = Settings
BIMC_MR_Write (channel, chip_select, JEDEC_MR_3, 0xB0); //10 < F <= 400MHz,
set 1 MR3[0],PU-CAL=VDDQ/2.5 for fsp0, it was 0xB1(VDDQ/3)
```

`BIMC_MR_Write (channel, chip_select, JEDEC_MR_3, 0xA8);` `//10 < F <= 400MHz,`

```
set 4 MR3[0],PU-CAL=VDDQ/2.5 for fsp0, it was 0xB1(VDDQ/3)
BIMC_MR_Write (channel, chip_select, JEDEC_MR_22, 0x0); //set SOC_ODT to
disable for fsp0. it was SOC_ODT=40Ω for FSP=0

File #2:
Boot_images/QcomPkg/Library/DDRLib/hw/hw_sequence/ddrss/bimc/mc230/src/bimc
_data_lpddr4.c

freq_switch_params_struct /* RD-DBI, ODT, FSP, MR1, MR3, MR11 */
bimc_freq_switch_params[] = {{ 0, 0, 0, 0x06, 0xB0, 0x00 }, // 10 < F <=
400MHz, WR-pre =1, DBI_WR=1

{ 0, 0, 0, 0x06, 0xB0, 0x00 }, // 400 < F <= 750MHz, PU-CAL=Vddq/2.5
{ 0, 0, 0, 0x06, 0xB0, 0x00 }, // 750 < F <= 1000MHz,
{ 0, 1, 1, 0x06, 0xB0, 0x33 }, // 1000 < F <= 1295MHz CA-ODT=Rzq/2, DQ-
ODT=Rzq/3
{ 0, 1, 1, 0x0E, 0xB0, 0x33 }, // 1295 < F <= 1570MHz RD-PRE=1, DQ ODT =
80Ω
{ 0, 1, 1, 0x0E, 0xB2, 0x34 }}; // 1570 < F <= 1890MHz WR-PST =1,DQ ODT =
60Ω

freq_switch_params_struct /* RD-DBI, ODT, FSP, MR1, MR3, MR11 */
```

```
bimc_freq_switch_params[] = {{ 0, 0, 0, 0x06, 0xA8, 0x00 }, // 10 < F <=
400MHz, WR-27 pre =1, DBI_WR=1
{ 0, 0, 0, 0x06, 0xA8, 0x00 }, // 400 < F <= 750MHz, PU-CAL=Vddq/2.5
{ 0, 0, 0, 0x06, 0xA8, 0x00 }, // 750 < F <= 1000MHz,
{ 0, 1, 1, 0x06, 0xA8, 0x33 }, // 1000 < F <= 1295MHz CA-ODT=Rzq/2, DQ-
ODT=Rzq/3
{ 0, 1, 1, 0x0E, 0xA8, 0x33 }, // 1295 < F <= 1570MHz RD-PRE=1, DQ ODT = 80Ω
{ 0, 1, 1, 0x0E, 0xAA, 0x34 }}; // 1570 < F <= 1890MHz WR-PST =1, DQ ODT =
60Ω
```

- From MSM to DDR device:

File: ddr_phy_config.c

uint32 ddr_phy_dq_config[][2]

uint32 ddr_phy_ca_config[][2]

Registers:

DDR_PHY_DDRPHY_FPM_PRFS_*x*_PWRS_1_LO_CFG

, where x is performance (PRFS) band number

DQ/DQS:

bit [14-12]: pull-down DQS

bit [11-9]: pull-up DQ

bit [8-6]: pull-down DQS

bit [5-3]: pull-down DQ

CK/CA:

bit [14-12]: pull-down CK

bit [11-9]: pull-up CA

bit [8-6]: pull-downCK

bit [5-3]: pull-down CA

Impedance value:

Impedance will be 240 Ωs / (code + 1)

0x0: 240, 0x1: 120, 0x2: 80, 0x3: 60, 0x4: 48, 0x5: 40, 0x6: 34, 0x7: 30

- ODT configuration
  - MSM side

File : ddr_phy_config.c

```
uint32 ddr_phy_dq_config[][2]
uint32 ddr_phy_ca_config[][2]
```

Registers:

```
DDR_PHY_DDRPHY_FPM_PRFS_x_PWRS_1_LO_CFG
DDR_PHY_DDRPHY_FPM_PRFS_x_PWRS_1_HI_CFG
```

, where x is performance (PRFS) band number.

ODT enable:

```
DQ: DDRPHY_FPM_PRFS_x_PWRS_1_HI_CFG[5]: DQ-ODT enable
DQS: DDRPHY_FPM_PRFS_x_PWRS_1_HI_CFG[6]: DQS-ODT enable
```

ODT_DQ:

```
DDRPHY_FPM_PRFS_x_PWRS_1_LO_CFG[31]
DDRPHY_FPM_PRFS_x_PWRS_1_HI_CFG[1:0]
```

ODT_DQS:

```
DDRPHY_FPM_PRFS_x_PWRS_1_HI_CFG[4:2]
```

Impedance will be 240 Ωs / (code + 1)

0x0: 240, 0x1: 120, 0x2: 80, 0x3: 60, 0x4: 48, 0x5: 40, 0x6: 34, 0x7: 30

- DDR side (MR11)

```
File #1
/boot_images/QcomPkg/Library/DDRLib/hw/hw_sequence/ddrss/bimc/mc230/src/bim
c_lpddr4.c

Codes highlighted in red are values to change.

void BIMC_Memory_Device_Init_Lpddr (DDR_STRUCT *ddr, DDR_CHANNEL channel ,
DDR_CHIPSELECT chip_select)
{
<SNIP>
#if TARGET_SILICON
// Enable DBI-WR, DBI-RD/PDDS/PU-CAL/WR_PST are set to default
//FSP = Settings
BIMC_MR_Write (channel, chip_select, JEDEC_MR_3, 0xB0); //10 < F <= 400MHz, set
MR3[0],PU-CAL=VDDQ/2.5 for fsp0, it was 0xB1(VDDQ/3)
<SNIP>
// Set ODT
BIMC_MR_Write (channel, chip_select, JEDEC_MR_11, 0x33); // was 0x34,DQ ODT
60Ω. Level 2: both CK and DQ ODT =80Ω.
```

```
File #2 -
/boot_images/QcomPkg/Library/DDRLib/hw/hw_sequence/ddrss/bimc/mc230/src/bim
c_data_lpddr4.c

freq_switch_params_struct /* RD-DBI, ODT, FSP, MR1, MR3, MR11 */
bimc_freq_switch_params[] = {{ 0, 0, 0, 0x06, 0xB0, 0x00 }, // 10 < F <=
400MHz, WR-pre =1, DBI_WR=1
{ 0, 0, 0, 0x06, 0xB0, 0x00 }, // 400 < F <= 750MHz, PU-CAL=Vddq/2.5
{ 0, 0, 0, 0x06, 0xB0, 0x00 }, // 750 < F <= 1000MHz,
{ 0, 1, 1, 0x06, 0xB0, 0x33 }, // 1000 < F <= 1295MHz CA-ODT=Rzq/2, DQ-
ODT=Rzq/3
{ 0, 1, 1, 0x0E, 0xB0, 0x33 }, // 1295 < F <= 1570MHz RD-PRE=1, DQ ODT = 80Ω
{ 0, 1, 1, 0x0E, 0xB2, 0x34 }}; // 1570 < F <= 1890MHz WR-PST =1, DQ ODT = 60Ω
```

4.2.1.12 SDM660 and SDM630

- From MSM to DDR device:

  Filename

  `boot_images\QcomPkg\Sdm660Pkg\Settings\DSF\boot\ddr_phy_config.c`

  For LPDDR4

  ```
  uint32 ddr_phy_dq_config[][2]
  uint32 ddr_phy_ca_config[][2]
  ```

  For LPDDR4X

  ```
  uint32 ddr_phy_dq_lp4x_config[][2]
  uint32 ddr_phy_ca_lp4x_config[][2]
  ```

  Registers

  DDR_PHY_DDRPHY_FPM_PRFS_x_PWRS_1_LO_CFG, *where x is performance (PRFS) band number*

  Bit information

DQ/DQS:

- bit [14-12]: pull-up DQS
- bit [11-9]: pull-up DQ
- bit [8-6]: pull-down DQS
- bit [5-3]: pull-down DQ

CK/CA

- bit [14-12]: pull-up CK
- bit [11-9]: pull-up CA
- bit [8-6]: pull-downCK
- bit [5-3]: pull-down CA

Example to change default (40 Ω) to 48 Ω

- From DDR device to MSM: MR3 is for input driver strength from the MSM perspective.

  File #1:

```
/boot_images/QcomPkg/Library/DDRLib/hw/hw_sequence/ddrss/bimc/mc230/src/bim
c_lpddr4.c
//FSP = Settings
      if (ddr->detected_ddr_device[0].device_type == DDR_TYPE_LPDDR4X) //
For LP4x apply these settings
      {
         BIMC_MR_Write (channel, chip_select, JEDEC_MR_3, ~~0xB0~~ 0xA8); //10
< F <= 400MHz, set MR3[0],PU-CAL=VDDQ/2.5 for fsp0, it was 0xB1(VDDQ/3)
      }
      else // for lp4 device apply these settings
      {
```

```
         BIMC_MR_Write (channel, chip_select, JEDEC_MR_3, ~~0xB0~~ 0xA8); //10
< F <= 400MHz, set MR3[0],PU-CAL=VDDQ/2.5 for fsp0, it was 0xB1(VDDQ/3)
      }
```

File #2:

```
\boot_images\QcomPkg\Sdm660Pkg\Library\DSFTargetLib\rpm\ddrss\bimc\mc230\sr
c\bimc_data_lpddr4_rpm.c
struct ecdt_bimc_freq_switch_params_runtime_struct
bimc_freq_switch_params_struct[] =
{
   /* RD-DBI, ODT, FSP, MR1,  MR3,  MR11, freq_switch_params_freq_range */
   { 0,    0,    0,    0x06,    ~~0xB0~~ 0xA8,    0x00,   400000}, /*  10  < F <=
400MHz, WR-pre =1, DBI_WR=1          */
   { 0,    0,    0,    0x06,    ~~0xB0~~ 0xA8,    0x00,   750000}, /* 400  < F <=
750MHz, PU-CAL=Vddq/2.5              */
   { 0,    0,    0,    0x06,    ~~0xB0~~ 0xA8,    0x00, 1000000}, /* 750  < F <=
1000MHz,  */
   { 0,    1,    1,    0x06,    ~~0xB0~~ 0xA8,    0x33, 1295000}, /* 1000 < F <=
1295MHz CA-ODT=Rzq/2, DQ-ODT=Rzq/3   */
   { 0,    1,    1,    0x0E,    ~~0xB0~~ 0xA8,    0x33, 1570000}, /* 1295 < F <=
1570MHz RD-PRE=1,  DQ ODT = 80Ω    */
   { 0,    1,    1,    0x0E,    ~~0xB2~~ 0xAA,    0x34, 1890000}, /* 1570 < F <=
1890MHz WR-PST =1, DQ ODT = 60Ω    */
   { 0,    0,    0,    0,       0,       0,       0}, /* reserved  */
   { 0,    0,    0,    0,       0,       0,       0}  /* reserved  */
};
```

- ODT configuration

MSM side

Filename

`boot_images\QcomPkg\Sdm660Pkg\Settings\DSF\boot\ddr_phy_config.c`

```
uint32 ddr_phy_dq_config[][2]
uint32 ddr_phy_ca_config[][2]
```

Registers:

```
DDR_PHY_DDRPHY_FPM_PRFS_x_PWRS_1_LO_CFG

DDR_PHY_DDRPHY_FPM_PRFS_x_PWRS_1_HI_CFG
```

```
where x is performance (PRFS) band number
```

Bit information

```
ODT enable:
DQ: DDRPHY_FPM_PRFS_x_PWRS_1_HI_CFG[5] : DQ-ODT enable
DQS: DDRPHY_FPM_PRFS_x_PWRS_1_HI_CFG[6] : DQS-ODT enable

ODT_DQ:
 DDRPHY_FPM_PRFS_x_PWRS_1_LO_CFG[31]
 DDRPHY_FPM_PRFS_x_PWRS_1_HI_CFG[1:0]

ODT_DQS:
DDRPHY_FPM_PRFS_x_PWRS_1_HI_CFG[4:2]
```

DDR side

File #1

```
/boot_images/QcomPkg/Library/DDRLib/hw/hw_sequence/ddrss/bimc/mc230/src/bim
c_lpddr4.c
```

NOTE: Codes highlighted in red are values to change.

```
void BIMC_Memory_Device_Init_Lpddr (DDR_STRUCT *ddr, DDR_CHANNEL channel ,
DDR_CHIPSELECT chip_select)
{
<SNIP>

#if TARGET_SILICON
// Enable DBI-WR, DBI-RD/PDDS/PU-CAL/WR_PST are set to default
//FSP = Settings
BIMC_MR_Write (channel, chip_select, JEDEC_MR_3, 0xB0); //10 < F <= 400MHz,
set MR3[0],PU-7 CAL=VDDQ/2.5 for fsp0 it was 0xB1(VDDQ/3)

<SNIP>

// Set ODT
BIMC_MR_Write (channel, chip_select, JEDEC_MR_11, 0x33); // was 0x34,, DQ
ODT 60Ω. Level 2: both CK and DQ ODT =80Ω

}
```

File #2:

```
\boot_images\QcomPkg\Sdm660Pkg\Library\DSFTargetLib\rpm\ddrss\bimc\mc230\sr
c\bimc_data_lpddr4_rpm.c

struct ecdt_bimc_freq_switch_params_runtime_struct
bimc_freq_switch_params_struct[] =
{
   /* RD-DBI, ODT, FSP, MR1,  MR3,  MR11, freq_switch_params_freq_range */
{ 0,   0,   0,   0x06,   0xB0,   0x00,  400000}, /*  10  < F <= 400MHz, WR-
pre =1, DBI_WR=1           */
{ 0,   0,   0,   0x06,   0xB0,   0x00,  750000}, /* 400  < F <= 750MHz, PU-
CAL=Vddq/2.5               */
{ 0,   0,   0,   0x06,   0xB0,   0x00, 1000000}, /* 750  < F <= 1000MHz, */
{ 0,   1,   1,   0x06,   0xB0,   0x33, 1295000}, /* 1000 < F <= 1295MHz CA-
ODT=Rzq/2, DQ-ODT=Rzq/3    */
{ 0,   1,   1,   0x0E,   0xB0,   0x33, 1570000}, /* 1295 < F <= 1570MHz RD-
PRE=1,  DQ ODT = 80Ω    */
{ 0,   1,   1,   0x0E,   0xB2,   0x34, 1890000}, /* 1570 < F <= 1890MHz WR-
PST =1, DQ ODT = 60Ω    */
{ 0,   0,   0,      0,      0,      0,       0}, /* reserved  */
{ 0,   0,   0,      0,      0,      0,       0}  /* reserved */
};
```

4.2.1.13 MSM8998

- From DDR device to MSM: MR3 is for input driver strength from the MSM perspective.

File #1:
/boot_images/QcomPkg/Msm8998Pkg/Library/DSFTargetLib/boot/ddrss/bimc/mc230/src/bimc_lpddr4.c

```
#if TARGET_SILICON
// Enable DBI-WR, DBI-RD/PDDS/PU-CAL/WR_PST are set to default
//FSP = Settings
BIMC_MR_Write (channel, chip_select, JEDEC_MR_3, 0xB0); //10 < F <= 400MHz,
set 1 MR3[0],PU-CAL=VDDQ/2.5 for fsp0, it was 0xB1(VDDQ/3)

BIMC_MR_Write (channel, chip_select, JEDEC_MR_3, 0xA8); //10 < F <= 400MHz,
set 4 MR3[0],PU-CAL=VDDQ/2.5 for fsp0, it was 0xB1(VDDQ/3)
BIMC_MR_Write (channel, chip_select, JEDEC_MR_22, 0x0); //set SOC_ODT to
disable for fsp0. it was SOC_ODT=40Ω for FSP=0
```

File #2:
Boot_images/QcomPkg/Msm8998Pkg/Library/DSFTargetLib/rpm/ddrss/bimc/mc230/src/bimc_data_lpddr4_rpm.c

```
freq_switch_params_struct /* RD-DBI, ODT, FSP, MR1, MR3, MR11 */
bimc_freq_switch_params[] = {{ 0, 0, 0, 0x06, 0xB0, 0x00 }, // 10 < F <=
400MHz, WR-pre =1, DBI_WR=1

{ 0, 0, 0, 0x06, 0xB0, 0x00 }, // 400 < F <= 750MHz, PU-CAL=Vddq/2.5
{ 0, 0, 0, 0x06, 0xB0, 0x00 }, // 750 < F <= 1000MHz,
{ 0, 1, 1, 0x06, 0xB0, 0x33 }, // 1000 < F <= 1295MHz CA-ODT=Rzq/2, DQ-
ODT=Rzq/3
{ 0, 1, 1, 0x0E, 0xB0, 0x33 }, // 1295 < F <= 1570MHz RD-PRE=1, DQ ODT =
80Ω
{ 0, 1, 1, 0x0E, 0xB2, 0x34 }}; // 1570 < F <= 1890MHz WR-PST =1,DQ ODT =
60Ω

freq_switch_params_struct /* RD-DBI, ODT, FSP, MR1, MR3, MR11 */
bimc_freq_switch_params[] = {{ 0, 0, 0, 0x06, 0xA8, 0x00 }, // 10 < F <=
400MHz, WR-27 pre =1, DBI_WR=1
{ 0, 0, 0, 0x06, 0xA8, 0x00 }, // 400 < F <= 750MHz, PU-CAL=Vddq/2.5
{ 0, 0, 0, 0x06, 0xA8, 0x00 }, // 750 < F <= 1000MHz,
{ 0, 1, 1, 0x06, 0xA8, 0x33 }, // 1000 < F <= 1295MHz CA-ODT=Rzq/2, DQ-
ODT=Rzq/3
{ 0, 1, 1, 0x0E, 0xA8, 0x33 }, // 1295 < F <= 1570MHz RD-PRE=1, DQ ODT = 80Ω
{ 0, 1, 1, 0x0E, 0xAA, 0x34 }}; // 1570 < F <= 1890MHz WR-PST =1, DQ ODT =
60Ω
```

- From MSM to DDR device:
  - File: ddr_phy_config.c
  - uint32 ddr_phy_dq_config[][2]
  - uint32 ddr_phy_ca_config[][2]

Registers:

DDR_PHY_DDRPHY_FPM_PRFS_*x*_PWRS_1_LO_CFG, *where x is performance (PRFS) band number*

DQ/DQS:

- bit [14-12]: pull-down DQS
- bit [11-9]: pull-up DQ
- bit [8-6]: pull-down DQS
- bit [5-3]: pull-down DQ

CK/CA:

- bit [14-12]: pull-down CK
- bit [11-9]: pull-up CA
- bit [8-6]: pull-downCK
- bit [5-3]: pull-down CA

Impedance value:

Impedance is 240 Ωs / (code + 1)

0x0: 240 , 0x1: 120 , 0x2: 80 , 0x3: 60 , 0x4: 48 , 0x5: 40 , 0x6: 34 , 0x7: 30

- ODT Configuration
  - MSM side

File : ddr_phy_config.c

```
uint32 ddr_phy_dq_config[][2]
uint32 ddr_phy_ca_config[][2]
```

Registers:

```
DDR_PHY_DDRPHY_FPM_PRFS_x_PWRS_1_LO_CFG
DDR_PHY_DDRPHY_FPM_PRFS_x_PWRS_1_HI_CFG
```

where x is performance (PRFS) band number.

ODT enable:

```
DQ: DDRPHY_FPM_PRFS_x_PWRS_1_HI_CFG[5] : DQ-ODT enable
DQS: DDRPHY_FPM_PRFS_x_PWRS_1_HI_CFG[6] : DQS-ODT enable
```

ODT_DQ:

```
DDRPHY_FPM_PRFS_x_PWRS_1_LO_CFG[31]
DDRPHY_FPM_PRFS_x_PWRS_1_HI_CFG[1:0]
```

ODT_DQS:

```
DDRPHY_FPM_PRFS_x_PWRS_1_HI_CFG[4:2]
```

Impedance is 240 Ωs / (code + 1)

0x0: 240 , 0x1: 120 , 0x2: 80 , 0x3: 60 , 0x4: 48 , 0x5: 40 , 0x6: 34 , 0x7: 30

- ▫ DDR side (MR11)

File #1
/boot_images/QcomPkg/Msm8998Pkg/Library/DSFTargetLib/boot/ddrss/bimc/mc230/src/bimc_lpddr4.c
Codes highlighted in red are values to change.

```
void BIMC_Memory_Device_Init_Lpddr (DDR_STRUCT *ddr, DDR_CHANNEL channel ,
DDR_CHIPSELECT chip_select)
{
<SNIP>

#if TARGET_SILICON
// Enable DBI-WR, DBI-RD/PDDS/PU-CAL/WR_PST are set to default
//FSP = Settings
BIMC_MR_Write (channel, chip_select, JEDEC_MR_3, 0xB0); //10 < F <= 400MHz, set
MR3[0],PU-CAL=VDDQ/2.5 for fsp0, it was 0xB1(VDDQ/3)

<SNIP>

// Set ODT
BIMC_MR_Write (channel, chip_select, JEDEC_MR_11, 0x33); // was 0x34,DQ ODT
60Ω. Level 2: both CK and DQ ODT =80Ω.
```

File #2
Boot_images/QcomPkg/Msm8998Pkg/Library/DSFTargetLib/rpm/ddrss/bimc/mc230/src/bimc_data_lpddr4_rpm.c

```
freq_switch_params_struct /* RD-DBI, ODT, FSP, MR1, MR3, MR11 */
bimc_freq_switch_params[] = {{ 0, 0, 0, 0x06, 0xB0, 0x00 }, // 10 < F <=
400MHz, WR-pre =1, DBI_WR=1
{ 0, 0, 0, 0x06, 0xB0, 0x00 }, // 400 < F <= 750MHz, PU-CAL=Vddq/2.5
{ 0, 0, 0, 0x06, 0xB0, 0x00 }, // 750 < F <= 1000MHz,
{ 0, 1, 1, 0x06, 0xB0, 0x33 }, // 1000 < F <= 1295MHz CA-ODT=Rzq/2, DQ-
ODT=Rzq/3
{ 0, 1, 1, 0x0E, 0xB0, 0x33 }, // 1295 < F <= 1570MHz RD-PRE=1, DQ ODT = 80Ω
{ 0, 1, 1, 0x0E, 0xB2, 0x34 }}; // 1570 < F <= 1890MHz WR-PST =1, DQ ODT = 60Ω
```

4.2.2 Drive strength tuning

The driver strength is a key number of DDR driver customization parameter. The settings determine the peak current sent on the line and delivered to the load. The largest timing margin can be secured with proper setting. The recommended driver strength is validated on QTI or QRD reference phones, but may be different from customers' board in terms of PCB layout and DRAM die output driver characteristics. Therefore, customers must tune driver strength setting, set up proper driver strength based on driver strength test results and recommendations from memory vendors. After changing the drive strength, it is possible to sweep the drive strength to tune it for stability issues.

Different platforms have different MR3 and ROUT values. Here we take MSM8952 as an example.

4.2.2.1 MR3

MR3=1	MR3=2	MR3=3
34.3 Ω	40 Ω	48 Ω

4.2.2.2 ROUT

A set of CH0_xxx_DDRPHY_xx_PAD_CFGx registers sets the chipset drive strength.

Register	Address	Bits	Field name	Software default	Comments
CH0_CA_DDRPHY_CA_PAD_CFG0	0x00480010	6:4	CA_ROUT	4	h5:30 Ω; 'h6:28 Ω;'h7:24 Ω (others use 240 Ωs/(Rout+3))
CH0_CA_DDRPHY_CA_PAD_CFG1	0x00480014	6:4	CK_ROUT	4	h5:30 Ω; 'h6:28 Ω;'h7:24 Ω (others use 240 Ωs/(Rout+3))
CH0_DQ0_DDRPHY_DQ_PAD_CFG0	0x00480810	6:4	DQ_ROUT (Byte0, Byte2)	4	h5:30 Ω; 'h6:28 Ω;'h7:24 Ω (others use 240 Ωs/(Rout+3))
CH0_DQ0_DDRPHY_DQ_PAD_CFG1	0x00480814	6:4	DQS_ROUT (Byte0, Byte2)	4	h5:30 Ω; 'h6:28 Ω;'h7:24 Ω (others use 240 Ωs/(Rout+3))
CH0_DQ1_DDRPHY_DQ_PAD_CFG0	0x00481010	6:4	DQ_ROUT (Byte1, Byte3)	4	h5:30 Ω; 'h6:28 Ω;'h7:24 Ω (others use 240 Ωs/(Rout+3))

Register	Address	Bits	Field name	Software default	Comments
CH0_DQ1_DDRPHY_DQ_PAD_CFG1	0x00481014	6:4	DQS_ROUT (Byte1, Byte3)	4	h5:30 Ω; 'h6:28 Ω;'h7:24 Ω (others use 240 Ωs/(Rout+3))

4.2.2.3 Tuning process

1. Sweep Rout, changing Rout by variety and setup MR3 to hardware DDR simulation recommended value. Change Rout or MR3 values, "fastboot erase DDR" partition to retrain DDR. Perform the DDR stress test for at least 3 phones and 12 hours. Record the following table for pass and fail results.

Rout	Test result
Rout=3	F
Rout=4	P <= best option in the middle of the pass area
Rout=5	P
Rout=6	P
Rout=7	F

2. Use Rout sweep MR3. Change Rout or MR3 values, "fastboot erase DDR" partition to retrain DDR. Perform the DDR stress test for at least 3 phones and 12 hours. Record the following table for pass and fail results.

MR3	Test result
1	P
2	P <= best option is the minimal pass setting
3	F

4.3 DDR voltage

This change is for general DDR stability issues, no specific pattern. Raise a PMIC case in case of queries.

4.3.1 LPDDR3

Bumping up the voltage refers to the DDR VDD voltage. LDO2 is used to confirm the source of the voltage.

4.3.1.1 MSM8952

```
1)
boot_images/core/systemdrivers/pmic/framework/src/pm_init.c

pm_err_flag_type pm_oem_init ( void )
{
```

```
pm_err_flag_type errFlag = PM_ERR_FLAG__SUCCESS;
+ err_flag = pm_ldo_volt_level(0, PM_LDO_2, 1250000);
return errFlag;

}

2)

rpm_proc/core/systemdrivers/pmic/config/msm8952/pm_config_target.c

pm_rpm_ldo_rail_info_type ldo_rail_a[] =
{
  {5, 62.5,  0, PM_ACCESS_ALLOWED, PM_NONE,   PM_NPA_SW_MODE_LDO__NPM,
PM_NPA_BYPASS_DISALLOWED, PM_DROOP_DETECT_DIS, 1000, 1050, 0,
PM_SETTLING_ERR_EN, PM_SETTLING_EN, 0}, // LDO1  ULT N600_Stepper
-  {5, 62.5,  0, PM_ACCESS_ALLOWED, PM_ALWAYS_ON,
PM_NPA_SW_MODE_LDO__IPEAK, PM_NPA_BYPASS_DISALLOWED, PM_DROOP_DETECT_DIS,
1200, 1300, 0, PM_SETTLING_ERR_EN, PM_SETTLING_EN, 0},  // LDO2  ULT
N600_Stepper
+  {5, 62.5,  0, PM_ACCESS_DISALLOWED, PM_ALWAYS_ON,
PM_NPA_SW_MODE_LDO__IPEAK, PM_NPA_BYPASS_DISALLOWED, PM_DROOP_DETECT_DIS,
1250, 1300, 0, PM_SETTLING_ERR_EN, PM_SETTLING_EN, 0},  // LDO2  ULT
N600_Stepper
```

3)

```
core/systemdrivers/pmic/config/msm8952/pm_config_rpm_npa_pam.c\pm_config_rp
m_npa_pam.c

/* LPDDR Client */
static pm_npa_ldo_int_rep
pm_rpm_pam_lpddr_a_ldo2[] =
{
  /**< Mode: PMIC_NPA_MODE_ID_DDR_SLEEP_SPEED*/
{
   PM_NPA_GENERIC_DISABLE, /**< [Disable (default), Enable] -> max
aggregation (left to right). */
   PM_NPA_SW_MODE_LDO__IPEAK, /**< [BYPASS, IPEAK (default), NPM] -> max
aggregation (left to right). */
   PM_NPA_PIN_CONTROL_ENABLE__NONE, /**< [NONE, EN1, EN2, EN3, EN4] -> ORed
value of list. */
   PM_NPA_PIN_CONTROL_POWER_MODE__NONE, /**< [NONE, EN1, EN2, EN3, EN4,
SLEEPB] -> ORed value of list. */
   2, /**< Perpherial Index */
   0, /**< Primary (0) or Secondary (1) PMIC */
   0, /**< If((old sw_en == disable) && (new sw_en == enable) || new pc_en
== enable then ldo_en_trans = true else ldo_en_trans = false */
   PM_NPA_BYPASS_DISALLOWED, /**< [Allowed (default), Disallowed]*/
   0, /**< reserve 1 - for 32 bit boundary */
```

```
    0, /**< [X uV] -> max aggregation. */
    0, /**< [X mA] -> summed aggregation. */
    0, /**< [X uV] -> voltage headroom needed. */
    0, /**< [X uV] -> max aggregation. */
    //PM_NPA_CORNER_MODE__NONE, /**< [None, Level1 (Retention), Level2, 
Level3, Level4, Level5, Level6 (SuperTurbo), Not Used] */
  },
  /**< Mode: PMIC_NPA_MODE_ID_DDR_LOW_SPEED*/
  {
    PM_NPA_GENERIC_ENABLE, /**< [Disable (default), Enable] -> max 
aggregation (left to right). */
    PM_NPA_SW_MODE_LDO__NPM, /**< [BYPASS, IPEAK (default), NPM] -> max 
aggregation (left to right). */
    PM_NPA_PIN_CONTROL_ENABLE__NONE, /**< [NONE, EN1, EN2, EN3, EN4] -> ORed 
value of list. */
    PM_NPA_PIN_CONTROL_POWER_MODE__NONE, /**< [NONE, EN1, EN2, EN3, EN4, 
SLEEPB] -> ORed value of list. */
    2, /**< Perpherial Index */
    0, /**< Primary (0) or Secondary (1) PMIC */
    0, /**< If((old sw_en == disable) && (new sw_en == enable) || new pc_en 
== enable then ldo_en_trans = true else ldo_en_trans = false */
    PM_NPA_BYPASS_DISALLOWED, /**< [Allowed (default), Disallowed]*/
    0, /**< reserve 1 - for 32 bit boundary */
    0, /**< [X uV] -> max aggregation. */
    300, /**< [X mA] -> summed aggregation. */
    +100, /**< [X uV] -> voltage headroom needed. */
    +1250000, /**< [X uV] -> max aggregation. */
    //PM_NPA_CORNER_MODE__NONE, /**< [None, Level1 (Retention), Level2, 
Level3, Level4, Level5, Level6 (SuperTurbo), Not Used] */
 },
  /**< Mode: PMIC_NPA_MODE_ID_DDR_MID_SPEED*/
  {
    PM_NPA_GENERIC_ENABLE, /**< [Disable (default), Enable] -> max 
aggregation (left to right). */
    PM_NPA_SW_MODE_LDO__NPM, /**< [BYPASS, IPEAK (default), NPM] -> max 
aggregation (left to right). */
    PM_NPA_PIN_CONTROL_ENABLE__NONE, /**< [NONE, EN1, EN2, EN3, EN4] -> ORed 
value of list. */
    PM_NPA_PIN_CONTROL_POWER_MODE__NONE, /**< [NONE, EN1, EN2, EN3, EN4, 
SLEEPB] -> ORed value of list. */
    2, /**< Perpherial Index */
    0, /**< Primary (0) or Secondary (1) PMIC */
    0, /**< If((old sw_en == disable) && (new sw_en == enable) || new pc_en 
== enable then ldo_en_trans = true else ldo_en_trans = false */
    PM_NPA_BYPASS_DISALLOWED, /**< [Allowed (default), Disallowed]*/
    0, /**< reserve 1 - for 32 bit boundary */
    0, /**< [X uV] -> max aggregation. */
    400, /**< [X mA] -> summed aggregation. */
```

```
      +100, /**< [X uV] -> voltage headroom needed. */
      +1250000, /**< [X uV] -> max aggregation. */
      //PM_NPA_CORNER_MODE__NONE, /**< [None, Level1 (Retention), Level2,
   Level3, Level4, Level5, Level6 (SuperTurbo), Not Used] */
     },
     /**< Mode: PMIC_NPA_MODE_ID_DDR_HIGH_SPEED*/
     {
      PM_NPA_GENERIC_ENABLE, /**< [Disable (default), Enable] -> max
   aggregation (left to right). */
      PM_NPA_SW_MODE_LDO__NPM, /**< [BYPASS, IPEAK (default), NPM] -> max
   aggregation (left to right). */
      PM_NPA_PIN_CONTROL_ENABLE__NONE, /**< [NONE, EN1, EN2, EN3, EN4] -> ORed
   value of list. */
      PM_NPA_PIN_CONTROL_POWER_MODE__NONE, /**< [NONE, EN1, EN2, EN3, EN4,
   SLEEPB] -> ORed value of list. */
      2, /**< Perpherial Index */
      0, /**< Primary (0) or Secondary (1) PMIC */
      0, /**< If((old sw_en == disable) && (new sw_en == enable) || new pc_en
   == enable then ldo_en_trans = true else ldo_en_trans = false */
      PM_NPA_BYPASS_DISALLOWED, /**< [Allowed (default), Disallowed]*/
      0, /**< reserve 1 - for 32 bit boundary */
      0, /**< [X uV] -> max aggregation. */
      700, /**< [X mA] -> summed aggregation. */
      +100, /**< [X uV] -> voltage headroom needed. */
      +1250000, /**< [X uV] -> max aggregation. */
      //PM_NPA_CORNER_MODE__NONE, /**< [None, Level1 (Retention), Level2,
   Level3, Level4, Level5, Level6 (SuperTurbo), Not Used] */
     },
   };
```

4.3.1.2 MSM8939 and MSM8916

4.3.1.2.1 SBL changes

```
\boot_images\core\systemdrivers\pmic\config\msm8936\pm_config_target_sbl_seque
nce.c
pm_sbl_seq [ ] =
{
-----------
-----------
// MODE - LDOs_CONFIG: 8
//sid data base_addr offset reg_op rev_id_op rev_id
{ 1, 0x80, 0x4000, 0x048, PM_SBL_WRITE, EQUAL, REV_ID_COMMON}, // 1
----------
---------
{ 1, 0x80, 0x4100, 0x046, PM_SBL_WRITE, EQUAL, REV_ID_COMMON}, // 31
{ 1, 0x46, 0x4100, 0x041, PM_SBL_WRITE, EQUAL, REV_ID_COMMON}, // Add this
line to set LDO2 to 1.25V
```

4.3.1.2.2 RPM changes

```
/rpm_proc/core/power/mpm/hal/source/8936/HALmpmVDDCommands.c
- static uint32 s3_offMicrovolts=1225000;
+ static uint32 s3_offMicrovolts=1275000;

/rpm_proc/core/systemdrivers/pmic/config/msm8936/pm_config_target.c

pm_rpm_ldo_rail_info_type ldo_rail_a[NUM_OF_LDO_A] =
{
- {5, 63, 0, PM_ACCESS_ALLOWED, PM_ALWAYS_ON, PM_NPA_SW_MODE_LDO__IPEAK,
PM_NPA_BYPASS_DISALLOWED, 1200, 1300}, // LDO2 N600_Stepper
+ {5, 63, 0, PM_ACCESS_DISALLOWED, PM_ALWAYS_ON, PM_NPA_SW_MODE_LDO__IPEAK,
PM_NPA_BYPASS_DISALLOWED, 1200, 1300}, // LDO2 N600_Stepper
```

4.3.1.3 MSM8953

For MSM8953 chipset, S3 is used to power DDR VDD2

```
rpm.bf/2.4/core/systemdrivers/pmic/config/msm8953/pm_config_target.c
pm_rpm_smps_rail_info_type smps_rail_a[] ={
{300, 0, PM_ACCESS_ALLOWED,  PM_ALWAYS_ON, PM_NPA_SW_MODE_SMPS__AUTO,
PM_CLK_1p6_MHz, PM_CLK_1p6_MHz, PM_DROOP_DETECT_DIS, 1225, 1225, 0,
PM_SETTLING_ERR_EN, PM_SETTLING_EN, 0}, // HFS 2 - DDR and low sub-reg
```

4.3.1.4 MSM8937 and MSM8940

4.3.1.4.1 SBL changes

/core/systemdrivers/pmic/target/msm8937_pm8937_pmi8937/system/src/pm_sbl_boot_oem.c

```
pm_err_flag_type
pm_device_post_init(void)
{
........

    for (pmic_index = 0; pmic_index < PM_MAX_NUM_PMICS; pmic_index++)
    {
        /* Write PBS ROM ID in REV_ID.PBS_OTP_ID (0x0154) */
        if (PM_ERR_FLAG__SUCCESS == pm_get_pbs_info(pmic_index, &pbs_info))
        {
            uint8 rom_id = (uint8)(((pbs_info.rom_version & 0x00001F00) >>
8) | ((pbs_info.rom_version & 0x00010000) >> 11) | ((pbs_info.rom_version &
0x00000003) << 6));
            err_flag |= pm_comm_write_byte(2*pmic_index, 0x0154, rom_id,
1);   /* Update REV_ID.PBS_OTP_ID */
        }
    }
+    err_flag = pm_ldo_volt_level(0, PM_LDO_2, 1250000);
    return err_flag;
}
```

4.3.1.4.2 RPM changes

core/systemdrivers/pmic/config/msm8937/pm_config_rpm_npa_pam.c

```
pm_rpm_pam_lpddr_a_ldo2[] =
       0, /**< reserve 1 - for 32 bit boundary */
       0, /**< [X uV] -> max aggregation. */
       300, /**< [X mA] -> summed aggregation. */
-      62, /**< [X uV] -> voltage headroom needed. */
-      1200000, /**< [X uV] -> max aggregation. */
+      100, /**< [X uV] -> voltage headroom needed. */
+      1250000, /**< [X uV] -> max aggregation. */
       //PM_NPA_CORNER_MODE__NONE, /**< [None, Level1 (Retention), Level2,
Level3, Level4, Level5, Level6 (SuperTurbo), Not Used] */
    },
     /**< Mode: PMIC_NPA_MODE_ID_DDR_MID_SPEED*/
pm_rpm_pam_lpddr_a_ldo2[] =
       0, /**< reserve 1 - for 32 bit boundary */
       0, /**< [X uV] -> max aggregation. */
       400, /**< [X mA] -> summed aggregation. */
-      62, /**< [X uV] -> voltage headroom needed. */
-      1225000, /**< [X uV] -> max aggregation. */
+      100, /**< [X uV] -> voltage headroom needed. */
+      1250000, /**< [X uV] -> max aggregation. */
       //PM_NPA_CORNER_MODE__NONE, /**< [None, Level1 (Retention), Level2,
Level3, Level4, Level5, Level6 (SuperTurbo), Not Used] */
    },
     /**< Mode: PMIC_NPA_MODE_ID_DDR_HIGH_SPEED*/
pm_rpm_pam_lpddr_a_ldo2[] =
       0, /**< [X uV] -> max aggregation. */
       700, /**< [X mA] -> summed aggregation. */
       100, /**< [X uV] -> voltage headroom needed. */
-      1225000, /**< [X uV] -> max aggregation. */
+      1250000, /**< [X uV] -> max aggregation. */
       //PM_NPA_CORNER_MODE__NONE, /**< [None, Level1 (Retention), Level2,
Level3, Level4, Level5, Level6 (SuperTurbo), Not Used] */
    },
 };
```

/core/systemdrivers/pmic/config/msm8937/pm_config_target.c

```
static pm_rpm_ldo_rail_info_type ldo_rail_a[] =
 {
     {5, 62.5,   0, PM_ACCESS_ALLOWED, PM_NONE,
PM_NPA_SW_MODE_LDO__NPM,   PM_NPA_BYPASS_DISALLOWED, PM_DROOP_DETECT_DIS,
1000, 1050, 0, PM_SETTLING_ERR_EN, PM_SETTLING_EN, 0}, // LDO1   ULT
N600_Stepper
-    {5, 62.5,   0, PM_ACCESS_ALLOWED, PM_ALWAYS_ON,
PM_NPA_SW_MODE_LDO__IPEAK, PM_NPA_BYPASS_DISALLOWED, PM_DROOP_DETECT_DIS,
1200, 1300, 0, PM_SETTLING_ERR_EN, PM_SETTLING_EN, 0},   // LDO2   ULT
N600_Stepper
```

```
+    {5, 62.5,   0, PM_ACCESS_ALLOWED, PM_ALWAYS_ON,
PM_NPA_SW_MODE_LDO__IPEAK, PM_NPA_BYPASS_DISALLOWED, PM_DROOP_DETECT_DIS,
1250, 1300, 0, PM_SETTLING_ERR_EN, PM_SETTLING_EN, 0},   // LDO2   ULT
N600_Stepper
     {5, 62.5,   0, PM_ACCESS_ALLOWED, PM_ALWAYS_ON,
PM_NPA_SW_MODE_LDO__IPEAK, PM_NPA_BYPASS_DISALLOWED, PM_DROOP_DETECT_DIS,
600,  1400, 0, PM_SETTLING_ERR_EN, PM_SETTLING_EN, 0},  // LDO3   ULT
N600_Stepper
     {5, 250,  0, PM_ACCESS_ALLOWED, PM_NONE,      PM_NPA_SW_MODE_LDO__NPM,
PM_NPA_BYPASS_DISALLOWED, PM_DROOP_DETECT_DIS, 1800, 1850, 0,
PM_SETTLING_ERR_EN, PM_SETTLING_EN, 0},   // LDO4   LDO LV_P600
     {5, 250,  0, PM_ACCESS_ALLOWED, PM_ALWAYS_ON,
PM_NPA_SW_MODE_LDO__IPEAK, PM_NPA_BYPASS_DISALLOWED, PM_DROOP_DETECT_DIS,
1800, 1800, 0, PM_SETTLING_ERR_EN, PM_SETTLING_EN, 0},   // LDO5   LDO
LV_P300
```

4.3.2 LPDDR4

4.3.2.1 MSM8996

DDR voltage VDDA_EBI is adjusted.

1. Bump up PX1 (1.1v DDR I/O)

NOTE: The recommended setting for S12A is 1.125 V. Hence, use the given approach for test purpose. If you need more details about this, create a PMIC case.

```
boot_images_org/QcomPkg/Library/PmicLib/target/msm8996_pm8994_pmi8994/syste
m/src/pm_sbl_boot_oem.c

/* Enable SMPS-12A to PWM mode for DDR PX-1 on V2, should be ok for V1 as
well*/
err_flag |= pm_comm_write_byte(1, 0x3545, 0x80, 1);
+
+ //Set S12A to 1.15V
+ err_flag |= pm_comm_write_byte(1, 0x3541, 0xD6, 1);
if (DalPlatformInfo_IsFusion() == TRUE)
rpm_proc/core/systemdrivers/pmic/config/msm8996/pm_config_target.c
pm_rpm_smps_rail_info_type smps_rail_a[] =
{
<snip>
{500, 0, PM_ACCESS_ALLOWED, PM_NONE, PM_NPA_SW_MODE_SMPS__AUTO,
PM_CLK_3p2_MHz, PM_CLK_6p4_MHz, PM_DROOP_DETECT_DIS, 375, 1100, 0,
PM_SETTLING_ERR_EN, PM_SETTLING_EN, 0}, //FTS11
- {500, 0, PM_ACCESS_ALLOWED, PM_NONE, PM_NPA_SW_MODE_SMPS__AUTO,
PM_CLK_3p2_MHz, PM_CLK_3p2_MHz, PM_DROOP_DETECT_DIS,1125, 1125,
0,PM_SETTLING_ERR_EN, PM_SETTLING_EN, 0}, //FTS12
+ {500, 0, PM_ACCESS_ALLOWED, PM_NONE, PM_NPA_SW_MODE_SMPS__AUTO,
PM_CLK_3p2_MHz, PM_CLK_3p2_MHz, PM_DROOP_DETECT_DIS,1125, 1155,
0,PM_SETTLING_ERR_EN, PM_SETTLING_EN, 0}, //FTS12
```

```
};
rpm_proc/core/systemdrivers/pmic/config/msm8996/pm_config_rpm_npa_pam.c
static pm_npa_smps_int_rep
pm_rpm_pam_ddr_smps12 [] =
<snip>
{
/**< Mode: PMIC_NPA_MODE_ID_DDR_CFG_0 */
/**< Comments: No Activity */
{
<snip>
0, /**< [X uV] -> voltage headroom needed. */
- 1125000, /**< [X uV] -> max aggregation. */
+ 1155000, /**< [X uV] -> max aggregation. */
0, /**< [X mA] -> summed aggregation. */
PM_SWITCHING_FREQ_FREQ_NONE, /**< [xx MHz] -> max within a priority group.
*/
PM_NPA_FREQ_REASON_NONE, /**< Freq4 BT -> Freq4 GPS -> Freq4 WLAN -> Freq 4
WAN (lowest to highest priority). */
PM_NPA_QUIET_MODE__DISABLE, /**< [None, Quiet, Super Quiet] -> max
aggregation (left to right). */
PM_NPA_BYPASS_ALLOWED, /**< [Allowed (default), Disallowed] */
0, /**< reserve 1 - for 32 bit boundary */
},
/**< Mode: PMIC_NPA_MODE_ID_DDR_CFG_1 */
/**< Comments: <=150 */
{
<snip>
0, /**< [X uV] -> voltage headroom needed. */
- 1125000, /**< [X uV] -> max aggregation. */
+ 1155000, /**< [X uV] -> max aggregation. */
0, /**< [X mA] -> summed aggregation. */
PM_SWITCHING_FREQ_FREQ_NONE, /**< [xx MHz] -> max within a priority group.
*/
PM_NPA_FREQ_REASON_NONE, /**< Freq4 BT -> Freq4 GPS -> Freq4 WLAN -> Freq 4
WAN (lowest to highest priority). */
PM_NPA_QUIET_MODE__DISABLE, /**< [None, Quiet, Super Quiet] -> max
aggregation (left to right). */
PM_NPA_BYPASS_ALLOWED, /**< [Allowed (default), Disallowed] */
0, /**< reserve 1 - for 32 bit boundary */
},
/**< Mode: PMIC_NPA_MODE_ID_DDR_CFG_2 */
/**< Comments: 151 - 200 */
{
<snip>
0, /**< [X uV] -> voltage headroom needed. */
- 1125000, /**< [X uV] -> max aggregation. */
+ 1155000, /**< [X uV] -> max aggregation. */
0, /**< [X mA] -> summed aggregation. */
PM_SWITCHING_FREQ_FREQ_NONE, /**< [xx MHz] -> max within a priority group.
*/
PM_NPA_FREQ_REASON_NONE, /**< Freq4 BT -> Freq4 GPS -> Freq4 WLAN -> Freq 4
WAN (lowest to highest priority). */
PM_NPA_QUIET_MODE__DISABLE, /**< [None, Quiet, Super Quiet] -> max
aggregation (left to right). */
PM_NPA_BYPASS_ALLOWED, /**< [Allowed (default), Disallowed] */
0, /**< reserve 1 - for 32 bit boundary */
},
<snip>
```

2. Disable CPR on VDDA_EBI (controller PHY)

```
\boot_images\\Msm8996Pkg\Library\CPRTargetLib\cpr_enablement_bsp.c

static const cpr_enablement_versioned_rail_config_t
vdda_ebi_8996_versioned_cpr_enablement_2 =
{
.hw_versions =
{
.foundry_range = (const cpr_config_foundry_range[])
{
// Foundry Chip Min Rev Chip Max Rev CPR Min Rev CPR Max Rev
{CPR_FOUNDRY_SS, CPR_CHIPINFO_VERSION(3,0), CPR_CHIPINFO_VERSION(0xFF,
0xFF), 0, 255},
},
.foundry_range_count = 1,
},
- .enablement_init_params = &CPR_ENABLE_OPEN_HIGH_VOLTAGE_PARTS,
+ .enablement_init_params = &CPR_ENABLE_GLOBAL_CEILING_VOLTAGE,
```

3. Vote for higher VDDA_EBI for lower DDR frequency

VDDA_EBI voting for higher SVS for all lower DDR frequencies is as follows:

```
\boot_images\\Msm8996Pkg\Library\\ClockTargetLib\ClockBIMC.c

RPMClockMuxConfigType DDRClockConfig_V2[] =
{
/*
============================================================================
============================================================================
===============
** {freq_Hz, {source, DDRCCPLL_idx:DDR_div2x
DDRCC_mode:GPLL23_idx:APSS_div2x, GFX_div2x:MPSS_div2x, VDDA_EBI_vote},
VDDCX_vote }
**
============================================================================
============================================================================
==============*/
{ 100000000, { HAL_CLK_SOURCE_GPLL0_DIV2, HAL_CLK_BIMC_DIV2X_VAL(0, 0, 6),
HAL_CLK_BIMC_M_VAL(0, 16, 12), HAL_CLK_BIMC_N_VAL(24, 12),
CLOCK_VREG_LEVEL_NOMINAL }, CLOCK_VREG_LEVEL_LOW_MINUS },
{ 150000000, { HAL_CLK_SOURCE_GPLL0_DIV2, HAL_CLK_BIMC_DIV2X_VAL(0, 1, 4),
HAL_CLK_BIMC_M_VAL(0, 12, 8), HAL_CLK_BIMC_N_VAL(16, 8),
CLOCK_VREG_LEVEL_NOMINAL }, CLOCK_VREG_LEVEL_LOW_MINUS },
{ 200000000, { HAL_CLK_SOURCE_GPLL2, HAL_CLK_BIMC_DIV2X_VAL(0, 2, 2),
HAL_CLK_BIMC_M_VAL(0, 6, 4), HAL_CLK_BIMC_N_VAL( 8, 4),
CLOCK_VREG_LEVEL_NOMINAL }, CLOCK_VREG_LEVEL_LOW_MINUS },
{ 300000000, { HAL_CLK_SOURCE_GPLL0, HAL_CLK_BIMC_DIV2X_VAL(0, 3, 4),
HAL_CLK_BIMC_M_VAL(0, 12, 8), HAL_CLK_BIMC_N_VAL(16, 8),
CLOCK_VREG_LEVEL_NOMINAL }, CLOCK_VREG_LEVEL_LOW },

// DDRCC mode levels, new
{ 412800000, { HAL_CLK_SOURCE_GPLL2, HAL_CLK_BIMC_DIV2X_VAL(1, 4, 16),
HAL_CLK_BIMC_M_VAL(3, 8, 6), HAL_CLK_BIMC_N_VAL(12, 12),
CLOCK_VREG_LEVEL_NOMINAL }, CLOCK_VREG_LEVEL_LOW },
{ 547200000, { HAL_CLK_SOURCE_GPLL2, HAL_CLK_BIMC_DIV2X_VAL(1, 5, 18),
HAL_CLK_BIMC_M_VAL(3, 8, 6), HAL_CLK_BIMC_N_VAL(12, 12),
CLOCK_VREG_LEVEL_NOMINAL }, CLOCK_VREG_LEVEL_LOW },
{ 681600000, { HAL_CLK_SOURCE_GPLL2, HAL_CLK_BIMC_DIV2X_VAL(1, 6, 20),
HAL_CLK_BIMC_M_VAL(3, 8, 6), HAL_CLK_BIMC_N_VAL(12, 12),
CLOCK_VREG_LEVEL_NOMINAL }, CLOCK_VREG_LEVEL_LOW },
<snip>
```

4. Disable VDDA_EBI CPR only for turbo

```
boot_images\QcomPkg\Msm8996Pkg\Library\CPRTargetLib\cpr_voltage_ranges_bsp.
c
  .voltage_level = (const cpr_config_voltage_level_t[])
  {
    //Mode,            Ceiling, Floor, Fuse-Ref, max_floor_to_ceil,
ceiling_correction, floor_correction,  volt_fuse_type,  offset_fuse_type
    {CPR_VOLTAGE_MODE_SVS2,     725000, 675000,  725000,         0,       -
25000,         0,  CPR_FUSE_SVS2,   CPR_FUSE_NO_FUSE},
    {CPR_VOLTAGE_MODE_SVS,     850000, 725000,  850000,         0,       -
25000,         0,  CPR_FUSE_SVS,   CPR_FUSE_NO_FUSE},
    {CPR_VOLTAGE_MODE_NOMINAL,   900000, 750000,  900000,         0,
-25000,         0,  CPR_FUSE_NOMINAL, CPR_FUSE_NO_FUSE},
-    {CPR_VOLTAGE_MODE_SUPER_TURBO, 1015000, 875000, 1015000,         0,
-55000,         0,  CPR_FUSE_TURBO,  CPR_FUSE_NO_FUSE},
+   {CPR_VOLTAGE_MODE_SUPER_TURBO, 1015000, 1015000, 1015000,         0,
-55000,         0,  CPR_FUSE_TURBO,  CPR_FUSE_NO_FUSE},
  },
  .voltage_level = (const cpr_config_voltage_level_t[])
  {
    //Mode,            Ceiling, Floor, Fuse-Ref, max_floor_to_ceil,
ceiling_correction, floor_correction,  volt_fuse_type,  offset_fuse_type
    {CPR_VOLTAGE_MODE_SVS2,     725000, 675000,  725000,         0,       -
25000,         0,  CPR_FUSE_SVS2,   CPR_FUSE_NO_FUSE},
    {CPR_VOLTAGE_MODE_SVS,     850000, 725000,  850000,         0,       -
25000,         0,  CPR_FUSE_SVS,   CPR_FUSE_NO_FUSE},
    {CPR_VOLTAGE_MODE_NOMINAL,   900000, 750000,  900000,         0,
0,         0,  CPR_FUSE_NOMINAL, CPR_FUSE_NO_FUSE},
-    {CPR_VOLTAGE_MODE_SUPER_TURBO, 1015000, 875000, 1015000,         0,
-25000,         0,  CPR_FUSE_TURBO,  CPR_FUSE_NO_FUSE},
+    {CPR_VOLTAGE_MODE_SUPER_TURBO, 1015000, 1015000, 1015000,         0,
-25000,         0,  CPR_FUSE_TURBO,  CPR_FUSE_NO_FUSE},
},
```

5. Vote CLOCK_VREG_LEVEL_HIGH restore from recovery

```
rpm_proc/core/systemdrivers/clock/hw/msm8996/ClockRPMBIMC.c
```

```
static uint32 Clock_BIMCRestore( BIMCResourceType *pBimcRes, uint32
nFreqKHz )
{
 boolean       bWaitForClkOn;
 uint32       nBimcMinFreqKHz = MIN_FREQ_KHZ;
 uint32       nNewFreqKHz;
 ClockResourceType  *pBimcClkRes = &pBimcRes->sClockRes;
 ClockNodeType    **apClocks  = pBimcClkRes->apClocks;

 // Only enter this state when BIMC is in BIMC_COLLAPSED. Abort otherwise
 assert ( pBimcRes->eBIMCState == BIMC_COLLAPSED );

 // Turn on VDDA_EBI voltage
- Clock_BIMCSetVDDAVoltage ( CLOCK_VREG_LEVEL_LOW );
+ Clock_BIMCSetVDDAVoltage ( CLOCK_VREG_LEVEL_HIGH ); //or try
CLOCK_VREG_LEVEL_HIGH one by one
```

6. Add 100 mV to VDDA_EBI

```
boot_images/QcomPkg/Msm8996Pkg/Library/CPRTargetLib/cpr_enablement_bsp.c
////////////////////////////////////////////////
// Vdda_Ebi config
////////////////////////////////////////////////
static const cpr_enablement_versioned_rail_config_t
vdda_ebi_8996_versioned_cpr_enablement_1 =
{
  .hw_versions =
  {
    .foundry_range = (const cpr_config_foundry_range[])
    {
      // Foundry    Chip Min Rev        Chip Max Rev        CPR Min Rev CPR
Max Rev
      {CPR_FOUNDRY_SS, CPR_CHIPINFO_VERSION(0,0),
CPR_CHIPINFO_VERSION(3,0), 0,      0xFF},
    },
    .foundry_range_count = 1,
  },
  .enablement_init_params = &CPR_ENABLE_GLOBAL_CEILING_VOLTAGE,
  .supported_level = (const cpr_enablement_supported_level_t[])
  {
    //Mode            Static-Margin (mV)
-    {CPR_VOLTAGE_MODE_SVS2,     0,         NULL,         1},
-    {CPR_VOLTAGE_MODE_SVS,     0,         NULL,         1},
-    {CPR_VOLTAGE_MODE_NOMINAL,   0,         NULL,         1},
-    {CPR_VOLTAGE_MODE_SUPER_TURBO, 0,         NULL,         1},
+    {CPR_VOLTAGE_MODE_SVS2,     0+100,         NULL,         1},
+    {CPR_VOLTAGE_MODE_SVS,     0+100,         NULL,         1},
+    {CPR_VOLTAGE_MODE_NOMINAL,   0+100,         NULL,         1},
+    {CPR_VOLTAGE_MODE_SUPER_TURBO, 0+100,         NULL,         1},
  },
  .supported_level_count = 4,
};

static const cpr_enablement_versioned_rail_config_t
vdda_ebi_8996_versioned_cpr_enablement_2 =
{
  .hw_versions =
  {
    .foundry_range = (const cpr_config_foundry_range[])
    {
      // Foundry    Chip Min Rev        Chip Max Rev            CPR Min Rev
CPR Max Rev
      {CPR_FOUNDRY_SS, CPR_CHIPINFO_VERSION(3,0),
CPR_CHIPINFO_VERSION(0xFF, 0xFF), 0,      255},
    },
    .foundry_range_count = 1,
  },
  .enablement_init_params = &CPR_ENABLE_OPEN_HIGH_VOLTAGE_PARTS,
  .supported_level = (const cpr_enablement_supported_level_t[])
  {
    //Mode            Static-Margin (mV)
-    {CPR_VOLTAGE_MODE_SVS2,     0,         NULL,         1},
-    {CPR_VOLTAGE_MODE_SVS,     0,         NULL,         1},
-    {CPR_VOLTAGE_MODE_NOMINAL,   0,         NULL,         1},
-    {CPR_VOLTAGE_MODE_SUPER_TURBO, 0,         NULL,         1},
+    {CPR_VOLTAGE_MODE_SVS2,     0+100,         NULL,         1},
+    {CPR_VOLTAGE_MODE_SVS,     0+100,         NULL,         1},
+    {CPR_VOLTAGE_MODE_NOMINAL,   0+100,         NULL,         1},
+    {CPR_VOLTAGE_MODE_SUPER_TURBO, 0+100,         NULL,         1},
  },
  .supported_level_count = 4,
};
```

7. Set PWM mode to VDDA_EBI

```
\rpm_proc\core\boot\ddr\hw\msm8996\ddr_automode.c

  void ddr_automode_toggle(uint32 clk_speed_in_khz)
  {
   uint8 i = 0;

   if(ddr_automode_en)
   {
    while(ddr_automode_cfg[i].freq != DDR_AUTOMODE_MAX)
    {
     if(clk_speed_in_khz <= ddr_automode_cfg[i].freq)
     {
```

```
    /* Apply pmic automode setting */
-     pm_npa_rpm_smps_auto_mode_config(ddr_automode_cfg[i].pmic_enum);
+     pm_npa_rpm_smps_auto_mode_config(PMIC_NPA_MODE_ID_DDR_CFG_9;
    /* perform sanity check to ensure pmic setting is what we expected */
-     pm_npa_rpm_verify_smps_mode(ddr_automode_cfg[i].pmic_enum);
+     pm_npa_rpm_verify_smps_mode(PMIC_NPA_MODE_ID_DDR_CFG_9);

    break;
   }
   i++;
  }
 }
} /* ddr_automode_toggle */
```

8. Revert vote retention for VDDA_EBI voltage

```
===========================================================================
====
rpm_proc/core/systemdrivers/clock/hw/msm8996/ClockRPMBIMC.c

static uint32 Clock_BIMCCollapse( BIMCResourceType *pBimcRes )
{
 uint32       nNewFreqKHz = MIN_FREQ_KHZ;
 ClockResourceType  *pBimcClkRes = &pBimcRes->sClockRes;
 ClockNodeType    **apClocks  = pBimcClkRes->apClocks;

 // Only enter this state when BIMC is in BIMC_ON. Abort otherwise
 assert ( pBimcRes->eBIMCState == BIMC_ON );

 Clock_BIMCSwitchFrequency( pBimcClkRes, nNewFreqKHz );

 /* If BIMC in power collapsed mode */
 if ( pBimcRes->bPCModeEn == TRUE )
 {
  /* Put DDR to power collapse and freeze IO */
  ddr_enter_power_collapse( nNewFreqKHz );
  /* Disable BIMC power domain for BIMC collapse */
  Clock_DisablePowerDomain( pBimcRes->nBIMCPowerID );
 }
 else
 {
  /* Put DDR in self-refresh mode */
  ddr_pre_xo_shutdown( nNewFreqKHz );
 }

 /* Disable BIMC clock */
 Clock_DisableClock( (ClockIdType)apClocks[CLK_RES_BIMC_DDR_CH_IDX] );
 Clock_DisableClock( (ClockIdType)apClocks[CLK_RES_BIMC_CORE_IDX] );
```

```
        pBimcRes->eBIMCState = BIMC_COLLAPSED;

        // Vote retention for VDDA_EBI voltage
       - Clock_BIMCSetVDDAVoltage ( CLOCK_VREG_LEVEL_RETENTION );
       + // Clock_BIMCSetVDDAVoltage ( CLOCK_VREG_LEVEL_RETENTION );
        return 0;
}
```

4.3.2.2 SDM660 and SDM630

VDD1, VDD2 and VDDQ need to be adjusted.

4.4 CDC tuning

4.4.1 For platforms with DDR boot training

It is suggested not to change CDC delay values and use the trained one.

If required, tune the CDC delay for debug purpose.

4.4.1.1 High-level DDR PHY platforms

4.4.1.1.1 MSM8953, MSM8994 and MSM8996

Refer to functions in XBL/SBL codes

```
DDR_PHY_hal_cfg_cdcext_slave_wr()
DDR_PHY_hal_cfg_cdcext_slave_rd()
DDR_PHY_hal_cfg_cdc_slave_wrlvl()
```

Change the CDC delay registers for test purpose

Register references

For example, MSM8953 chipset

```
[PRFS(Performance) bands vs. Freq]
boot_images\core\boot\ddr\hw\hw_sequence\8953\target\8953\header\ddr_phy_te
chnology.h
#define F_RANGE_0 250000
#define F_RANGE_1 450000
#define F_RANGE_2 790000
#define F_RANGE_3 900000
#define F_RANGE_4 1066000
#define F_RANGE_5 1066000
#define F_RANGE_6 1066000
#define F_RANGE_7 1066000

Note : Only 0 to 4 PRFS band is supported for MSM8953.
e.g. F_RANGE_0 covers "freq. <= 250000"

[Register names for PRFS [0:7] CDC Delay Setting]
DDRPHY_CDCEXT_WRLVL_[0:7]_CTL_CFG
```

```
DDRPHY_CDCEXT_WR_[0:7]_CTL_CFG
DDRPHY_CDCEXT_RD_[0:7]_CTL_CFG
DDRPHY_CDCEXT_RDT2_[0:7]_CTL_CFG

Definition of bits array:
Bits 29:25 : LP mode coarse delay control for rank 1
Bits 24:20 : LP mode coarse delay control for rank 0
Bits 19:15 : HP mode fine delay control for rank 1
Bits 14:10 : HP mode coarse delay control for rank 1
Bits 9:5 : HP mode fine delay control for rank 0
Bits 4:0 : HP mode coarse delay control for rank 0

HP mode coarse step : 50 ps
HP mode fine step : 8 ps
LP mode step : 100 ps

Note 1 : PRFS [0:7] is related with F_RANGE_x.
e.g. DDRPHY_CDCEXT_WRLVL_0_CTL_CFG is related with "freq. <= 250000".

Note 2 : DDRPHY_CDCEXT_RDT2_[0:7]_CTL_CFG uses fixed value and it is in
boot_images\core\boot\ddr\hw\hw_sequence\msm8953\target\msm8953\settings\dd
r_phy_config.c

Note 3 : LP mode VS HP mode
If DDRPHY_FPM_PRFS_X_PWRS_1_HI_CFG[13] is b’1, LP mode.
If DDRPHY_FPM_PRFS_X_PWRS_1_HI_CFG[14] is b’1, HP mode.
```

[CA CDC] - name : Address

```
<WRLVL for CA signal>
CH0_CA0_DDRPHY_CDCEXT_WRLVL_0_CTL_CFG : 0x00480460
CH0_CA0_DDRPHY_CDCEXT_WRLVL_1_CTL_CFG : 0x00480464
...
CH0_CA0_DDRPHY_CDCEXT_WRLVL_4_CTL_CFG : 0x00480470

CH0_CA1_DDRPHY_CDCEXT_WRLVL_0_CTL_CFG : 0x00481460
...
CH0_CA1_DDRPHY_CDCEXT_WRLVL_4_CTL_CFG : 0x00481470
<WR for CA signal>
CH0_CA0_DDRPHY_CDCEXT_WR_0_CTL_CFG : 0x00480480
...
CH0_CA0_DDRPHY_CDCEXT_WR_4_CTL_CFG : 0x00480490

CH0_CA1_DDRPHY_CDCEXT_WR_0_CTL_CFG : 0x00481480
...
CH0_CA1_DDRPHY_CDCEXT_WR_4_CTL_CFG : 0x00481490

<DDRPHY_CDCEXT_RD_x and DDRPHY_CDCEXT_RDT2_x for CA uses fixed value>
CH0_CA0_DDRPHY_CDCEXT_RD_0_CTL_CFG : 0x004804A4
...
CH0_CA0_DDRPHY_CDCEXT_RD_4_CTL_CFG : 0x004804B4
```

```
CH0_CA1_DDRPHY_CDCEXT_RD_0_CTL_CFG : 0x004814A4
...
CH0_CA1_DDRPHY_CDCEXT_RD_4_CTL_CFG : 0x004814B4

CH0_CA0_DDRPHY_CDCEXT_RDT2_0_CTL_CFG : 0x004804E4
...
CH0_CA0_DDRPHY_CDCEXT_RDT2_4_CTL_CFG : 0x004804F4

CH0_CA1_DDRPHY_CDCEXT_RDT2_0_CTL_CFG : 0x004814E4
...
CH0_CA1_DDRPHY_CDCEXT_RDT2_4_CTL_CFG : 0x004814F4
[DQ CDC] - name : address
<WRLVL for DQ signal>
CH0_DQ0_DDRPHY_CDCEXT_WRLVL_0_CTL_CFG : 0x00482460
CH0_DQ0_DDRPHY_CDCEXT_WRLVL_1_CTL_CFG : 0x00482464
...
CH0_DQ0_DDRPHY_CDCEXT_WRLVL_4_CTL_CFG : 0x00482470

CH0_DQ1_DDRPHY_CDCEXT_WRLVL_0_CTL_CFG : 0x00483460
...
CH0_DQ1_DDRPHY_CDCEXT_WRLVL_4_CTL_CFG : 0x00483470

CH0_DQ2_DDRPHY_CDCEXT_WRLVL_0_CTL_CFG : 0x00484460
...
CH0_DQ2_DDRPHY_CDCEXT_WRLVL_4_CTL_CFG : 0x00484470

CH0_DQ3_DDRPHY_CDCEXT_WRLVL_0_CTL_CFG : 0x00485460
...
CH0_DQ3_DDRPHY_CDCEXT_WRLVL_4_CTL_CFG : 0x00485470
<WR for DQ signal>
CH0_DQ0_DDRPHY_CDCEXT_WR_0_CTL_CFG : 0x00482480
...
CH0_DQ0_DDRPHY_CDCEXT_WR_4_CTL_CFG : 0x00482490
CH0_DQ1_DDRPHY_CDCEXT_WR_0_CTL_CFG : 0x00483480
...
CH0_DQ1_DDRPHY_CDCEXT_WR_4_CTL_CFG : 0x00483490

CH0_DQ2_DDRPHY_CDCEXT_WR_0_CTL_CFG : 0x00484480
...
CH0_DQ2_DDRPHY_CDCEXT_WR_4_CTL_CFG : 0x00484490

CH0_DQ3_DDRPHY_CDCEXT_WR_0_CTL_CFG : 0x00485480
...
CH0_DQ3_DDRPHY_CDCEXT_WR_4_CTL_CFG : 0x00485490
<RD for DQ signal>
CH0_DQ0_DDRPHY_CDCEXT_RD_0_CTL_CFG : 0x004824A4
...
CH0_DQ0_DDRPHY_CDCEXT_RD_4_CTL_CFG : 0x004824B4

CH0_DQ1_DDRPHY_CDCEXT_RD_0_CTL_CFG : 0x004834A4
...
```

```
CH0_DQ1_DDRPHY_CDCEXT_RD_4_CTL_CFG : 0x004834B4

CH0_DQ2_DDRPHY_CDCEXT_RD_0_CTL_CFG : 0x004844A4
...
CH0_DQ2_DDRPHY_CDCEXT_RD_4_CTL_CFG : 0x004844B4

CH0_DQ3_DDRPHY_CDCEXT_RD_0_CTL_CFG : 0x004854A4
...
CH0_DQ3_DDRPHY_CDCEXT_RD_4_CTL_CFG : 0x004854B4
<DDRPHY_CDCEXT_RDT2_x for DQ uses fixed value>
CH0_DQ0_DDRPHY_CDCEXT_RDT2_0_CTL_CFG : 0x004824E4
...
CH0_DQ0_DDRPHY_CDCEXT_RDT2_4_CTL_CFG : 0x004824F4
CH0_DQ1_DDRPHY_CDCEXT_RDT2_0_CTL_CFG : 0x004834E4
...
CH0_DQ1_DDRPHY_CDCEXT_RDT2_4_CTL_CFG : 0x004834F4
CH0_DQ2_DDRPHY_CDCEXT_RDT2_0_CTL_CFG : 0x004844E4
...
CH0_DQ2_DDRPHY_CDCEXT_RDT2_4_CTL_CFG : 0x004844F4

CH0_DQ3_DDRPHY_CDCEXT_RDT2_0_CTL_CFG : 0x004854E4
...
CH0_DQ3_DDRPHY_CDCEXT_RDT2_4_CTL_CFG : 0x004854F4
```

4.4.1.1.2 MSM8974

All registers:

```
DQ WR CDC Delay (two channels, 4 bytes):
0xFC4E0834 CH0_DQ0_DDRPHY_DQ_WR_CDC_DELAY0
0xFC4E1034 CH0_DQ1_DDRPHY_DQ_WR_CDC_DELAY0
0xFC4E1834 CH0_DQ2_DDRPHY_DQ_WR_CDC_DELAY0
0xFC4E2034 CH0_DQ3_DDRPHY_DQ_WR_CDC_DELAY0

0xFC4E5834 CH1_DQ0_DDRPHY_DQ_WR_CDC_DELAY0
0xFC4E6034 CH1_DQ1_DDRPHY_DQ_WR_CDC_DELAY0
0xFC4E6834 CH1_DQ2_DDRPHY_DQ_WR_CDC_DELAY0
0xFC4E7034 CH1_DQ3_DDRPHY_DQ_WR_CDC_DELAY0
DQ RD CDC Delay (two channels, 4 bytes):
0xFC4E0858 CH0_DQ0_DDRPHY_DQ_RD_CDC_DELAY0
0xFC4E1058 CH0_DQ1_DDRPHY_DQ_RD_CDC_DELAY0
0xFC4E1858 CH0_DQ2_DDRPHY_DQ_RD_CDC_DELAY0
0xFC4E2058 CH0_DQ3_DDRPHY_DQ_RD_CDC_DELAY0
0xFC4E5858 CH1_DQ0_DDRPHY_DQ_RD_CDC_DELAY0
0xFC4E6058 CH1_DQ1_DDRPHY_DQ_RD_CDC_DELAY0
0xFC4E6858 CH1_DQ2_DDRPHY_DQ_RD_CDC_DELAY0
0xFC4E7058 CH1_DQ3_DDRPHY_DQ_RD_CDC_DELAY0
```

4.4.1.2 Middle level DDR PHY platforms

4.4.1.2.1 MSM8939, MSM8952, and MSM8976

For example, MSM8952 chipset

Registers:

```
CH0_CA_DDRPHY_CA_CDC_WR90_DELAY0 00480034
CH0_DQ0_DDRPHY_DQ_CDC_WR90_DELAY0 0x00480848
CH0_DQ1_DDRPHY_DQ_CDC_WR90_DELAY0 0x00481048
CH0_DQ0_DDRPHY_DQ_CDC_RD90_DELAY0 0x00480858
CH0_DQ1_DDRPHY_DQ_CDC_RD90_DELAY0 0x00481058

The register definition:
bit[6:0]: BYTE0_CDC_WR90/RD90_DELAY_VALUE
bit[14: 8]: BYTE1_CDC_WR90/RD90_DELAY_VALUE
```

Software changes:

Besides changing the CDC registers directly, adjust the fine_lut[7] and re-train the DDR by erase DDR partition, which will impact the cdc delay for WR/RD/CA

In boot images:

151 value is just an example. You can change to others and verify by stress tests and SI/PI measurement of the vendor.

```
ddr_phy_config_8936.h  static const uint16 fine_lut[8] = {0, 12, 26, 39,
53, 63, 74, 89=>151};
ddr_phy_config_8952.h  static const uint16 fine_lut[8] = {0, 12, 26, 39,
53, 63, 74, 89=>151};
ddr_phy_config_8937.h  static const uint16 fine_lut[8] = {0, 12, 26, 39,
53, 63, 74, 89=>151};
ddr_phy_config_8956.h  static const uint16 fine_lut[8] = {0, 12, 26, 39,
53, 63, 74, 89=>151};
```

4.4.1.2.2 MSM8937 and MSM8940

```
<CH0> - MSM8937 has only CH0
CA CDC(T/4)
"834.029MHz > Freq. >= 555.865MHz" -
CH0_CA_DDRPHY_CA_CDC_WR90_DELAY1(0x00480038)
"555.865MHz > Freq. >= 400.321MHz" -
CH0_CA_DDRPHY_CA_CDC_WR90_DELAY2(0x0048003C)
"400.321MHz > Freq." - CH0_CA_DDRPHY_CA_CDC_WR90_DELAY3(0x00480040)
: Bits[6:3] coarse delay(73 ps), Bits[2:0] fine delay(about 12 ps)

DQ WD CDC(T/4)
"834.029MHz > Freq. >= 555.865MHz" -
CH0_DQ0_DDRPHY_DQ_CDC_WR90_DELAY1(0x0048084C)
"555.865MHz > Freq. >= 400.321MHz" -
CH0_DQ0_DDRPHY_DQ_CDC_WR90_DELAY2(0x00480850)
"400.321MHz > Freq." - CH0_DQ0_DDRPHY_DQ_CDC_WR90_DELAY3(0x00480854)
```

```
: Bits[14:11] coarse delay(73 ps) for byte 1(in Phy), Bits[10:8] fine
delay(about 12 ps) for byte 1, Bits[6:3] coarse delay(73 ps) for byte 0,
Bits[2:0] fine delay(about 12 ps) for byte 0

DQ RD CDC(T/4)
"834.029MHz > Freq. >= 555.865MHz" -
CH0_DQ0_DDRPHY_DQ_CDC_RD90_DELAY1(0x0048085C)
"555.865MHz > Freq. >= 400.321MHz" -
CH0_DQ0_DDRPHY_DQ_CDC_RD90_DELAY2(0x00480860)
"400.321MHz > Freq." - CH0_DQ0_DDRPHY_DQ_CDC_RD90_DELAY3(0x00480864)
: Bits[14:11] coarse delay(73 ps) for byte 1, Bits[10:8] fine delay(about
12 ps) for byte 1, Bits[6:3] coarse delay(73 ps) for byte 0, Bits[2:0] fine
delay(about 12 ps) for byte 0
```

Note - in case of DQ WD CDC(T/4) and DQ RD CDC(T/4), they have same register set for 2nd phy in bus.

```
CH0_DQ1_DDRPHY_DQ_CDC_WR90_DELAY1(0x0048104C)
CH0_DQ1_DDRPHY_DQ_CDC_WR90_DELAY2(0x00481050)
CH0_DQ1_DDRPHY_DQ_CDC_WR90_DELAY3(0x00481054)

CH0_DQ1_DDRPHY_DQ_CDC_RD90_DELAY1(0x0048105C)
CH0_DQ1_DDRPHY_DQ_CDC_RD90_DELAY2(0x00481060)
CH0_DQ1_DDRPHY_DQ_CDC_RD90_DELAY3(0x00481064)
```

4.4.1.2.3 MSM8917

```
CA
DIM_C00_DIM_CA_CDC_DELAY_CFG[11:0] : 0x00480050

DQ - RD
DIM_D00_DIM_DQ_RD_CDC_DELAY_CFG[11:0] : 0x00480850
DIM_D01_DIM_DQ_RD_CDC_DELAY_CFG[11:0] : 0x00481050
DIM_D02_DIM_DQ_RD_CDC_DELAY_CFG[11:0] : 0x00481850
DIM_D03_DIM_DQ_RD_CDC_DELAY_CFG[11:0] : 0x00482050

DQ - WR
DIM_D00_DIM_DQ_WR_CDC_DELAY_CFG[11:0] : 0x004808B0
DIM_D01_DIM_DQ_WR_CDC_DELAY_CFG[11:0] : 0x004810B0
DIM_D02_DIM_DQ_WR_CDC_DELAY_CFG[11:0] : 0x004818B0
DIM_D03_DIM_DQ_WR_CDC_DELAY_CFG[11:0] : 0x004820B0
```

4.4.2 For platforms without DDR boot training

4.4.2.1 MSM8x10, MSM8926, and MSM8916

Changes only for debug purpose of hardware-specific issues are used.

For example, MSM8916 chipset:

Registers:

To perform CDC sweep, refer to the following registers.

NOTE: CDC sweep during run-time makes unexpected issues and does not reflects exact value.

```
0x0048004C DIM_C00_DIM_CA_CDC_OFFSET_CFG

0x0048084C DIM_D00_DIM_DQ_RD_CDC_OFFSET_CFG
0x0048104C DIM_D01_DIM_DQ_RD_CDC_OFFSET_CFG
0x0048184C DIM_D02_DIM_DQ_RD_CDC_OFFSET_CFG
0x0048204C DIM_D03_DIM_DQ_RD_CDC_OFFSET_CFG

0x004808AC DIM_D00_DIM_DQ_WR_CDC_OFFSET_CFG
0x004810AC DIM_D01_DIM_DQ_WR_CDC_OFFSET_CFG
0x004818B0 DIM_D02_DIM_DQ_WR_CDC_DELAY_CFG
0x004820AC DIM_D03_DIM_DQ_WR_CDC_OFFSET_CFG
```

The following bit map is helpful:

- [7] UNIT_OFFSET_SIGN
- [1'b1] Offset is subtracted from the unit step count
- [1'b0] Offset is added to the unit step count
- [5:0] UNIT_OFFSET Software unit step offset

Software changes:

```
1 Init values:
boot_images\core\boot\ddr\hw\msm8916\ddr_config.c
uint32 ddr_dqphy_config_MSM8x16_LPDDR2[][2] =
{
{HWIO_ADDR(PHY_DQ_ADDR(RD_CDC_DELAY_CFG)), 0x00001D5}

{HWIO_ADDR(PHY_DQ_ADDR(WR_CDC_DELAY_CFG)), 0x00001D5},

rpm_proc\core\boot\ddr\hw\msm8916\ddr_target.c

2 Runtime settings
rpm_proc/core/boot/ddr/hw/msm8916/ddr_target.c

/* DDR PHY dynamic settings per vddmx voltage level */
static uint32 ddr_phy_config_vddmx[RAILWAY_CORNERS_COUNT] =
{
0x271, /* RAILWAY_NO_REQUEST */
0x271, /* RAILWAY_RETENTION */
0x271, /* RAILWAY_SVS_KRAIT */
0x271, /* RAILWAY_SVS_SOC */
0x271, /* RAILWAY_NOMINAL */
0x1D5, /* RAILWAY_TURBO */
0x1D5, /* RAILWAY_TURBO_HIGH */
0x1D5, /* RAILWAY_SUPER_TURBO */
0x1D5, /* RAILWAY_SUPER_TURBO_NO_CPR */
};
Modify for - 100ps
===========================================
/* DDR PHY dynamic settings per vddmx voltage level */
static uint32 ddr_phy_config_vddmx[RAILWAY_CORNERS_COUNT] =
{
(0x271-100), /* RAILWAY_NO_REQUEST */
(0x271-100), /* RAILWAY_RETENTION */
(0x271-100), /* RAILWAY_SVS_KRAIT */
(0x271-100), /* RAILWAY_SVS_SOC */
(0x271-100), /* RAILWAY_NOMINAL */
(0x1D5-100), /* RAILWAY_TURBO */
(0x1D5-100), /* RAILWAY_TURBO_HIGH */
(0x1D5-100), /* RAILWAY_SUPER_TURBO */
(0x1D5-100), /* RAILWAY_SUPER_TURBO_NO_CPR */
};
Modify for + 100ps
===========================================
/* DDR PHY dynamic settings per vddmx voltage level */
static uint32 ddr_phy_config_vddmx[RAILWAY_CORNERS_COUNT] =
{
(0x271+100), /* RAILWAY_NO_REQUEST */
(0x271+100), /* RAILWAY_RETENTION */
(0x271+100), /* RAILWAY_SVS_KRAIT */
(0x271+100), /* RAILWAY_SVS_SOC */
(0x271+100), /* RAILWAY_NOMINAL */
(0x1D5+100), /* RAILWAY_TURBO */
(0x1D5+100), /* RAILWAY_TURBO_HIGH */
(0x1D5+100), /* RAILWAY_SUPER_TURBO */
(0x1D5+100), /* RAILWAY_SUPER_TURBO_NO_CPR */
};
```

4.5 Manual CDC margin test

See Section 4.4 for more details. The CDC delay registers address and value format for most of the QTI platforms are known. Based on that, it is possible to manually scan all the valid CDC delay register values.

For each value:

1. Perform a Memtester/QMESA for some minutes (about 10 minutes).
2. Record all the pass and fail values.
3. Fill a table to describe the “valid” window.

It is similar to the QDUTT tool, but the test time to get more precise results is controlled. For example, for the MSM8939 chipset. For other platforms, the registers are different but the basic process is the same.

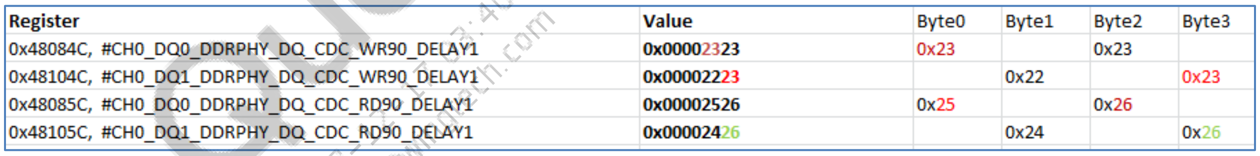
4.5.1 Setup

1. Memtester and Qmesa (both can also be used at the same time)
2. Fastboot flash TZ tzbsp_no_xpu.mbn to unblock the access of DDR registers

4.5.2 Process

1. Read all out default CDC values
   - adb shell /system/bin/r 0x0048084c
   - adb shell /system/bin/r 0x0048104c
   - adb shell /system/bin/r 0x0048085c
   - adb shell /system/bin/r 0x0048105c
2. Sweep write CDC
   a. Change testing byte write CDC to testing value
   b. Setup all other 3 bytes, write CDC to default CDC value
   c. Setup all 4 bytes, read CDC to default CDC value
   d. Execute two instance memtester or QMesa for 10 minutes
      i If no failure is reported, then it passed
      ii If failure is reported or the device crashes, then it is failed

   Refer to the MSM8939 chipset CDC register mapping (For Fmax=800 MHz)



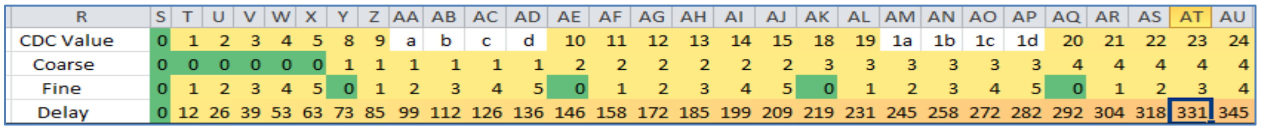
Register	Value	Byte0	Byte1	Byte2	Byte3
0x48084C, #CH0_DQ0_DDRPHY_DQ_CDC_WR90_DELAY1	**0x00002323**	0x23		0x23	
0x48104C, #CH0_DQ1_DDRPHY_DQ_CDC_WR90_DELAY1	**0x00002223**		0x22		0x23
0x48085C, #CH0_DQ0_DDRPHY_DQ_CDC_RD90_DELAY1	**0x00002526**	0x25		0x26	
0x48105C, #CH0_DQ1_DDRPHY_DQ_CDC_RD90_DELAY1	**0x00002426**		0x24		0x26

3. Sweep read CDC
   a. Change one testing byte read CDC to testing value
   b. Setup all 3 bytes, read CDC to default CDC value
   c. Setup all 4 bytes, write CDC to default CDC value
   d. Execute two instance memtester for 10 minutes
      i If no failure is reported, then it is passed
      ii If failure is reported or the device crashes, then it is failed
4. Change CDC value example

Byte#2 and Byte#0 Default 800 MHz Write CDC is 0x23, Change Byte#2 write CDC to 0x1C

adb shell /system/bin/r -s 0x0048084c 0x231C

For example, CDC Value 0x23: the actual delay from the training is 331 ps



R	S	T	U	V	W	X	Y	Z	AA	AB	AC	AD	AE	AF
CDC Value	0	1	2	3	4	5	8	9	a	b	c	d	10	11
Coarse	0	0	0	0	0	0	1	1	1	1	1	1	2	2
Fine	0	1	2	3	4	5	0	1	2	3	4	5	0	1
Delay	0	12	26	39	53	63	73	85	99	112	126	136	146	158

R	AG	AH	AI	AJ	AK	AL	AM	AN	AO	AP	AQ	AR	AS	AT	AU
CDC Value	12	13	14	15	18	19	1a	1b	1c	1d	20	21	22	23	24
Coarse	2	2	2	2	3	3	3	3	3	3	4	4	4	4	4
Fine	2	3	4	5	0	1	2	3	4	5	0	1	2	3	4
Delay	172	185	199	209	219	231	245	258	272	282	292	304	318	331	345

5. An example for the RD results

Some continuous pass/fail areas are given here:

- F-fail
- P-Pass
- D-default value or trained value

CDC Value	0x 10	0x 11	0x 12	0x 13	0x 14	0x 1D	0x 1E	0x 1F	0x 20	0x 21	0x 22	0x 30	0x 31	0x 32	0x 33	0x 34	0x 35	0x 36
RD Byte 0	F	P	P	P	P	P	P	P	P	D	P	P	P	P	P	P	P	F
RD Byte 2	F	P	P	P	P	P	D	P	P	P	P	P	P	P	F	F	F	F
RD Byte 1	F	F	P	P	P	P	D	P	P	P	P	P	P	P	P	P	P	F
RD byte 3	F	F	F	F	P	P	P	P	P	D	P	P	F	F	F	F	F	F

6. Suggestions based on results
   a. If the default values are not next to the failed area, it is fine.
   b. If the default values are next or inside the failed area, see Section 4.4 to adjust the default CDC or use fine_lut[7] to adjust the trained CDC.

4.6 Disable hardware self-refresh

Usually, this is for bitflips which are linked to DDR refresh problems with temperature.

4.6.1 High-level DDR PHY platforms

4.6.1.1 MSM8953, MSM8994 and MSM8996

- For the MSM8953 and MSM8996 chipsets using eCDT:

  Disable hardware SR:

```
---------------------
/boot_images/QcomPkg/Msm8996Pkg/Library/DSFTargetLib/target/8996/setting
s/target_config.c

void DDRSS_Create_Ecdt_Runtime_Structs(DDR_STRUCT *ddr,
EXTENDED_CDT_STRUCT *ecdt)
{
uint8 ecdt_index;
uint8 runtime_index;

if(ecdt != NULL)
{
// Create DRAM latency run-time structures.
ddr->extended_cdt_runtime.hw_self_refresh_enable = ecdt-
>hw_self_refresh_enable;
ddr->extended_cdt_runtime.MR4_polling_enable = ecdt->MR4_polling_enable;
ddr->extended_cdt_runtime.periodic_training_enable = ecdt-
>periodic_training_enable;
ddr->extended_cdt_runtime.page_close_timer = ecdt->page_close_timer;
}
else
{
ddr->extended_cdt_runtime.hw_self_refresh_enable = 1;
ddr->extended_cdt_runtime.MR4_polling_enable = 1;
ddr->extended_cdt_runtime.periodic_training_enable = 0;
ddr->extended_cdt_runtime.page_close_timer = 256;
}


+ ddr->extended_cdt_runtime.hw_self_refresh_enable = 0; //disable HW
self refresh
```

4.6.1.2 MSM8994

```
File :/boot_images/core/boot/ddr/hw/msm8994/ddr_target.c

void ddr_pre_setup_forDSF(void) {
+ /* Disable HW self Refresh */
+ddrsns_share_data->misc.misc_cfg[0] = 0 ; // disable HW self Refresh

- /* Enable HW self Refresh */
- ddrsns_share_data->misc.misc_cfg[0] = 1 ; // by default enable HW self
Refresh
/* Force Apply the HW self Refresh to SHKE registers */
ddr_enter_self_refresh(SDRAM_INTERFACE_BOTH, SDRAM_BOTH, 200000);
ddr_exit_self_refresh(SDRAM_INTERFACE_BOTH, SDRAM_BOTH, 200000);
}
```

4.6.1.3 SDM660 and SDM630

```
 boot_images\QcomPkg\Sdm660Pkg\Library\DSFTargetLib\rpm\ddrss\bimc\mc230\src
\bimc_common_rpm.c

//================================================//
// Enable/Disable HW activity based self refresh
//================================================//
void BIMC_HW_Self_Refresh_Ctrl (DDR_STRUCT *ddr, uint8 ch, uint8 cs, uint8
enable)
{

+enable = 0;
   // Check if this feature is enabled in the DDR_STRUCT
   if (ddr->extended_cdt_runtime.hw_self_refresh_enable[ch] == 1) {
      if (cs == 0) {
         HWIO_OUTXF (REG_OFFSET_SHKE(ch), SHKE_SELF_REFRESH_CNTL,
HW_SELF_RFSH_ENABLE_RANK0, enable);
```

```
            }
            if (cs == 1) {
               HWIO_OUTXF (REG_OFFSET_SHKE(ch), SHKE_SELF_REFRESH_CNTL,
      HW_SELF_RFSH_ENABLE_RANK1, enable);
            }
         }
      }
```

4.6.1.4 MSM8998

```
       boot_images\MSM8998Pkg\Sdm660Pkg\Library\DSFTargetLib\rpm\ddrss\bimc\mc230\
      src\bimc_common_rpm.c
       //==============================================//
      // Enable/Disable HW activity based self refresh
      //==============================================//
      void BIMC_HW_Self_Refresh_Ctrl (DDR_STRUCT *ddr, uint8 ch, uint8 cs, uint8
      enable)
      {
       +enable = 0;
         // Check if this feature is enabled in the DDR_STRUCT
         if (ddr->extended_cdt_runtime.hw_self_refresh_enable[ch] == 1) {
            if (cs == 0) {
               HWIO_OUTXF (REG_OFFSET_SHKE(ch), SHKE_SELF_REFRESH_CNTL,
      HW_SELF_RFSH_ENABLE_RANK0, enable);
            }
            if (cs == 1) {
               HWIO_OUTXF (REG_OFFSET_SHKE(ch), SHKE_SELF_REFRESH_CNTL,
      HW_SELF_RFSH_ENABLE_RANK1, enable);
            }
         }
      }
```

4.6.2 Mid-level DDR PHY platforms

4.6.2.1 MSM8939, MSM8952, MSM8976, and MSM8937

```
boot_images\core\boot\ddr\hw\hw_sequence\BIMC\v2.2\bimc_mc_shke.c
void ABHN_SHKE_Enable_HW_Self_Refresh( uint32 _inst_, uint32 chip_select, uint32
concurrent_sr )
{
+#if 0
/// Concurrent self-refresh is needed for dual rank where inactive rank
/// is providing ODT.
/// Recommended for Dual rank PCDDR3 & Dual rank external LPDDR3
if (concurrent_sr) {
<snip>
if (chip_select & SDRAM_RANK_CS1) {
HWIO_OUTXF (_inst_, ABHN_SHKE_SELF_REFRESH_CNTL, HW_SELF_RFSH_ENABLE_RANK1,
HWIO_ABHN_SHKE_SELF_REFRESH_CNTL_HW_SELF_RFSH_ENABLE_RANK1_ENABLE_FVAL);
}
}
+#endif
}
<snip>

void ABHN_SHKE_Disable_HW_Self_Refresh( uint32 _inst_, uint32 chip_select )
{
+#if 0
/// disable concurrent self refresh
HWIO_OUTXF (_inst_, ABHN_SHKE_SELF_REFRESH_CNTL, CONCURRENT_SELF_RFSH_EN,
HWIO_ABHN_SHKE_SELF_REFRESH_CNTL_CONCURRENT_SELF_RFSH_EN_DISABLE_FVAL);
<snip>
if (chip_select & SDRAM_RANK_CS1) {
HWIO_OUTXF (_inst_, ABHN_SHKE_SELF_REFRESH_CNTL, HW_SELF_RFSH_ENABLE_RANK1,
HWIO_ABHN_SHKE_SELF_REFRESH_CNTL_HW_SELF_RFSH_ENABLE_RANK1_DISABLE_FVAL);
}
+#endif
}
```

that is, makes ABHN_SHKE_Enable_HW_Self_Refresh() and ABHN_SHKE_Disable_HW_Self_Refresh() to empty functions

4.7 Disable software self-refresh (Deep sleep)

It rules out crash issues that happen during sleep and wake-up. This change is for most of the platforms. For example, the MSM8996 chipset (Same with SDM660/SDM630/MSM8998)

4.7.1 Disable Low-power mode

rpm_proc/core/power/sleep/src/sleep_perform.c

```
    #ifdef MSM_DISABLE_SLEEP
    volatile boolean sleep_allow_low_power_modes = FALSE;
    #else
    - volatile boolean sleep_allow_low_power_modes = TRUE;
    + volatile boolean sleep_allow_low_power_modes = FALSE;
    #endif
```

4.7.2 Disable BIMC power collapse and Low-power mode

rpm_proc/core/systemdrivers/clock/hw/msm8996/ClockRPMBIMC.c

```
    uint32 Clock_BIMCSetFrequency
    (
     BIMCResourceType  *pBimcRes,
     uint32       nNewClkKHz
    )
    {
     uint32    nFreqKHz = 0;

     /* Switch clock if BIMC is already on, and new requesting frequency is not
    off (0) */
     if ( (Clock_Resources.BIMCResource.eBIMCState == BIMC_ON) &&
    (nNewClkKHz != 0) )
     {
      return Clock_BIMCSwitchFrequency( &pBimcRes->sClockRes, nNewClkKHz );
```

```
        }

        /* New frequency is zero, collapsing BIMC */
        if ( nNewClkKHz == 0 )
        {
       -  return Clock_BIMCCollapse( pBimcRes );
       +  ; //return Clock_BIMCCollapse( pBimcRes );
        }
        /* New frequency is non-zero, restore BIMC */
        else
        {
         nFreqKHz = Clock_BIMCRestore( pBimcRes, nNewClkKHz );
        }
        return nFreqKHz;
       }
rpm_proc/core/power/sleep/src/sleep_perform.c
       #ifdef MSM_DISABLE_SLEEP
       volatile boolean sleep_allow_low_power_modes = FALSE;
       #else
       - volatile boolean sleep_allow_low_power_modes = TRUE;
       + volatile boolean sleep_allow_low_power_modes = FALSE;
       #endif
       rpm_proc\core\power\sleep\src\Lpr_definition_uber.c
       #ifdef MSM_DISABLE_BIMC_PC
       BIMC_PC_CFG = BIMC_PC_DISABLE;
       #else
       - BIMC_PC_CFG = BIMC_PC_ENABLE;
       + BIMC_PC_CFG = BIMC_PC_DISABLE;
       #endif
```

4.7.3 Keep BIMC power domain during Low-power mode

```
       rpm_proc/core/systemdrivers/clock/hw/msm8996/ClockRPMNPA.c
        .BIMCResource =
        {
         .sClockRes      = CLK_RESOURCE_DECL(CLK_RES_BIMC_NUM_CLKS),
         .eBIMCState      = BIMC_ON,
       -  .bPCModeEn      = TRUE,
       +  .bPCModeEn      = FALSE,
          .bBIMCAppsEnabled   = FALSE,
        },
```

4.8 Increase and fix auto refresh rate tREFI

It is for bitflips that are linked to DDR refresh problems with temperature.

4.8.1 High-level DDR PHY platforms

4.8.1.1 MSM8953, MSM8994, and MSM8996

- MSM8994

Setting DISABLE_AUTO_REFRESH_TEMP_UPDATE in BIMC_S_DDR0_SHKE_AUTO_REFRESH_CNTL/BIMC_S_DDR1_SHKE_AUTO_REFRESH_CNTL to b'1 disables the temperature compensated auto refresh. Thus the refresh takes place based on the programmed tREFI regardless of the temperature variation.

```
boot_images/core/boot/ddr/hw/hw_sequence/target/8994/settings/bimc_config.c

uint32 bimc_shke_config[][2] =
{
- {HWIO_ADDRX((SEQ_BIMC_BIMC_S_DDR0_SHKE_OFFSET), SHKE_AUTO_REFRESH_CNTL),
0x00000049},
+ {HWIO_ADDRX((SEQ_BIMC_BIMC_S_DDR0_SHKE_OFFSET), SHKE_AUTO_REFRESH_CNTL),
0x00010049}, //Bit 16: DISABLE_AUTO_REFRESH_TEMP_UPDATE = 1
{HWIO_ADDRX((SEQ_BIMC_BIMC_S_DDR0_SHKE_OFFSET), SHKE_AUTO_REFRESH_CNTL_1),
0x00000049},
{HWIO_ADDRX((SEQ_BIMC_BIMC_S_DDR0_SHKE_OFFSET), SHKE_AUTO_REFRESH_CNTL_2),
0x00080000},

boot_images/core/boot/ddr/hw/hw_sequence/ddrss/bimc/mc230/src/bimc.c

// cdt_params has resolution of 100ps, multiplied by 100 to convert to 1ps
- tREFI_in_XO = (ddr->cdt_params[ch].lpddr.tREFI * 100) / XO_PERIOD_IN_PS - 1;
+ tREFI_in_XO = ( ( (ddr->cdt_params[ch].lpddr.tREFI)/4 ) * 100) / XO_PERIOD_IN_PS
- 1; //Default TREFI is 39000 in jedec_lpddr4_2gb_2ch_1rank_interleave.xml ,
change to 0.25X
HWIO_OUTXF (reg_offset_shke, SHKE_AUTO_REFRESH_CNTL, TREFI, tREFI_in_XO);
HWIO_OUTXF (reg_offset_shke, SHKE_AUTO_REFRESH_CNTL_1, TREFI, tREFI_in_XO);
```

4.8.2 Mid-level DDR PHY platforms

4.8.2.1 MSM8939, MSM8952, MSM8976, MSM8937, and MSM8974

```
1 Dummy HAL_SDRAM_SHKE_Enable_Periodic_SRR()
{
#if 0
….
#endif
}
2 change default tREFI
```

The most common way for such platforms is to change the tREFI value in CDT XML to ¼*tREFI value and re-generate or replace the boot_cdt_array.c and then rebuild the sbl image.

4.9 Thermal offset changes

It is for bitflips that are linked to DDR refresh problems with temperature. It is specific for high platforms that are using the LPDDR4.

4.9.1 MSM8994 and MSM8996

```
file: boot_images/core/boot/ddr/hw/hw_sequence/ddrss/bimc/mc230/src/bimc.c

void BIMC_Memory_Device_Init (DDR_STRUCT *ddr, DDR_CHANNEL channel,
DDR_CHIPSELECT chip_select,
uint32 clk_freq_khz)
{
uint8 ch = 0x0;
uint32 reg_offset_shke = 0;
+ uint32 temp;

<snip>

BIMC_ZQ_Calibration (ddr, channel, chip_select);

+ temp = BIMC_MR_Read(channel, chip_select, JEDEC_MR_4);
+ BIMC_MR_Write(channel, chip_select, JEDEC_MR_4, temp|0x40); //or 0x60
```

4.10 Bump up and RBCPR for VddCx/VddMx

This is a general way to improve stability issues which are linked to vddcx/vddmx. Not specific for DDR issues, but voltage like vddmx impacts DDR controller.

For more details, refer to the RPM related SOP documents or raise RPM and PMIC case to check further.

4.10.1 MSM8926

4.10.1.1 VddCx bump up

Bump up the retention voltage by 50mV.

```
/rpm_proc/core/power/sleep/src/8x26/sleep_target_config.c
static const uint32 vddcx_pvs_retention_data[8] =
{
/* 000 */ 700000+50000,
/* 001 */ 500000+50000,
/* 010 */ 500000+50000,
/* 011 */ 500000+50000,
/* 100 */ 500000+50000,
/* 101 */ 500000+50000,
/* 110 */ 500000+50000,
/* 111 */ 500000+50000
};
```

Bump up vdd_cx by 50 mv

```
\rpm_proc\core\power\railway_v2\src\8x26\railway_config.c
// VddCx
{
.vreg_name = "vddcx",
..........
.default_uvs = (const unsigned[])
{
0, // RAILWAY_NO_REQUEST
550000, // RAILWAY_RETENTION → it is not used actually.
1050000+50000, // RAILWAY_SVS_KRAIT
1050000+50000, // RAILWAY_SVS_SOC
1150000+50000, // RAILWAY_NOMINAL
1275000+50000, // RAILWAY_TURBO
1275000+50000, // RAILWAY_TURBO_HIGH
1275000+50000, // RAILWAY_SUPER_TURBO
1275000+50000, // RAILWAY_SUPER_TURBO_NO_CPR
},
….
},
```

4.10.1.2 VddMx bump up

```
bump up the vdd_mx to by 50mv
      (1) SBL change:
      the following code will change the range of LDO3(vdd_mx) in PMIC registor:
      pm_device_post_init(void)
      {
      pm_err_flag_type err = PM_ERR_FLAG__SUCCESS;

      //LDO_UL_LL_CONFIG:


      +err |= pm_spmi_lite_write_byte(1, 0x42d0, 0xA5, 0);
      + err |= pm_spmi_lite_write_byte(1, 0x426a, 0x2, 0);

      +err |= pm_spmi_lite_write_byte(1, 0x42d0, 0xA5, 0);
      + err |= pm_spmi_lite_write_byte(1, 0x426b, 0x34, 0);

      + err |= pm_spmi_lite_write_byte(1, 0x4240, 0x2, 0);
      + err |= pm_spmi_lite_write_byte(1, 0x4241, 0x34, 0);

      }
      (2)RPM change:
      rpm_proc\core\power\railway_v2\src\8x26\railway_config.c:
      default_uvs_mx_8x26_v1
      default_uvs_mx_8x26_v2
      default_uvs_mx_8x26_v2p1
      0, // RAILWAY_NO_REQUEST
      750000,  //RAILWAY_RETENTION
      1150000+50000,  //RAILWAY_SVS_KRAIT
      1150000+50000,  //RAILWAY_SVS_SOC
      1150000+50000,  //RAILWAY_NOMINAL
```

```
1275000+50000,  //RAILWAY_TURBO
1275000+50000,  //RAILWAY_TURBO_HIGH
1275000+50000,  //RAILWAY_SUPER_TURBO
1275000+50000,  //RAILWAY_SUPER_TURBO_NO_CPR
(3)rpm_proc\core\systemdrivers\pmic\config\msm8x26\pm_config_target.c
the following code will change the range of LDO3(vdd_mx) in software.
In pm_rpm_ldo_rail_info_type ldo_rail_a[NUM_OF_LDO_A] =
Change from:
{10, 25, 0, 1, 1, PM_NPA_SW_MODE_LDO__IPEAK, 700, 1330 }, // LDO3 NMOS;
To:
{10, 25, 0, 1, 1, PM_NPA_SW_MODE_LDO__IPEAK, 700, 1400 }, // LDO3 NMOS;
```

4.10.1.3 Disable CPR

```
\rpm_proc\core\power\rbcpr\src\target\8x26\rbcpr_bsp.c
const rbcpr_bsp_rail_params_type rbcpr_settings_8x26_tn1[] =
....
-.rbcpr_enablement=RBCPR_ENABLED_CLOSED_LOOP,
+.rbcpr_enablement=RBCPR_DISABLED,

const rbcpr_bsp_rail_params_type rbcpr_settings_8926_tn2[] =
....
-.rbcpr_enablement=RBCPR_ENABLED_CLOSED_LOOP,
+.rbcpr_enablement=RBCPR_DISABLED,
const rbcpr_bsp_rail_params_type rbcpr_settings_8926_tn3[]
....
-.rbcpr_enablement=RBCPR_ENABLED_CLOSED_LOOP,
+.rbcpr_enablement=RBCPR_DISABLED,
```

4.10.2 MSM8939

4.10.2.1 VddCx bump up

For the MSM8936 and MSM8939 platforms, the RBCPR for vdd_cx is moved to the SBL code.

For the MSM8916 platform, the RBCPR for vdd_cx is in RPM code.

There are two configurations about vdd_cx in railway_config.c and rbcpr_bsp.c.

.corner = RAILWAY_SVS_SOC,

.voltage_ceil = 1050000, this is the ceil voltage of CPR for the SVS mode.

.voltage_floor = 900000, this is the floor voltage of CPR for the SVS mode

The voltage in temp_config_data equals to the voltage_ceil.

If the CPR disabled, the code uses the voltage in temp_config_data

If the CPR enabled, the code uses the min voltage in these two configures.

Hence, there are three ways to bump up the vdd_cx.

4.10.2.1.1 Disable RBCPR+Modify code in RPM(railway_config.c)

(1)Bump up the retention voltage by 50mV.

```
/rpm_proc/core/power/sleep/src/8936/sleep_target_config.c
static const uint32 vddcx_pvs_retention_data[8] =
{
 /* 000 */ 650000+50000,
 /* 001 */ 500000+50000,
 /* 010 */ 650000+50000,
 /* 011 */ 650000+50000,
 /* 100 */ 650000+50000,
 /* 101 */ 650000+50000,
 /* 110 */ 650000+50000,
 /* 111 */ 650000+50000
};
```

(2)\rpm_proc\core\power\railway_v2\src\8936\railway_config.c

```
// VddCx
{
.vreg_name = "VddCx",
..........
.default_uvs = (const unsigned[])
{
0, // RAILWAY_NO_REQUEST
550000, // RAILWAY_RETENTION → it is not used actually.
1050000+50000, // RAILWAY_SVS_KRAIT
1050000+50000, // RAILWAY_SVS_SOC
1150000+50000, // RAILWAY_NOMINAL
1275000+50000, // RAILWAY_TURBO
1275000+50000, // RAILWAY_TURBO_HIGH
1275000+50000, // RAILWAY_SUPER_TURBO
1275000+50000, // RAILWAY_SUPER_TURBO_NO_CPR
},
….
},
(3)disable RBCPR in sbl code.
 boot_images\core\power\cpr\src\target\8936\rbcpr_bsp.c
rbcpr_enablement=RBCPR_DISABLED
```

4.10.2.1.2 Enable RBCPR+Modify code in RPM and SBL

(1)Bump up the retention voltage by 50mV.

```
/rpm_proc/core/power/sleep/src/8936/sleep_target_config.c
static const uint32 vddcx_pvs_retention_data[8] =
{
 /* 000 */ 650000+50000,
 /* 001 */ 500000+50000,
 /* 010 */ 650000+50000,
 /* 011 */ 650000+50000,
 /* 100 */ 650000+50000,
 /* 101 */ 650000+50000,
 /* 110 */ 650000+50000,
 /* 111 */ 650000+50000
};

(2)\rpm_proc\core\power\railway_v2\src\8936\railway_config.c
// VddCx
{
.vreg_name = "VddCx",
..........
.default_uvs = (const unsigned[])
{
0, // RAILWAY_NO_REQUEST
550000, // RAILWAY_RETENTION → it is not used actually.
1050000+50000, // RAILWAY_SVS_KRAIT
1050000+50000, // RAILWAY_SVS_SOC
1150000+50000, // RAILWAY_NOMINAL
1275000+50000, // RAILWAY_TURBO
1275000+50000, // RAILWAY_TURBO_HIGH
1275000+50000, // RAILWAY_SUPER_TURBO
1275000+50000, // RAILWAY_SUPER_TURBO_NO_CPR
},
….
},
(3)\boot_images\core\power\cpr\src\target\8936\rbcpr_bsp.c
Pls modify all the xxx_xx__cpr_settings[] struct.
        .voltage_ceil  = 1050000 +50000,
        .voltage_floor = 900000+50000,


const rbcpr_bsp_rail_params_type gf_tn1_cpr_settings[] =
{
   {  // VddCx
    .target_params = (rbcpr_corner_params_type[])
    {
      {
          …..
        .corner      = RAILWAY_SVS_SOC,
        .fuse       = HAL_RBCPR_FUSE_SVS,
        .voltage_ceil  = 1050000 +50000,
        .voltage_floor = 900000+50000,
      },
      {
        ………….
        .corner      = RAILWAY_NOMINAL,
        .fuse       = HAL_RBCPR_FUSE_NOMINAL,
        .voltage_ceil  = 1150000+50000,
        .voltage_floor = 1000000+50000,
      },
      {
        …………….
        .corner     = RAILWAY_SUPER_TURBO,
        .fuse      = HAL_RBCPR_FUSE_TURBO,
        .voltage_ceil  = 1287500+50000,
        .voltage_floor = 1137500+50000,
      },
    },
    .rbcpr_enablement= RBCPR_ENABLED_CLOSED_LOOP,
    .number_of_target_params=3,
  },
};

const rbcpr_bsp_rail_params_type gf_tn3_cpr_settings[] =
{
   {  // VddCx
    .target_params = (rbcpr_corner_params_type[])
    {
      {
        …………
        .corner     = RAILWAY_SVS_SOC,
        .fuse      = HAL_RBCPR_FUSE_SVS,
        .voltage_ceil  = 1050000+50000,
        .voltage_floor = 900000+50000,
      },
  ……………..
  },
};

……………
tsmc_tn1_cpr_settings[]
tsmc_tn3_cpr_settings[]
..
```

4.10.2.1.3 Enable RBCPR+ bump floor_voltage only

```
\boot_images\core\power\cpr\src\target\8936\rbcpr_bsp.c
Pls modify all the xxx_xx__cpr_settings[] struct.
          .voltage_ceil  = 1050000,
          .voltage_floor = 900000+50000,

const rbcpr_bsp_rail_params_type gf_tn1_cpr_settings[] =
{
   {  // VddCx
     .target_params = (rbcpr_corner_params_type[])
     {
       {
             ….. 
          .corner     = RAILWAY_SVS_SOC,
          .fuse       = HAL_RBCPR_FUSE_SVS,
          .voltage_ceil  = 1050000,
          .voltage_floor = 900000+50000,
        },
        {
          …………….
          .corner      = RAILWAY_NOMINAL,
          .fuse       = HAL_RBCPR_FUSE_NOMINAL,
          .voltage_ceil  = 1150000,
          .voltage_floor = 1000000+50000,
        },
        {
          ………………….
          .corner      = RAILWAY_SUPER_TURBO,
          .fuse       = HAL_RBCPR_FUSE_TURBO,
          .voltage_ceil  = 1287500,
          .voltage_floor = 1137500+50000,
        },
      },
      .rbcpr_enablement= RBCPR_ENABLED_CLOSED_LOOP,
      .number_of_target_params=3,
    },
  };

  const rbcpr_bsp_rail_params_type gf_tn3_cpr_settings[] =
  {
     {  // VddCx
      .target_params = (rbcpr_corner_params_type[])
      {
        {
          ……………
          .corner      = RAILWAY_SVS_SOC,
          .fuse       = HAL_RBCPR_FUSE_SVS,
          .voltage_ceil  = 1050000,
          .voltage_floor = 900000+50000,
        },
    ………………..
    },
  };

  ………………
  tsmc_tn1_cpr_settings[]
  tsmc_tn3_cpr_settings[]
```

4.10.2.2 VddMx bump up

Bump up the retention voltage by 50mV.

```
/rpm_proc/core/power/sleep/src/8936/sleep_target_config.c
      static const uint32 vddmx_pvs_retention_data[4] =
      {
       /* 00 */ 750000 +50000,
       /* 01 */ 650000+50000,
       /* 10 */ 750000+50000,
       /* 11 */ 750000+50000,
      };
```

Bump up the vdd_mx to by 50mv

```
       (1)RPM change:

      rpm_proc\core\power\railway_v2\src\8936\railway_config.c:

      temp_config_data{
      ……
      .vreg_name   = "vddmx",
      ……
              0,            // RAILWAY_NO_REQUEST
              750000+50000,         // RAILWAY_RETENTION
              1050000+50000,        // RAILWAY_SVS_KRAIT
              1050000+50000,        // RAILWAY_SVS_SOC
              1150000+50000,        // RAILWAY_NOMINAL
              1287500+50000,        // RAILWAY_TURBO
              1287500+50000,        // RAILWAY_TURBO_HIGH
              1287500+50000,        // RAILWAY_SUPER_TURBO
              1287500+50000,        // RAILWAY_SUPER_TURBO_NO_CPR

      (2)
      File Changed :
      /rpm_proc/core/systemdrivers/pmic/config/msm8939/pm_config_target.c
      As there should be 62.5mV headroom between S3 and it’ s child LDO’ s, we
      need to bump up the S3 voltage as well.

      pm_rpm_smps_rail_info_type smps_rail_a[NUM_OF_SMPS_A] =
      {
      -  {300, 0, PM_ACCESS_ALLOWED,  PM_ALWAYS_ON,  PM_NPA_SW_MODE_SMPS__NPM,
      PM_CLK_1p6_MHz, PM_CLK_1p6_MHz, 1250, 1350}, //HFS3
        +  {300, 0, PM_ACCESS_ALLOWED,  PM_ALWAYS_ON,  PM_NPA_SW_MODE_SMPS__NPM,
      PM_CLK_1p6_MHz, PM_CLK_1p6_MHz, 1250, 1400}, //HFS3
      };
```

4.10.2.3 Disable CPR

```
      boot_images/core/power/cpr/src/target/8936/rbcpr_bsp.c

      37rbcpr_bsp_rail_params_type gf_tn3_cpr_settings[] =
      38{
```

```
39 { // VddCx
- .rbcpr_enablement=RBCPR_ENABLED_CLOSED_LOOP,
+ .rbcpr_enablement=RBCPR_DISABLED,


109const rbcpr_bsp_rail_params_type tsmc_tn3_cpr_settings[] =
110{
111 { // VddCx
- .rbcpr_enablement=RBCPR_ENABLED_CLOSED_LOOP,
+ .rbcpr_enablement=RBCPR_DISABLED,
```

4.10.3 MSM8996

4.10.3.1 VddCx and VddMx bump up

```
File: Cpr_voltage_ranges_bsp.c
(boot_images\qcompkg\msm8996pkg\library\cprtargetlib)
static const cpr_config_versioned_voltage_ranges_t
SS_8996_mx_voltage_ranges =
…snip…
.voltage_level = (const cpr_config_voltage_level_t[])
{
//Mode, Ceiling, Floor, Fuse-Ref, max_floor_to_ceil, ceiling_correction,
floor_correction, volt_fuse_type, offset_fuse_type
- {CPR_VOLTAGE_MODE_SVS, 850000, 850000, 850000, 0, 0, 0, CPR_FUSE_NO_FUSE,
CPR_FUSE_NO_FUSE},
- {CPR_VOLTAGE_MODE_NOMINAL, 950000, 850000, 950000, 0, 0, 0,
CPR_FUSE_NOMINAL, CPR_FUSE_NO_FUSE},
- {CPR_VOLTAGE_MODE_SUPER_TURBO, 1015000, 855000, 1015000, 0, 0, 0,
CPR_FUSE_TURBO, CPR_FUSE_NO_FUSE},
+ {CPR_VOLTAGE_MODE_SVS, 850000+50000, 850000, 850000+50000, 0, 0, 0,
CPR_FUSE_NO_FUSE, CPR_FUSE_NO_FUSE},
+ {CPR_VOLTAGE_MODE_NOMINAL, 950000+50000, 850000, 950000+50000, 0, 0, 0,
CPR_FUSE_NOMINAL, CPR_FUSE_NO_FUSE},
+ {CPR_VOLTAGE_MODE_SUPER_TURBO, 1015000+50000, 855000, 1015000+50000, 0,
0, 0, CPR_FUSE_TURBO, CPR_FUSE_NO_FUSE},
},
static const cpr_config_versioned_voltage_ranges_t cx_voltage_ranges_v1_2 =
…snip…
.voltage_level = (const cpr_config_voltage_level_t[])
{
//Mode, Ceiling, Floor, Fuse-Ref, max_floor_to_ceil, ceiling_correction,
floor_correction, volt_fuse_type, offset_fuse_type
- {CPR_VOLTAGE_MODE_SVS2, 670000, 520000, 670000, 70000, 0, 0,
CPR_FUSE_SVS2, CPR_FUSE_NO_FUSE},
- {CPR_VOLTAGE_MODE_SVS, 745000, 520000, 745000, 75000, 0, 0, CPR_FUSE_SVS,
CPR_FUSE_NO_FUSE},
- {CPR_VOLTAGE_MODE_NOMINAL, 935000, 520000, 905000, 90000, 0, 0,
CPR_FUSE_NOMINAL, CPR_FUSE_NO_FUSE},
- {CPR_VOLTAGE_MODE_SUPER_TURBO, 1015000, 520000, 1015000, 100000, 0, 0,
CPR_FUSE_TURBO, CPR_FUSE_NO_FUSE},
+ {CPR_VOLTAGE_MODE_SVS2, 670000+50000, 520000, 670000+50000, 70000, 0, 0,
CPR_FUSE_SVS2, CPR_FUSE_NO_FUSE},
+ {CPR_VOLTAGE_MODE_SVS, 745000+50000, 520000, 745000+50000, 75000, 0, 0,
CPR_FUSE_SVS, CPR_FUSE_NO_FUSE},
+ {CPR_VOLTAGE_MODE_NOMINAL, 935000+50000, 520000, 905000+50000, 90000, 0,
0, CPR_FUSE_NOMINAL, CPR_FUSE_NO_FUSE},
+ {CPR_VOLTAGE_MODE_SUPER_TURBO, 1015000+50000, 520000, 1015000+50000,
100000, 0, 0, CPR_FUSE_TURBO, CPR_FUSE_NO_FUSE},
},
static const cpr_config_versioned_voltage_ranges_t cx_voltage_ranges_v3_0 =
…snip…
.voltage_level = (const cpr_config_voltage_level_t[])
{
//Mode, Ceiling, Floor, Fuse-Ref, max_floor_to_ceil, ceiling_correction,
floor_correction, volt_fuse_type, offset_fuse_type
- {CPR_VOLTAGE_MODE_SVS2, 670000, 520000, 670000, 70000, 0, 0,
CPR_FUSE_SVS2, CPR_FUSE_NOMINAL_OFFSET},
- {CPR_VOLTAGE_MODE_SVS, 745000, 520000, 745000, 75000, 0, 0, CPR_FUSE_SVS,
CPR_FUSE_NOMINAL_OFFSET},
- {CPR_VOLTAGE_MODE_NOMINAL, 935000, 520000, 905000, 90000, 30000, 0,
CPR_FUSE_NOMINAL, CPR_FUSE_NO_FUSE},
- {CPR_VOLTAGE_MODE_SUPER_TURBO, 1015000, 520000, 1015000, 100000, -10000,
0, CPR_FUSE_TURBO, CPR_FUSE_TURBO_OFFSET},
+ {CPR_VOLTAGE_MODE_SVS2, 670000+50000, 520000, 670000+50000, 70000, 0, 0,
CPR_FUSE_SVS2, CPR_FUSE_NOMINAL_OFFSET},
+ {CPR_VOLTAGE_MODE_SVS, 745000+50000, 520000, 745000+50000, 75000, 0, 0,
CPR_FUSE_SVS, CPR_FUSE_NOMINAL_OFFSET},
+ {CPR_VOLTAGE_MODE_NOMINAL, 935000+50000, 520000, 905000+50000, 90000,
30000, 0, CPR_FUSE_NOMINAL, CPR_FUSE_NO_FUSE},
+ {CPR_VOLTAGE_MODE_SUPER_TURBO, 1015000+50000, 520000, 1015000+50000,
100000, -10000, 0, CPR_FUSE_TURBO, CPR_FUSE_TURBO_OFFSET},
},
…snip…
static const cpr_config_versioned_voltage_ranges_t cx_voltage_ranges_v3_1 =
…snip…
.voltage_level = (const cpr_config_voltage_level_t[])
{
//Mode, Ceiling, Floor, Fuse-Ref, max_floor_to_ceil, ceiling_correction,
floor_correction, volt_fuse_type, offset_fuse_type
- {CPR_VOLTAGE_MODE_SVS2, 670000, 520000, 670000, 70000, -30000, 0,
CPR_FUSE_SVS2, CPR_FUSE_NOMINAL_OFFSET},
- {CPR_VOLTAGE_MODE_SVS, 745000, 520000, 745000, 75000, -30000, 0,
CPR_FUSE_SVS, CPR_FUSE_NOMINAL_OFFSET},
- {CPR_VOLTAGE_MODE_NOMINAL, 935000, 520000, 905000, 90000, 0, 0,
CPR_FUSE_NOMINAL, CPR_FUSE_NO_FUSE},
- {CPR_VOLTAGE_MODE_SUPER_TURBO, 1015000, 520000, 1015000, 100000, -10000,
0, CPR_FUSE_TURBO, CPR_FUSE_TURBO_OFFSET},
+ {CPR_VOLTAGE_MODE_SVS2, 670000+50000, 520000, 670000+50000, 70000, -
30000, 0, CPR_FUSE_SVS2, CPR_FUSE_NOMINAL_OFFSET},
+ {CPR_VOLTAGE_MODE_SVS, 745000+50000, 520000, 745000+50000, 75000, -30000,
0, CPR_FUSE_SVS, CPR_FUSE_NOMINAL_OFFSET},
+ {CPR_VOLTAGE_MODE_NOMINAL, 935000+50000, 520000, 905000+50000, 90000, 0,
0, CPR_FUSE_NOMINAL, CPR_FUSE_NO_FUSE},
+ {CPR_VOLTAGE_MODE_SUPER_TURBO, 1015000+50000, 520000, 1015000+50000,
100000, -10000, 0, CPR_FUSE_TURBO, CPR_FUSE_TURBO_OFFSET},
},
static const cpr_config_versioned_voltage_ranges_t cx_voltage_ranges_v3_2 =
…snip…
.voltage_level = (const cpr_config_voltage_level_t[])
{
//Mode, Ceiling, Floor, Fuse-Ref, max_floor_to_ceil, ceiling_correction,
floor_correction, volt_fuse_type, offset_fuse_type
- {CPR_VOLTAGE_MODE_SVS2, 670000, 520000, 670000, 70000, -30000, 0,
CPR_FUSE_SVS2, CPR_FUSE_NOMINAL_OFFSET},
- {CPR_VOLTAGE_MODE_SVS, 745000, 520000, 745000, 75000, -30000, 0,
CPR_FUSE_SVS, CPR_FUSE_NOMINAL_OFFSET},
- {CPR_VOLTAGE_MODE_NOMINAL, 935000, 520000, 905000, 90000, 0, 0,
CPR_FUSE_NOMINAL, CPR_FUSE_NO_FUSE},
- {CPR_VOLTAGE_MODE_SUPER_TURBO, 1015000, 520000, 1015000, 100000, -10000,
0, CPR_FUSE_TURBO, CPR_FUSE_TURBO_OFFSET},
+ {CPR_VOLTAGE_MODE_SVS2, 670000+50000, 520000, 670000+50000, 70000, -
30000, 0, CPR_FUSE_SVS2, CPR_FUSE_NOMINAL_OFFSET},
+ {CPR_VOLTAGE_MODE_SVS, 745000+50000, 520000, 745000+50000, 75000, -30000,
0, CPR_FUSE_SVS, CPR_FUSE_NOMINAL_OFFSET},
+ {CPR_VOLTAGE_MODE_NOMINAL, 935000+50000, 520000, 905000+50000, 90000, 0,
0, CPR_FUSE_NOMINAL, CPR_FUSE_NO_FUSE},
+ {CPR_VOLTAGE_MODE_SUPER_TURBO, 1015000+50000, 520000, 1015000+50000,
100000, -10000, 0, CPR_FUSE_TURBO, CPR_FUSE_TURBO_OFFSET},
},
…snip…

static const cpr_config_versioned_voltage_ranges_t cx_voltage_ranges_v3_3 =
…snip…
.voltage_level = (const cpr_config_voltage_level_t[])
{
//Mode, Ceiling, Floor, Fuse-Ref, max_floor_to_ceil, ceiling_correction,
floor_correction, volt_fuse_type, offset_fuse_type
- {CPR_VOLTAGE_MODE_SVS2, 670000, 520000, 670000, 70000, 0, 0,
CPR_FUSE_SVS2, CPR_FUSE_NOMINAL_OFFSET},
- {CPR_VOLTAGE_MODE_SVS, 745000, 520000, 745000, 75000, 0, 0, CPR_FUSE_SVS,
CPR_FUSE_NOMINAL_OFFSET},
- {CPR_VOLTAGE_MODE_NOMINAL, 935000, 520000, 905000, 90000, 0, 0,
CPR_FUSE_NOMINAL, CPR_FUSE_NO_FUSE},
- {CPR_VOLTAGE_MODE_SUPER_TURBO, 1015000, 520000, 1015000, 100000, 0, 0,
CPR_FUSE_TURBO, CPR_FUSE_TURBO_OFFSET},
+ {CPR_VOLTAGE_MODE_SVS2, 670000+50000, 520000, 670000+50000, 70000, 0, 0,
CPR_FUSE_SVS2, CPR_FUSE_NOMINAL_OFFSET},
+ {CPR_VOLTAGE_MODE_SVS, 745000+50000, 520000, 745000+50000, 75000, 0, 0,
CPR_FUSE_SVS, CPR_FUSE_NOMINAL_OFFSET},
+ {CPR_VOLTAGE_MODE_NOMINAL, 935000+50000, 520000, 905000+50000, 90000, 0,
0, CPR_FUSE_NOMINAL, CPR_FUSE_NO_FUSE},
+ {CPR_VOLTAGE_MODE_SUPER_TURBO, 1015000+50000, 520000, 1015000+50000,
100000, 0, 0, CPR_FUSE_TURBO, CPR_FUSE_TURBO_OFFSET},
},
```

4.10.3.2 Pin VddCx or VddMx to Turbo mode

```
File: railway_config.c (rpm_proc\core\power\railway_v2\src\8996)
void railway_init_proxies_and_pins(void)
{
//snip//
rpm_send_init_proxy_vote(RPM_SMPS_A_REQ, // Cx is S1
1,
0, // APPS is 0
kvp);
+const int cx_rail_id = rail_id("vddcx");
+assert(RAIL_NOT_SUPPORTED_BY_RAILWAY != cx_rail_id);
+railway_voter_t cx_pin = railway_create_voter(cx_rail_id, true,
RAILWAY_RPM_INIT_VOTER);
+railway_corner_vote(cx_pin, RAILWAY_SUPER_TURBO);
}
```

4.10.3.3 Disable CPR on VddCx and VddMx

```
File: cpr_enablement_bsp.c
(boot_images\QcomPkg\Msm8996Pkg\Library\CPRTargetLib)
static const cpr_enablement_versioned_rail_config_t mx_8996 =
{
.hw_versions =
{
.foundry_range = (const cpr_config_foundry_range[])
{
// Foundry Chip Min Rev Chip Max Rev CPR Min Rev CPR Max Rev
{CPR_FOUNDRY_SS, CPR_CHIPINFO_VERSION(0,0), CPR_CHIPINFO_VERSION(0xFF,
0xFF), 0, 0xFF},
},
.foundry_range_count = 1,
},
- .enablement_init_params = &CPR_ENABLE_CLOSED_HIGH_VOLTAGE_PARTS,
+ .enablement_init_params = &CPR_ENABLE_GLOBAL_CEILING_VOLTAGE,
<snip>
};

<snip>
```

```
static const cpr_enablement_versioned_rail_config_t cx_8996_v1_2 =
{
.hw_versions =
{
.foundry_range = (const cpr_config_foundry_range[])
{
// Foundry Chip Min Rev Chip Max Rev CPR Min Rev CPR Max Rev
{CPR_FOUNDRY_SS, CPR_CHIPINFO_VERSION(0,0), CPR_CHIPINFO_VERSION(3, 0), 0,
0xFF},
},
.foundry_range_count = 1,
},
- .enablement_init_params = &CPR_ENABLE_CLOSED_HIGH_VOLTAGE_PARTS,
+ .enablement_init_params = &CPR_ENABLE_GLOBAL_CEILING_VOLTAGE,
<snip>
};

static const cpr_enablement_versioned_rail_config_t cx_8996_v3_1 =
{
.hw_versions =
{
.foundry_range = (const cpr_config_foundry_range[])
{
// Foundry Chip Min Rev Chip Max Rev CPR Min Rev CPR Max Rev
{CPR_FOUNDRY_SS, CPR_CHIPINFO_VERSION(3,0), CPR_CHIPINFO_VERSION(0xFF,
0xFF), 0, 2},
},
.foundry_range_count = 1,
},
- .enablement_init_params = &CPR_ENABLE_CLOSED_HIGH_VOLTAGE_PARTS,
+ .enablement_init_params = &CPR_ENABLE_GLOBAL_CEILING_VOLTAGE,
<snip>
};

static const cpr_enablement_versioned_rail_config_t cx_8996_v3_2 =
{
.hw_versions =
{
.foundry_range = (const cpr_config_foundry_range[])
{
// Foundry Chip Min Rev Chip Max Rev CPR Min Rev CPR Max Rev
{CPR_FOUNDRY_SS, CPR_CHIPINFO_VERSION(3,0), CPR_CHIPINFO_VERSION(0xFF,
0xFF), 2, 3},
},
.foundry_range_count = 1,
},
- .enablement_init_params = &CPR_ENABLE_CLOSED_HIGH_VOLTAGE_PARTS,
+ .enablement_init_params = &CPR_ENABLE_GLOBAL_CEILING_VOLTAGE,
        <snip>
        };

        static const cpr_enablement_versioned_rail_config_t cx_8996_v3_3 =
        {
        .hw_versions =
        {
        .foundry_range = (const cpr_config_foundry_range[])
        {
        // Foundry Chip Min Rev Chip Max Rev CPR Min Rev CPR Max Rev
        {CPR_FOUNDRY_SS, CPR_CHIPINFO_VERSION(3,0), CPR_CHIPINFO_VERSION(0xFF,
        0xFF), 3, 4},
        },
        .foundry_range_count = 1,
        },
        - .enablement_init_params = &CPR_ENABLE_CLOSED_HIGH_VOLTAGE_PARTS,
        + .enablement_init_params = &CPR_ENABLE_GLOBAL_CEILING_VOLTAGE,
        <snip>
        };

        static const cpr_enablement_versioned_rail_config_t cx_8996_v3_4 =
        {
        .hw_versions =
        {
        .foundry_range = (const cpr_config_foundry_range[])
        {
        // Foundry Chip Min Rev Chip Max Rev CPR Min Rev CPR Max Rev
        {CPR_FOUNDRY_SS, CPR_CHIPINFO_VERSION(3,0), CPR_CHIPINFO_VERSION(0xFF,
        0xFF), 4, 0xff},
        },
        .foundry_range_count = 1,
        },
        - .enablement_init_params = &CPR_ENABLE_CLOSED_HIGH_VOLTAGE_PARTS,
        + .enablement_init_params = &CPR_ENABLE_GLOBAL_CEILING_VOLTAGE,
        <snip>
};
```

4.11 Disable periodic training

This is applicable for LPDDR4. This is a hardware feature for LPDDR4, when frequency is greater than the threshold; periodically training some params.

4.11.1 MSM8996

```
boot_images/QcomPkg/Msm8996Pkg/Library/DSFTargetLib/target/8996/settings/ta
rget_config.h
- #define DSF_PERIODIC_TRAINING_EN           1
```

```
+ #define DSF_PERIODIC_TRAINING_EN           0

rpm_proc/core/boot/ddr/hw/hw_sequence/target/8996/settings/target_config.h
- #define DSF_PERIODIC_TRAINING_EN           1
+ #define DSF_PERIODIC_TRAINING_EN           0

boot_images/QcomPkg/Msm8996Pkg/Library/DDRTargetLib/ddr_target.c
- #define PERIODIC_TRAINING TRUE
+ #define PERIODIC_TRAINING FALSE
```

4.11.2 MSM8994

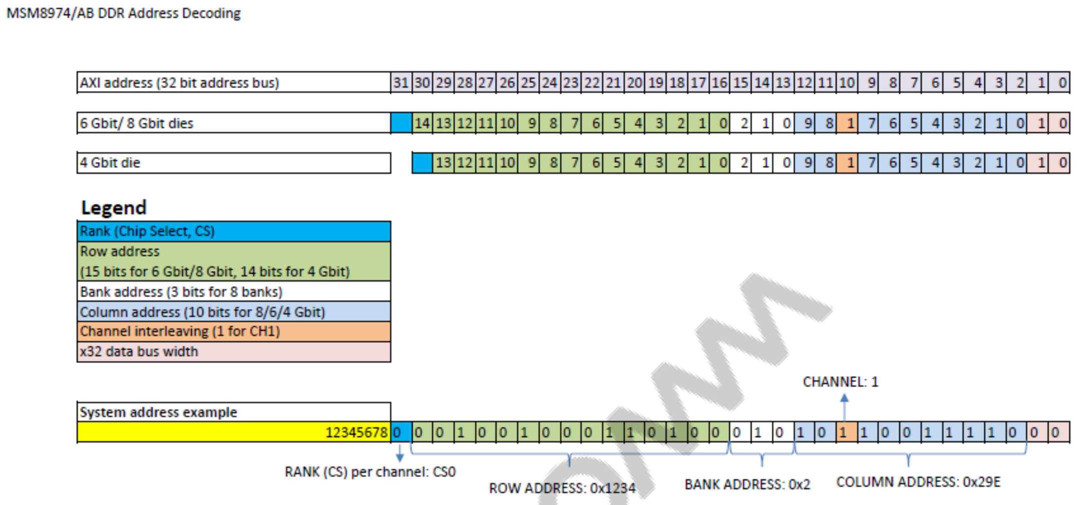
```
boot_images/core/boot/ddr/hw/hw_sequence/target/8994/settings/
target_config.h

- #define DSF_PERIODIC_TRAINING_EN           1
+ #define DSF_PERIODIC_TRAINING_EN           0

rpm_proc/core/boot/ddr/hw/hw_sequence/target/8994/settings/target_config.h
- #define DSF_PERIODIC_TRAINING_EN           1
+ #define DSF_PERIODIC_TRAINING_EN           0

boot_images/core/boot/ddr/hw/msm8994/ ddr_target.c

- #define PERIODIC_TRAINING TRUE
+ #define PERIODIC_TRAINING FALSE
```

4.12 DDR decode addressing

If a crash indicating bitflips on fixed physical addresses is found, it is linked to specific DDR chipset issues. For further checking from the DDR vendor, it may be required to provide platform decode addressing to map the address to DDR inside locations (rank, row, column, and bank).

QTI platforms uses RKRBC map.

4.12.1 MSM8974


MSM8974/AB DDR Address Decoding
AXI address (32 bit address bus) | 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
6 Gbit/ 8 Gbit dies | 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0 2 1 0 9 8 1 7 6 5 4 3 2 1 0 1 0
4 Gbit die | 13 12 11 10 9 8 7 6 5 4 3 2 1 0 2 1 0 9 8 1 7 6 5 4 3 2 1 0 1 0
Legend
Rank (Chip Select, CS)
Row address (15 bits for 6 Gbit/8 Gbit, 14 bits for 4 Gbit)
Bank address (3 bits for 8 banks)
Column address (10 bits for 8/6/4 Gbit)
Channel interleaving (1 for CH1)
x32 data bus width
System address example
12345678 | 0 0 0 1 0 0 1 0 0 0 1 1 0 1 0 0 0 1 0 1 0 1 1 0 0 1 1 1 1 0 0 0
CHANNEL: 1
RANK (CS) per channel: CS0
ROW ADDRESS: 0x1234
BANK ADDRESS: 0x2
COLUMN ADDRESS: 0x29E

4.12.2 MSM8939

Num of Cols : 10, Num of Banks : 8, Num of Rows : 15

Address	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Physical address	X	RK	Row address															Bank			Col										32 bit Data	

4.12.3 MSM8994

Address Bit	b31	b30	b17	b16	b15	b14	b13	b12	b11	b10	b9	b3	b2	b1	b0
Bit Descriptions	Rank	Row			bank			Col2		Ch-Int	Col1				
	RK	r14	…	r0	b2	b1	b0	c9	c8	I	c7	…	c0		
Addressing	0	0	…	0	0	0	0	0	0	1	0	…	0	X	X

For LPDDR4, MSM side, EBI0 is connected to CH_A, CH_C of DDR side and EBI1 is connected to CH_B, CH_D of DDR side

DQ Byte#0 (00b) = CH_A DQ0 ~ DQ7 or CH_B DQ0 ~ DQ7

DQ Byte#1 (01b) = CH_A DQ8 ~ DQ15 or CH_B DQ8 ~ DQ15

DQ Byte#2 (10b) = CH_C DQ0 ~ DQ7 or CH_D DQ0 ~ DQ7

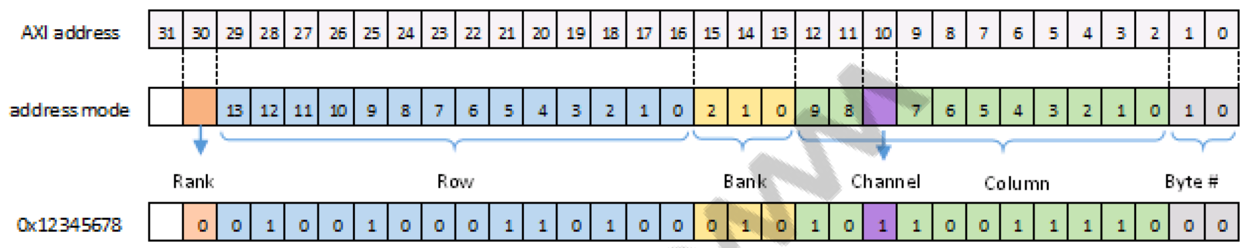
DQ Byte#3 (11b) = CH_C DQ8 ~ DQ15 or CH_D DQ8 ~ DQ15

4.12.4 SDM660 and SDM630

For a 2 GB DDR part whose addressing is configured with Row/Bank/Column = 14/8/10, the bus address can be decoded as described here. Decode address 0x12345678 as follows:

- rank – 0(0x0)

- row – 0010 0100 0111 0100 (0x1234)
- bank – 010 (0x2)
- channel – 1 (0x1)
- column – 10 1001 1110(0x29E)
- byte #(byte 0 ~ byte 3) – 00(0x0)



4.13 Erase DDR partition

For all the platforms that have DDR training, ensure to erase DDR partition for any changes about DDR parameters, including the driver strength, CDC delay, and AC timings in CDT XML.

This is also needed when DDR-related voltage or global voltage VddMx/VddCx changes.

Refer to the following command at fastboot:

```
Fastboot erase DDR
```

A References

A.1 Related documents

Title	Number
Qualcomm Technologies, Inc.	
MSM8909 DDR Test Instructions	80-NP408-15
DDR SDRAM CDT/ECDT User Guide	80-N1218-1
Guidelines New NAND_RAM Configuration in QSC6270/QSC6240	80-VN648-1
QMESA: Qualcomm Memory Stress Application User Guide	80-NH759-1
MSM8909/APQ8009 DDR Bitflip Issue	80-NP408-14
Flash Memory Customization Guide	80-VR646-1
Proposed Solution For 3 GB LPDDR3 in MSM8939	80-NK808-20
MSM8X26/MSM8X28+WTR2605 DDR Desense Issue	80-NC832-17
MSM8926/MSM8928 Chipset DDR Desense Issue Application Note	80-NE925-17
QRD Hardware Component Verification Schedule	80-VL976-6
MDM8215(M), MDM8615M, MDM8110M, MDM9X15(M) Software Interface	80-N5423-2
MSM8916 Hardware Register Description Document for OEMS	80-NK807-2x
QDUTT Application Notes For MSM8917/8937/8940/8952/8953/8976/8976pro Simplified Chinese	80-NR058-2SC
QDUTT Tool and DDR Test Training Simplified Chinese	80-NR058-3SC
QDUTT Application Notes For MSM8909/MSM8916 – Simplified Chinese	80-NR058-5SC
SDM660 SDM630 Boot DDR Debug Image User Guide Simplified Chinese	80-PC982-1SC

A.2 Acronyms and terms

Acronym or term	Definition
DDR	Double data rate
FPM	Frequency power manager
MCP	Multichip package