T23x BCT

Deployment Guide
## Document History

<table>
<thead>
<tr>
<th>Version</th>
<th>Date</th>
<th>Authors</th>
<th>Description of Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>01</td>
<td>June 2022</td>
<td></td>
<td>Initial release</td>
</tr>
</tbody>
</table>
# Table of Contents

## Chapter 1. Introduction
- 1.1 BR-BCT
- 1.2 MB1-BCT
- 1.3 Mem-BCT
- 1.4 MB2-BCT

## Chapter 2. Pinmux and GPIO Configuration

## Chapter 3. Common Prod Configuration

## Chapter 4. Controller Prod Configuration

## Chapter 5. Pad Voltage Binding

## Chapter 6. PMIC Configuration
- 6.1 Common Configuration
- 6.2 Rail-Specific Configuration
- 6.3 Relative Order of Execution of the PMIC Configuration by MB1

## Chapter 7. Storage Device Configuration
- 7.1 QSPI Flash Parameters
- 7.2 SDMMC Parameters
- 7.3 UFS Parameters
- 7.4 SATA Parameters

## Chapter 8. UPHY Lane Configuration

## Chapter 9. OEM FW Ratchet Configuration

## Chapter 10. BootROM Reset PMIC Configuration
- 10.1 Specifying AO Blocks

## Chapter 11. Miscellaneous Configuration
- 11.1 MB1 Feature Fields
- 11.1.1 Clock Data
- 11.1.2 AST Data
- 11.1.3 MB1 AST Data
- 11.1.4 Carveout Configuration
- 11.1.5 Coresight Data
- 11.1.6 Firmware Load and Entry Configuration
- 11.1.7 CPU Configuration
- 11.1.8 Other Configuration
- 11.1.8.1 MB1 Soft Fuse Configurations
11.1.8.2 Debug Controls ................................................................. 50
11.1.8.3 Boot Failure Controls .................................................. 50
11.1.8.4 On/Off IST Mode Controls ......................................... 51
11.1.8.5 Frequency Monitor Controls ....................................... 51

Chapter 12. SDRAM Configuration ........................................... 59
  12.1 Carveouts ........................................................................ 60
  12.2 GSC Carveouts ............................................................... 60
  12.3 Non-GSC Carveouts ........................................................ 61

Chapter 13. GPIO Interrupt Mapping Configuration ..................... 62

Chapter 14. MB2 BCT Misc Configuration .................................. 64
  14.1 MB2 Feature Fields ....................................................... 64
  14.2 MB2 Firmware Data ..................................................... 64

Chapter 15. Security Configuration ........................................... 66
Chapter 1. Introduction

Boot Configuration Table (BCT) is a set of platform-specific configuration data that is consumed by a boot component. BootROM and MB1 consume BCT in binary form, which is generated by parsing device tree source configuration files (with dts file extension) by tegrabct_v2.

Starting from T23x, the config file format has changed from legacy `<parameter> = <value>` to the Device Tree Source (DTS) format for following reasons:

- DTS format supports recursive inclusion and the overriding of properties though inclusion.
- DTC can covert DTS/DTSI to DTB, which is a essentially tree-like format and is easier to parse compared to the current config file format.

1.1 BR-BCT

BR-BCT is loaded by BootROM from storage in coldboot and by MB1 in recovery mode. The primary consumer of this BCT is BootROM, but some fields are also consumed by MB1 and CPU-BL.

1.2 MB1-BCT

MB1 BCT is loaded by MB1 from storage (in coldboot mode) or over USB (in RCM mode). It is primarily consumed by MB1 and is constructed out of the following multiple configuration files:

- “Pinmux and GPIO Configuration” on page 3
- “Common Prod Configuration” on page 7
- “Controller Prod Configuration” on page 9
- “Pad Voltage Binding” on page 12
- “PMIC Configuration” on page 14
- “Storage Device Configuration” on page 27
- “UPHY Lane Configuration” on page 33
1.3 Mem-BCT

MemBCT is like MB1-BCT in terms of loading and usage. However, it primarily contains SDRAM parameters that are used to initialize MC and EMC. Refer to SDRAM Configuration for more information.

1.4 MB2-BCT

MB2 BCT is loaded by MB1 from storage (in coldboot mode) or over USB (in RCM mode). It is primarily consumed by MB2.

It is constructed out of multiple configuration files as listed below:

- GPIO Interrupt Mapping Configuration
- Security Configuration
- MB2 BCT Misc Configuration
Chapter 2. Pinmux and GPIO Configuration

The pinmux configuration file provides pinmux and GPIO configuration. Pinmux and GPIO configuration is generated using pinmux spreadsheet.

The stark contrast in the New DTS format with respect to old legacy format is becasue of the pinmux sheet output.

The pinmux DTS file are kept in the hardware/nvidia/platform/t23x/<platform>/bct/ directory.

NEW DTS format example of pinmux configuration file:

/*This dtsi file was generated by
e3360_1099_slt_a01.xlsm Revision: 126 */ #include
<dt-bindings/pinctrl/pinctrl-tegra.h>

/ {
  pinmux@2430000 {
    pinctrl-names =
    "default", "drive",
    "unused";
    pinctrl-0 =
    &pinmux_default;
    pinctrl-1 = &drive_default;
    pinctrl-2 = &pinmux_unused_lowpower;

    pinmux_default: common {
      /* SPIO Pin Configuration */
      dap1_sclk_ps0 {
        nvidia,pins =
        "dap1_sclk_ps0";
        nvidia,function = "i2s1";
        nvidia,pull =
        <TEGRA_PIN_PULL_NONE>;
        nvidia,tristate =
        <TEGRA_PIN_DISABLE>;
      };
    }
  }
}
Pinmux and GPIO Configuration

nvidia, enable-input = <TEGRA_PIN_DISABLE>;
nvidia, lpdr = <TEGRA_PIN_DISABLE>;
}

dap1_dout_ps1 {
  nvidia, pins = "dap1_dout_ps1";
nvidia, function = "i2s1";
nvidia, pull = <TEGRA_PIN_PULL_NONE>;
nvidia, tristate = <TEGRA_PIN_DISABLE>;
nvidia, enable-input = <TEGRA_PIN_DISABLE>;
nvidia, lpdr = <TEGRA_PIN_DISABLE>;
}

dap1_din_ps2 {
  nvidia, pins = "dap1_din_ps2";
nvidia, function = "i2s1";
nvidia, pull = <TEGRA_PIN_PULL_DOWN>;
nvidia, tristate = <TEGRA_PIN_ENABLE>;
nvidia, enable-input = <TEGRA_PIN_ENABLE>;
nvidia, lpdr = <TEGRA_PIN_DISABLE>;
}

dap1_fs_ps3 {
  nvidia, pins = "dap1_fs_ps3";
nvidia, function = "i2s1";
nvidia, pull = <TEGRA_PIN_PULL_NONE>;
nvidia, tristate = <TEGRA_PIN_DISABLE>;
nvidia, enable-input = <TEGRA_PIN_DISABLE>;
nvidia, lpdr = <TEGRA_PIN_DISABLE>;
}
aud_mclk_ps4 {
    nvidia,pins = "aud_mclk_ps4";
    nvidia,function = "aud";
    nvidia,pull = <TEGRA_PIN_PULL_NONE>;
    nvidia,tristate = <TEGRA_PIN_DISABLE>;
    nvidia,enable-input = <TEGRA_PIN_DISABLE>;
    nvidia,lpdr = <TEGRA_PIN_DISABLE>;
};

soc_gpio31_ps6 {
    nvidia,pins = "soc_gpio31_ps6";
    nvidia,function = "sdmmc1";
    nvidia,pull = <TEGRA_PIN_PULL_UP>;
    nvidia,tristate = <TEGRA_PIN_ENABLE>;
    nvidia,enable-input = <TEGRA_PIN_ENABLE>;
    nvidia,lpdr = <TEGRA_PIN_DISABLE>;
};

soc_gpio32_ps7 {
    nvidia,pins = "soc_gpio32_ps7";
    nvidia,function = "spdif";
    nvidia,pull = <TEGRA_PIN_PULL_DOWN>;
    nvidia,tristate = <TEGRA_PIN_ENABLE>;
    nvidia,enable-input = <TEGRA_PIN_ENABLE>;
    nvidia,lpdr = <TEGRA_PIN_DISABLE>;
};

soc_gpio33_pt0 {
    nvidia,pins = "soc_gpio33_pt0"
};
nvidia, function = "spdif";
nvidia, pull =<TEGRA_PIN_PULL_NONE>;
nvidia, tristate =<TEGRA_PIN_DISABLE>;
nvidia, enable-input =<TEGRA_PIN_DISABLE>;
nvidia, lpdr =<TEGRA_PIN_DISABLE>;
}
}

drive_default: drive {
}
};

OLD CFG format

//////// Pinmux for used pins //////////
pinmux.0x02434060 = <value1>; //
gen1_i2c_scl_pc5.PADCTL_CONN_GEN1_I2C_SCL_0 pinmux.0x02434064 =
<value2>; // gen1_i2c_scl_pc5.PADCTL_CONN_CFG2TMC_GEN1_I2C_SCL_0
pinmux.0x02434068 = <value1>; //
gen1_i2c_sda_pc6.PADCTL_CONN_GEN1_I2C_SDA_0 pinmux.0x0243406C =
<value2>; // gen1_i2c_sda_pc6.PADCTL_CONN_CFG2TMC_GEN1_I2C_DA_0

//////// Pinmux for unused pins for low-power
configuration ////////// pinmux.0x02434040 = <value1>; //
gpio_wan4_ph0.PADCTL_CONN_GPIO_WAN4_0 pinmux.0x02434044 = <value2>; //
gpio_wan4_ph0.PADCTL_CONN_CFG2TMC_GPIO_WAN4_0 pinmux.0x02434048 = <value1>; //
gpio_wan3_ph1.PADCTL_CONN_GPIO_WAN3_0 pinmux.0x0243404C = <value2>; //
gpio_wan3_ph1.PADCTL_CONN_CFG2TMC_GPIO_WAN3_0
Chapter 3. Common Prod Configuration

The prod configurations are the system characterized values of interface and controller settings, which are required for the given interface to work reliably for a platform. The prod configuration are set separately at controller and pinmux/pad levels. This file contains the common pinmux/pad level prod settings.

Required properties:

- addr-value-data: List of <Absolute PADCTL register address, mask, data>

For each such entry in the prod configuration file, MB1 reads the data from the specified address, modifies the data based on mask and value, and writes the data back to the address.

```
val = read(address)
val = (val & ~mask) | (value & mask); write(val, address);
```

The common prod DTS file are kept in the hardware/nvidia/platform/t23x/<platform>/bct/ directory.

NEW DTS format example of the prod config file:

```
/dts-v1/;

/ {
  prod {
    addr-mask-data = <0x0c302030 0x0000100 0x00000000>,
    <0x0c302040 0x0000100 0x00000000>,
    <0x0244100c 0xffffffff 0x0000a000>,
    <0x02441004 0xffffffff 0x0000a000>;
  }
};
```

OLD CFG format

```
prod.major = 1;
prod.minor = 0;
prod.0x0c302030.0x0000100 =
```
0x00000000;
prod.0x0c302040.0x0000100 = 0x00000000;
prod.0x0244100c.0xff1ff000 = 0x0a00a000;
prod.0x02441004.0xff1ff000 = 0x0a00a000;
Chapter 4. Controller Prod Configuration

The prod configurations are the system characterized values of interface and controller settings, which are required for the given interface to work reliably for a platform. The prod configuration are set separately at controller and pinmux/pad levels. This file contains the controller level prod settings.

The DTS configuration file is of the following form:

```
/ {
    deviceprod {
        <controller-name>-
        <Instance> = <&Label>;
        #prod-cells = <N>;
        <Label>: <controller-name>@<base-address> {
            <mode> {
                prod = <<address offset> <mask> <value>>;
            };
        };
    };
}
```

where:

- Instance is controller instance id
- Label is the label assigned to the node which can be referenced for mapping instance to a node
- `<controller-name>` is predefined module name (sdmmc, qspi, se, i2c)
- `<base-address>` is base address of the controller
- `<mode>` is controller mode for which the prod setting needs to be applied (e.g. default, hs400, etc)
- `<address offset>` is the register address offset from base address of the controller/instance
- `<mask>` is the mask value (4 bytes, unsigned)
<value> is the data value (4 bytes, unsigned)

N specifies how many entries are there on prod setting tuples per configurations.

if #prod-cells == 3, there are three entries per prod configuration (address_offset, mask, and value).

The legacy config format used device instance <controller-name>,<instance-index> instead of using the base address of the device. The legacy format keeps one byte to store the instance, but new DTS format has a base address, which requires four bytes in the BCT. As a result, some of the fields in the BCT structure have to be shifted accordingly.

For each entry in the prod configuration file, MB1 reads data from the specified address, modifies the data based on the mask and value, and the data back to the address.

\[
\text{val} = \text{read(address)} \\
\text{val} = (\text{val} \& \sim \text{mask}) \mid (\text{value} \& \text{mask}); \text{write(val, address)};
\]

The common prod configuration file is in the

```
hardware/nvidia/platform/t23x/<platform>/bct/
```
directory.

**NEW DTS example of prod configuration file:**

```
/dts-v1/;
/
 {
 deviceprod {
 qspi-0 = <&qspi0>; qspi-1 = <&qspi1>; sdmmc-3 = <&sdmmc3>;
 #prod-cells = <0x3>;
 qspi0:
 qspi@327
 0000 { 
 default {
 prod = <0x00000004 0x7C00 0x0>, <0x00000004 0xFF 0x10>;
 };
 }
 qspi1:
 qspi @330
 0000 {
 default {
 }
 };
 }
```
Controller Prod Configuration

prod = <0x00000004 0x7C00 0x0>,
<0x00000004 0xFF 0x10>;
};
};
sdmmc3:
sdmm
c034
6000
0 {
default {
    prod = <0x000001e4 0x00003FFF 0x0>;
};
hs400 {
    prod = <0x00000100 0x1FFF0000 0x14080000>,
<0x0000010c 0x00003F00 0x00000028>;
};
    ddr52 {
        prod = <0x00000100 0x1FFF0000 0x14080000>;
    }
};
OLD CFG format

//Qspi0
deviceprod.qspi.0.default.0x03270004.0
x7C00 = 0x0 //TX Trimmer
deviceprod.qspi.0.default.0x03270004.0
xFF = 0x10 //RX Trimmer

//Qspi1
deviceprod.qspi.1.default.0x03300004.0
x7C00 = 0x0 //TX Trimmer
deviceprod.qspi.1.default.0x03300004.0
xFF = 0x10 //RX Trimmer

//SDMMC
deviceprod.sdmmc.3.default.0x034601e4.0x00003FFF = 0x0 // auto cal pd and
pu offsets deviceprod.sdmmc.3.hs400.0x03460100.0x1FF0000 = 0x14080000 //
tap and trim values deviceprod.sdmmc.3.hs400.0x0346010c.0x00003F00 =
0x00000028 // DQS trim val deviceprod.sdmmc.3.ddr52.0x03460100.0x1FF0000 =
0x14080000 // tap and trim values
Chapter 5. Pad Voltage Binding

Tegra pins and pads are designed to support multiple voltage levels at an interface. They can operate at 1.2 volts (V), 1.8 V or 3.3 V. Based on the interface and power tree of a given platform, the software must write to the correct voltage of these pads to enable interface. If pad voltage is higher than the I/O power rail, the pin does not work on that level. If pad voltage is lower than the I/O power rail, it can damage the SoC pads. Consequently, configuring the correct pad voltage is required, and this configuration is based on the power tree.

The Pad voltage DTSI is generated using pinmux spread sheet.

The prod configuration files are kept in the
hardware/nvidia/platform/t23x/<platform>/bct/ directory. The contrast in the New DTS format is because of the pinmux sheet output.

NEW DTS format example of pad-voltage configuration file:

/*This dtsi file was generated by e3360_1099_slt_a01.xlsm Revision: 126 */

/ { 
  pmc@c360000 { 
    io-pad-defaults { 
      sdmcc1_hv { 
        nvidia,io-pad-init-voltage = <IO_PAD_VOLTAGE_1_8V>; 
      }; 
      sdmcc3_hv { 
        nvidia,io-pad-init-voltage = <IO_PAD_VOLTAGE_1_8V>; 
      }; 
      eqos { 
        nvidia,io-pad-init-voltage = <IO_PAD_VOLTAGE_1_8V>; 
      }; 
      qspi { 
        nvidia,io-pad-init-voltage = <IO_PAD_VOLTAGE_1_8V>; 
      }; 
      debug { 
        nvidia,io-pad-init-voltage = <IO_PAD_VOLTAGE_1_8V>; 
      };
  }
old_cfg { 
    nvidia,io-pad-init-voltage = <IO_PAD_VOLTAGE_3_3V>; 
};

audio_hv {
    nvidia,io-pad-init-voltage = <IO_PAD_VOLTAGE_3_3V>; 
};

ufs {
    nvidia,io-pad-init-voltage = <IO_PAD_VOLTAGE_1_2V>; 
};

};

old_cfg format

pad-voltage.major = 1;
pad-voltage.minor = 0;
pad-voltage.0x0c36003c = 0x0000003e;
// PMC_IMPL_E_18V_PWR_0 pad-
voltage.0x0c360040 = 0x00000079; // 
PMC_IMPL_E_33V_PWR_0
Chapter 6. PMIC Configuration

During system boot, MB1 enables system power rails for CPU, CORE, DRAM and completes some system PMIC configurations. Here is a list of the typical configurations:

- Enabling rails
- Setting rail voltages
- FPS configurations

Enabling and setting of voltages of rails might require following platform-specific configurations:

- I2C commands to devices
- PWM commands to devices
- MMIO accesses to Tegra registers, either read-modify-write or write-only
- Delay after the commands

The entries in PMIC configuration files are either common or rail-specific.

6.1 Common Configuration

The common configuration parameters are applicable to all rails. Each PMIC common configuration is of the following form:

/ {
    pmic {
        <parameter> = <value>;
    };
};
where:

- `<parameter>` is one of the following:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>rail-count</td>
<td>Number of rails in the configuration file.</td>
</tr>
<tr>
<td>command-retries-count</td>
<td>The number of allowed command attempts.</td>
</tr>
<tr>
<td>wait-before-start-bus-clear-us</td>
<td>Wait timeout, in microseconds before issuing the bus clear command. The wait time is calculated as $1 &lt;&lt; n$ microseconds where $n$ is provided by this parameter.</td>
</tr>
</tbody>
</table>

### 6.2 Rail-Specific Configuration

The rail-specific configuration are divided into blocks.

- Each rail can have one or more blocks.
- Each block can have only one type of commands (I2C, PWM, or MMIO).

Each PMIC rail-specific configuration is of the following form:

```plaintext
/ {
  pmic {
    <rail-name> {
      block@<index> {
        <parameter> = <value>;
      };
    };
  };
};
```

where:

- `<rail-name>` identifies the rail and is one of the following:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>system</td>
<td>System PMIC configuration</td>
</tr>
<tr>
<td>cpu/cpu0</td>
<td>CPU rail configuration</td>
</tr>
<tr>
<td>cpu1</td>
<td>CPU rail configuration</td>
</tr>
<tr>
<td>core</td>
<td>Core/SOC rail configuration</td>
</tr>
<tr>
<td>memio</td>
<td>DRAM related rail configuration</td>
</tr>
<tr>
<td>thermal</td>
<td>External thermal sensor configuration</td>
</tr>
<tr>
<td>Name</td>
<td>Description</td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
</tr>
<tr>
<td>platform</td>
<td>Platform's other I2C configuration</td>
</tr>
</tbody>
</table>

- **block-<index>** The rail specific commands are divided into blocks. Each rail can have multiple blocks. Each block of given rails indexed starting from 0.
- **<parameter>** is one of the following:

<table>
<thead>
<tr>
<th>&lt;parameter&gt;</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;type&gt;</td>
<td>Type of commands in the block. Valid properties are i2c-controller, pwm or mmio.</td>
</tr>
</tbody>
</table>

```plaintext
commands {
  <group-name> { 
    command@N { 
      reg-addr = <reg-address>; mask = <reg-mask>; 
      value = <reg-value>; 
    };
  };
}
```

- **<group-name> node** is the logical command group name and it is OPT←IONAL. N is the sequential number for the command starting from 0. MMIO or I2C commands (based on type)

<table>
<thead>
<tr>
<th>Type of command block</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mmio</td>
<td>MMIO command (valid only if block has mmio property), where, &lt;address&gt; is absolute address of MMIO register and &lt;mask&gt; is the 32bit mask that is applied to the value read from the MMIO address to facilitate read-modify-write operation. &lt;value&gt; value written into the register.</td>
</tr>
<tr>
<td>i2c-controller</td>
<td>I2c command, where &lt;address&gt; is the I2c slave register address, and &lt;mask&gt; is I2C slave mask that is applied to the value read.</td>
</tr>
</tbody>
</table>
### PMIC Configuration

<table>
<thead>
<tr>
<th><code>&lt;i2c-parameter&gt;</code></th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;i2c-parameter&gt;</code></td>
<td>I2C parameter (valid only if block has i2c-controller property), which is one of the following:</td>
</tr>
<tr>
<td>block-delay</td>
<td>Delay (in microseconds), after each command in the block. <code>&lt;index&gt;</code> is the block index (starting from 0).</td>
</tr>
<tr>
<td>i2c-controller-id</td>
<td>I2C controller instance</td>
</tr>
<tr>
<td>slave-addr</td>
<td>7-bit I2C slave address</td>
</tr>
<tr>
<td>reg-data-size</td>
<td>Register size in bits. Valid values are 0 (1-byte), 8 (1-byte) and 16 (2-byte)</td>
</tr>
<tr>
<td>reg-addr-size</td>
<td>Register address size in bits. Valid values are 0 (1-byte)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><code>&lt;pwm-parameter&gt;</code></th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pwm</td>
<td>PWM parameter (valid only if block has pwm property), which is one of the following:</td>
</tr>
<tr>
<td>controller-id</td>
<td>PWM controller instance (0-7)</td>
</tr>
<tr>
<td>source-freq-hz</td>
<td>PWM clock source frequency (in Hz)</td>
</tr>
<tr>
<td>period-ns</td>
<td>PWM time period (in nanoseconds)</td>
</tr>
<tr>
<td>min-microvolts</td>
<td>Vout from PWM regulator if duty cycle is 0</td>
</tr>
<tr>
<td>max-microvolts</td>
<td>Vout from PWM regulator if duty cycle is 100</td>
</tr>
<tr>
<td>init-microvolts</td>
<td>Vout from PWM regulator after initialization</td>
</tr>
<tr>
<td>enable</td>
<td>0 [just configure PWM, do not enable]; 1 [enable PWM after configuring]</td>
</tr>
</tbody>
</table>

### 6.3 Relative Order of Execution of the PMIC Configuration by MB1

Apart from the order and point in boot in which the different rail configurations are executed, there is no difference in how MB1 treats each of these configurations. Depending on the platform, some of these configurations might be optional. Therefore, MB1 treats all configurations as optional and prints only a warning when a configuration is not provided in the MB1-BCT.
These sequences are executed in the following order by MB1:

1. External thermal sensor configuration.
2. Generic PMIC configuration.
3. SOC rail configuration.
4. DRAM related rail configuration.
5. DRAM initialization.
6. CPU rail configuration.
7. Loading CPU related microcode and enabling CPUs.
8. Platform’s other I2C configuration.

The PMIC configuration files are kept in the 
`hardware/nvidia/platform/t23x/<platform>/bct/` directory.

**NEW DTS format example of PMIC configuration file**

```dts
/dts-v1/;
/
{
    pmic {
        system {
            block@0 {
                controller-id = <4>;
                slave-addr = <0x78>; // 7Bit:0x3c
                reg-data-size = <8>;
                reg-addr-size = <8>;
                block-delay = <10>;
                i2c-update-verify = <1>; //update and verify
                commands {
                    cpu-rail cmds {
                        command@0 {
                            reg-addr = <0x50>;
                            mask = <0x0C0>;
                            value = <0x00>;
                        },
                        command@1 {
                            reg-addr = <0x51>;
                            mask = <0x0C0>;
                            value = <0x00>;
                        },
                        command@2 {
                            reg-addr = <0x4A>;
                            mask = <0x0C0>;
                            value = <0x00>;
                        },
                        command@3 {
                            reg-addr = <0x4B>;
                            mask = <0x0C0>;
                            value = <0x00>;
                        },
                        command@4 {
                            reg-addr = <0x4C>;
                            mask = <0x0C0>;
                            value = <0x00>;
                        }
                }
            }
        }
    }
}
```

PMIC Configuration

```c
mask = <0xC0>;
value = <0x00>;
}

gpio07-cmds {
    command@0 {
        reg-addr = <0xAA>;
        mask = <0xBB>;
        value = <0xCC>;
    }
    command@1 {
        reg-addr = <0xDD>;
        mask = <0xEE>;
        value = <0xFF>;
    }
}
misc-cmds {
    command@0 {
        reg-addr = <0x53>;
        mask = <0x38>;
        value = <0x00>;
    }
    command@1 {
        reg-addr = <0x55>;
        mask = <0x38>;
        value = <0x10>;
    }
    command@2 {
        reg-addr = <0x41>;
        mask = <0x1C>;
        value = <0x1C>;
    }
}
}
core {
    block@0 {
        pwm;
        controller-id = <6>;
        source-frq-hz = <204000000>;
        period-ns = <1255>;
        min-microvolts = <400000>;
        max-microvolts = <1200000>;
        init-microvolts = <850000>; enable;
    }
    block@1 {
        mmio;
        block-delay = <3000>;
        commands {
            command@0 {
                reg-addr = <0x02434080>;
                mask = <0x10>;
                value = <0x0>;
            }
        }
    }
}
```
cpu@0 {
    block@0 {
        pwm;
        controller-id = <5>;
        source-frq-hz = <204000000>;
        period-ns = <1255>;
        min-microvolts = <400000>;
        max-microvolts = <1200000>;
        init-microvolts = <800000>; enable;
    };
    block@1 {
        mmio;
        block-delay = <3>;
        commands {
            commands@0 {
                reg-addr = <0x02214e00>;
                mask = <0x3>;
                value = <0x00000003>;
            };
            command@1 {
                reg-addr = <0x02214e0c>;
                mask = <0x1>;
                value = <0x00000000>;
            };
            command@2 {
                reg-addr = <0x02214e10>;
                mask = <0x1>;
                value = <0x00000001>;
            };
            command@3 {
                reg-addr = <0x02446008>;
                mask = <0x400>;
                value = <0x00000000>;
            };
            command@4 {
                reg-addr = <0x02446008>;
                mask = <0x10>;
                value = <0x00000000>;
            };
        };
    };
    block@2 {
        mmio;
        commands {
            command@0 {
                reg-addr = <0x02434098>;
                mask = <0x10>;
                value = <0x00>; }
        };
    };
} platform {
    block@0 {
        i2c-controller; controller-id = <1>;
        slave-addr = <0x40>;
        reg-data-size = <8>;
    };
}
PMIC Configuration

reg-addr-size = <8>;
block-delay = <10>;
i2c-update-verify = <1>;

commands {
    command@0 {
        reg-addr = <0x03>;
        mask = <0x30>;
        value = <0x00>;
    }
    command@1 {
        reg-addr = <0x01>;
        mask = <0x30>;
        value = <0x20>;
    }
}

/dts-v1/
/
{
    pmic {
        system {
            block@0 {
                controller-id = <4>;
                slave-addr = <0x78>; // 7BIt:0x3c reg-data-size = <8>;
                reg-addr-size = <8>;
                block-delay = <10>;
                i2c-update-verify = <1>; // update and verify commands {
                    cpu-rail-cmds {
                        command@0 {
                            reg-addr = <0x50>;
                            mask = <0xC0>;
                            value = <0x00>;
                        }
                        command@1 {
                            reg-addr = <0x51>;
                            mask = <0xC0>;
                            value = <0x00>;
                        }
                        command@2 {
                            reg-addr = <0x4A>;
                            mask = <0xC0>;
                            value = <0x00>;
                        }
                        command@3 {
                            reg-addr = <0x4B>;
                            mask = <0xC0>;
                            value = <0x00>;
                        }
                        command@4 {
                            reg-addr = <0x4C>;
                            mask = <0xC0>;
                            value = <0x00>;
                        }
                    }
                }
            }
        }
    }
}
PMIC Configuration

};
};
gpio07-cmds {
    command@0 {
        reg-addr = <0xAA>;
        mask = <0xBB>;
        value = <0xCC>;
    };
    command@1 {
        reg-addr = <0xDD>;
        mask = <0xEE>;
        value = <0xFF>;
    };
};
misc-cmds {
    command@0 {
        reg-addr = <0x53>;
        mask = <0x38>;
        value = <0x00>;
    };
    command@1 {
        reg-addr = <0x55>;
        mask = <0x38>;
        value = <0x10>;
    };
    command@2 {
        reg-addr = <0x41>;
        mask = <0x1C>;
        value = <0x1C>;
    };
};
}
}
core {
    block@0 {
        pwm;
        controller-id = <6>;
        source-frq-hz = <204000000>;
        period-ns = <1255>;
        min-microvolts = <400000>;
        max-microvolts = <1200000>;
        init-microvolts = <850000>; enable;
    };
    block@1 {
        mmio;
        block-delay = <3000>;
        commands {
            command@0 {
                reg-addr = <0x02434080>;
                mask = <0x10>;
                value = <0x00>;
            };
        };
    };
};
cpu@0 {
    block@0 {
        pwm;
        controller-id = <5>;
        source-frq-hz = <204000000>;
        period-ns = <1255>;
        min-microvolts = <400000>;
        max-microvolts = <1200000>;
        init-microvolts = <800000> ; enable;
    };
    block@1 {
        mmio;
        block-delay = <3> ; commands {
            commands@0 {
                reg-addr = <0x02214e00>;
                mask = <0x3> ;
                value = <0x00000003> ;
            };
            command@1 {
                reg-addr = <0x02214e0c>;
                mask = <0x1> ;
                value = <0x00000000> ;
            };
            command@2 {
                reg-addr = <0x02214e10> ;
                mask = <0x1> ;
                value = <0x00000001> ;
            };
            command@3 {
                reg-addr = <0x02446008> ;
                mask = <0x400> ;
                value = <0x00000000> ;
            };
            command@4 {
                reg-addr = <0x02446008> ;
                mask = <0x10> ;
                value = <0x00000000> ;
            };
        };
    };
    block@2 {
        mmio;
        commands {
            command@0 {
                reg-addr = <0x02434098> ;
                mask = <0x10> ;
                value = <0x00> ;
            };
        };
    };
    platform {
        block@0 {
            i2c-controller;
            controller-id = <1> ;
            slave-addr = <0x40> ;
            reg-data-size = <8> ;
        };
    }
}
reg-addr-size = <8>;
block-delay = <10>;
i2c-update-verify = <1>;
commands {
    command@0 {
        reg-addr = <0x03>;
        mask = <0x30>;
        value = <0x00>;
    }
    command@1 {
        reg-addr = <0x01>;
        mask = <0x30>;
        value = <0x20>;
    }
};

OLD CFG format

//////////////////////////////////////////////////// System Configurations
////////////////////////////////////////////////////////
// PMIC FPS to turn SD4 (VDD_DDR_1V1) on in time slot 0
// PMIC FPS to set GPIO2 (EN_DDR_VDDQ) high in time slot 1
// Set SLPEN = 1 and CLRSE on
POR reset
pmic.system.block[0].type = 1; //I2C
pmic.system.block[0].controller-id = 4;
pmic.system.block[0].slave-add = 0x78; // 7BIt:0x3c
pmic.system.block[0].reg-data-size = 8;
pmic.system.block[0].reg-add-size = 8;
pmic.system.block[0].block-delay = 10;
pmic.system.block[0].i2c-update-verify = 1; //update and verify
pmic.system.block[0].commands[0].0x53.0x38 = 0x00; // SD4 FPS UP
slot 0 pmic.system.block[0].commands[1].0x55.0x38 = 0x10; // GPIO2 FPS UP slot 2
pmic.system.block[0].commands[2].0x41.0x1C = 0x1C; // SLPEN=1, CLRSE = 11

// PMIC FPS programming to reassign SD1, SD2, LDO4 and LDO5 to
// FPS0 to leave those rails
on in SC7
pmic.system.block[1].type = 1; //I2C
pmic.system.block[1].controller-id = 4;
pmic.system.block[1].slave-add = 0x78; // 7BIt:0x3c
pmic.system.block[1].reg-data-size = 8;
pmic.system.block[1].reg-add-size = 8;
pmic.system.block[1].block-delay = 10;
pmic.system.block[1].i2c-update-verify = 1; //update and verify
pmic.system.block[1].commands[0].0x50.0xC0 = 0x00; // SD1 FPS to
PMIC Configuration

FPS0 pmic.system.block[1].commands[1].0x51.0xC0 = 0x00;  // SD2
FPS to FPS0 pmic.system.block[1].commands[2].0x4A.0xC0 = 0x00;  // LDO4 FPS to FPS0 pmic.system.block[1].commands[3].0x4B.0xC0 = 0x00;  // LDO5 FPS to FPS0 pmic.system.block[1].commands[4].0x4C.0xC0 = 0x00;  // LDO6 FPS to FPS0

// VDDIO_DDR to 1.1V, SD4 to 1.1V pmic.system.block[2].type = 1;  //I2C pmic.system.block[2].controller-id = 4;
pmic.system.block[2].slave-add = 0x78;  // 7Bit:0x3c pmic.system.block[2].reg-data-size = 8;
pmic.system.block[2].reg-add-size = 8;
pmic.system.block[2].block-delay = 10;
pmic.system.block[2].i2c-update-verify = 1;  //update and verify pmic.system.block[2].commands[0].0x1A.0xFF = 0x28;  // SD4 to 1.1V

// CORE(SOC) RAIL Configurations

// 1. Set 850mV voltage.

pmic.core.block[0].type = 2;  // PWM Type pmic.core.block[0].controller-id = 6;  //SOC_GPIO10: PWM7 pmic.core.block[0].source-frq-hz = 204000000;  //204MHz pmic.core.block[0].period-ns = 1255;  // 800KHz.
pmic.core.block[0].min-microvolts = 400000;
pmic.core.block[0].max-microvolts = 1200000;
pmic.core.block[0].init-microvolts = 850000;
pmic.core.block[0].enable = 1;

// 2. Make soc_gpio10 pin in non-tristate
pmic.core.block[1].type = 0;  // MMIO TYPE pmic.core.block[1].block-delay = 3000;
pmic.core.block[1].commands[0].0x02434080.0x10 = 0x0;  // soc_gpio10: tristate (b4) = 0

// CPU0 RAIL configurations

// 1. Set 800mV voltage.

pmic.cpu0.block[0].type
e = 2; // PWM Type
pmic.cpu0.block[0].controller-id = 5; //soc_gpio13; PWM6
pmic.cpu0.block[0].source-frq-hz = 204000000; //204MHz
pmic.cpu0.block[0].period-ns = 1255; // 800KHz.
pmic.cpu0.block[0].min-microvolts = 400000;
pmic.cpu0.block[0].max-microvolts = 1200000;
pmic.cpu0.block[0].init-microvolts = 800000;
pmic.cpu0.block[0].enable = 1;

// 2. CPU PWR_REQ
cpu_pwr_req_0_pb0 to be
1
pmic.cpu0.block[1].type = 0; // MMIO TYPE
pmic.cpu0.block[1].bloc
k-delay = 3;

pmic.cpu0.block[1].commands[0].0x02214e00.0x3 = 0x00000003; // CONFIG B0
pmic.cpu0.block[1].commands[1].0x02214e0c.0x1 = 0x00000000; // CONTROL B0
pmic.cpu0.block[1].commands[2].0x02214e10.0x1 = 0x00000001; // OUTPUT B0
pmic.cpu0.block[1].commands[3].0x02446008.0x400 = 0x00000000; //
cpu_pwr_req_0_pb0 to GPIO mode
pmic.cpu0.block[1].commands[4].0x02446008.0x10 = 0x00000000; //
cpu_pwr_req_0_pb0 tristate (b4) = 0

// 3. Set soc_gpio13 to
untristate
pmic.cpu0.block[2].type = 0; //
MMIO Type
pmic.cpu0.block[2].commands[4].0x02434098.0x10 = 0x00; // soc_gpio13 to be
untristate

/////////////////////////////////////////////////////////////////////////////////////////// Platform Configurations
/////////////////////////////////////////////////////////////////////////////////////////// // Configure pin4/pin5 as output of gpio expander (0x40)
// Configure pin4 low and pin5 high
of gpio expander (0x40)
pmic.platform.block[0].type = 1;
//I2C
pmic.platform.block[0].controller-
id = 1; //gen2
pmic.platform.block[0].slave-add = 0x40; // 7Bit:0x20
pmic.platform.block[0].reg-data-
size = 8;

pmic.platform.block[0].reg-add-size = 8;
pmic.platform.block[0].block-delay = 10;

pmic.platform.block[0].i2c-update-verify = 1; //update and verify
pmic.platform.block[0].commands[0].0x03.0x30 = 0x00; // Configure pin4/pin5
as output pmic.platform.block[0].commands[1].0x01.0x30 = 0x20; // Configure
pin4 low and pin5 high
Chapter 7. Storage Device Configuration

The Storage Device configuration file contains the platform-specific settings for storage devices in the MB1/MB2 stages.

The DTS Configuration file is of the following form:

```
/ {
   device {
      <storage_device>@instance-#  {
         <parameter> = <value>;
      };
   };
}
```

where:

- `<storage-device>` is the storage device controller (`qspiflash/ufs/sdmmc/sata`).
- `<instance-#>` is the instance of the storage controller
- `<parameter>` is controller-specific parameter as shown below.

### 7.1 QSPI Flash Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
</table>
| clock-source-id        | QSPI controller Clock Source 1: CLK_M 3: PLLP_OUT0
|                        | 4: PLLM_OUT0 5: PLLC_OUT0 6: PLLC4_MUXED        |
| clock-source-frequency | Frequency of clock source (in Hz)                |
| interface-frequency    | QSPI controller frequency (in Hz)                |
| enable-ddr-mode        | 0: QSPI SDR mode 1: QSPI DDR mode                |
### Storage Device Configuration

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>maximum-bus-width</td>
<td>Maximum QSPI bus width 0: QSPI x1 lane 2: QSPI x4 lane</td>
</tr>
<tr>
<td>fifo-access-mode</td>
<td>0: PIO mode 1: DMA mode</td>
</tr>
<tr>
<td>ready-dummy-cycle</td>
<td>No. of dummy cycles as per QSPI flash</td>
</tr>
<tr>
<td>trimmer1-val</td>
<td>TX trimmer value</td>
</tr>
<tr>
<td>trimmer2-val</td>
<td>RX trimmer value</td>
</tr>
</tbody>
</table>

#### SDMMC Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>clock-source-id</td>
<td>SDMMC controller Clock Source 0: PLLP_OUT0 1: PLLC4_OUT2_LJ 2: PLLC4_OUT0_LJ</td>
</tr>
<tr>
<td></td>
<td>3: PLLC4_OUT2 4: PLLC4_OUT1 5: PLLC4_OUT1_LJ 6: CLK_M 7: PLLC4_VCO</td>
</tr>
<tr>
<td>clock-source-frequency</td>
<td>Frequency of clock source (in Hz)</td>
</tr>
<tr>
<td>best-mode</td>
<td>Highest supported mode of operation 0: SDR26 1: DDR52 2: HS200 3: HS400</td>
</tr>
<tr>
<td>pd-offset</td>
<td>Pull-down offset</td>
</tr>
<tr>
<td>pu-offset</td>
<td>Pull-up offset</td>
</tr>
<tr>
<td>enable-strobe-hs400</td>
<td>Enable HS400 strobe</td>
</tr>
<tr>
<td>dqsl-trim-hs400</td>
<td>HS400 DQS trim value</td>
</tr>
</tbody>
</table>
## 7.3 UFS Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>max-hs-mode</td>
<td>Highest HS mode supported by UFS device 1: HS GEAR1&lt;br&gt;2: HS GEAR2&lt;br&gt;3: HS GEAR3</td>
</tr>
<tr>
<td>max-pwm-mode</td>
<td>Highest PWM mode supported by UFS device 1: PWM GEAR1&lt;br&gt;2: PWM GEAR2&lt;br&gt;3: PWM GEAR3&lt;br&gt;4: PWM GEAR4</td>
</tr>
<tr>
<td>max-active-lanes</td>
<td>Maximum number of UFS lanes [1-2]</td>
</tr>
<tr>
<td>page-align-size</td>
<td>Alignment of pages used for UFS data structures [in bytes]</td>
</tr>
<tr>
<td>enable-hs-mode</td>
<td>Whether to enable UFS HS modes 0: disable&lt;br&gt;1: enable</td>
</tr>
<tr>
<td>enable-fast-auto-mode</td>
<td>Enable fast auto mode 0: disable&lt;br&gt;1: enable</td>
</tr>
<tr>
<td>enable-hs-rate-a</td>
<td>Enable HS rate A 0: disable&lt;br&gt;1: enable</td>
</tr>
</tbody>
</table>
### Table 7.4. SATA Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
</table>
| transfer-speed | 0: GEN1  
1: GEN2 |

The storage device configuration file are kept in the `hardware/nvidia/platform/t23x/<platform>/bct/` directory.

**NEW DTS format example of storage device configuration file:**

```
/dts-v1/;

#include <defines.h>

/ {
  device {
    qspiflash@0 {
      clock-source-id = <PLLC_MUXED>;
      clock-source-frequency = <13000000>;
      interface-frequency = <13000000>;
      enable-ddr-mode;
      maximum-bus-width = <QSPI_4_LANE>;
      fifo-access-mode = <DMA_MODE>;
    }
  }
```

---

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
</table>
| enable-hs-rate-b| Enable HS rate B 0: disable  
1: enable |
| init-state      | Initial state of UFS device at MB1 entry 0: UFS not initialized by BootROM  
1: UFS is initialized by BootROM [MB1 can skip certain steps] |
read-dummy-
cycle = <8>;
trimmer1-val = <0>;
trimmer2-val = <0>;
};
sdmmc@3 {
  clock-source-id
  = <PLLC4_OUT2>;
  clock-source-
  frequency =
  <52000000>;
  best-mode =
  <HS400>;
  pd-offset = <0>;
  pu-offset = <0>;
  //enable-strobe-hs400; This property is not
  there means it is disabled dqs-trim-hs400 =
  <0>;
};
ufs@0 {
  max-hs-
  mode =
  <HS_GEAR_3
  >; max-
  pwm-mode =
  <PWM_GEAR_
  >4>; max-
  active-
  lanes =
  <2>;
  page-
  align-
  size =
  <4096>;
  enable-
  hs-
  mode;
  //enabl
  e-fast-
  auto-
  mode;
  enable-
  hs-
  rate-b;
  //enable-hs-rate-a = <0>;
  init-state = <0>;
};
};
OLD CFG format

// QSPI flash 0
device.qspiflash.0.clock-source-id = 6;
device.qspiflash.0.clock-source-frequency = 13000000;
device.qspiflash.0.interface-frequency = 13000000;
device.qspiflash.0.enable-ddr-mode = 0;
device.qspiflash.0.maximum-bus-width = 2;
device.qspiflash.0.fifo-access-mode = 1;
device.qspiflash.0.read-dummy-cycle = 8;
device.qspiflash.0.trimmer1-val = 0;
device.qspiflash.0.trimmer2-val = 0;

// Sdmmc 3
device.sdmmc.3.clock-source-id = 3; //PLLP_OUT0
device.sdmmc.3.clock-source-frequency = 52000000;
device.sdmmc.3.best-mode = 3; //1=DDR52, 3=HS400
device.sdmmc.3.pd-offset = 0;
device.sdmmc.3.pu-offset = 0;
device.sdmmc.3.enable-strobe-hs400 = 0;
device.sdmmc.3.dqs-trim-hs400 = 0;

// Ufs 0
device.ufs.0.max-hs-mode = 3;
device.ufs.0.max-pwm-mode = 4;
device.ufs.0.max-active-lanes = 2;
device.ufs.0.page-align-size = 4096;
device.ufs.0.enable-hs-mode = 1;
device.ufs.0.enable-fast-auto-mode = 0;
device.ufs.0.enable-hs-rate-b = 1;
device.ufs.0.enable-hs-rate-a = 0;
device.ufs.0.init-state = 0;
Chapter 8. UPHY Lane Configuration

UPHY lanes can be configured to be owned by various IPs such as XUSB, NVME, MPH, PCIE, NVLINK, and so on. MB1 supports NVME, UFS as boot devices for the UPHY lanes that need to be configured to access the storage in MB1 and MB2. This file defines the UPHY lane configuration that is necessary for MB1.

In T23x, BPMP-FW is loaded by MB1 and MB2 relies on BPMP-FW for UPHY configuration. Each entry in the configuration file is of the form:

Each entry in the configuration file is of the following form:

```plaintext
/ {
  uphy-lane {
    <instance-type> {
      lane-owner-map = < <id> <owner-id> >,
      < <id> <owner-id> >;
    }
  }
};
}
```

Where:

- `<instance-type>` is the type of UPHY which needs to be configured, it can be hsio or nvhs
- `<uphy-component>` is either lane or pll which needs to be configured
- `<id>` is the lane/pll number which needs to be configured
- `<owner-id>` is the unique id of the owner to which lane/pll will be assigned

The UPHY lane configurations are kept in the `hardware/nvidia/platform/t23x/<platform>/bct/` directory.
NEW DTS example uphy lane DTS configuration file and old CFG file format:

```
/dts-v1/;
/
uphy-lane {
    hsio {
        lane-owner-map = <10 2>,
        <11 1>;
    };
};
```

OLD CFG format

```
//UPHY
uphy-lane.major = 1;
uphy-lane.minor = 0;
uphy-lane.hsio.lane.10 = 2;
uphy-lane.hsio.lane.11 = 1;
```
Chapter 9. OEM FW Ratchet Configuration

Roll-back prevention for oem-fw is controlled through the OEM-FW Ratchet configuration. Ratchetting is when older version of software is precluded from loading. The ratchet version of a software is incremented after fixing the security bugs, and this version is compared to the version stored in the Boot Component Header (BCH) of the software before loading. This file defines the minimum ratchet level for OEM-FW components. If the version in BCH is lower than the minimum ratchet level in BCT, the binary/firmware will not be loaded.

Each entry in the config file is of the following form:

```
/dts-v1/;

/ {
  ratchet {
    <loader_name1> {
      <fw_name1> = < <fw_index1> <ratchet_value> >;
      <fw_name2> = < <fw_index2> <ratchet_value> >;
    };
    <loader_name2> {
      <fw_name3> = < <fw_index3> <ratchet_value> >;
    };
  }
};
```

Where:

- `<fw_index#>` is the unique index for each oem-fw.
- `<loader_name#>` is the name of the Boot Stage binary, which loads the firmware that corresponds to fw_index.
- `<fw_name#>` is the name of the firmware.
- `<ratchet_value>` is the ratchet_value for the firmware.
The ratchet configuration file is in the
hardware/nvidia/platform/t23x/<platform>/bct/ratchet directory.

**NEW DTS example**

```bash
/dts-v1/;

/ {
    ratchet {
        mb1 {
            mb1bct = <1  3>;
            spefw = <2  0>;
        };
        mb2 {
            cpubl = <11  5>;
        };
    };
}
```

**OLD CFG format**

```bash
//ratchet
ratchet.1.
mb1.mb1bct = 3;
ratchet.2.mb1.spefw = 0;
ratchet.11.mb2.cpubl = 5;
```
Chapter 10. BootROM Reset PMIC Configuration

For some T23x platforms, in L1 and L2 reset boot paths, BootROM might be required to bring PMIC rails to OTP values. This process is completed by issuing I2C commands, which are encoded in AO scratch registers by MB1 and are based on the BootROM reset configuration in MB1 BCT.

- The reset cases where the BootROM issues these commands includes:
  - Watchdog 5 expiry
  - Watchdog 4 expiry
  - SC7 exit
  - SC8 exit
  - SW-Reset
  - AO-Tag/sensor reset
  - VF Sensor reset
  - HSM reset

- Each reset case can have three sets of AO blocks of commands.
- Each AO block has multiple blocks, and each block can have multiple commands.

In the configuration file, AO blocks are specified first, and then the reset conditions are initialized by using the ID of the AO blocks.

10.1 Specifying AO Blocks

Each AO block-related line in the configuration file is of the following format:

```}
<ResetType>-<Ao-comamnd-index> = &<AoBlock-Label>
reset {
    <AoBlock-Label>: aoblock@<aoblock-index> {
        <parameter> = <value>;
        ...
        block@<block-index> {
            <parameter> = <value>;
        }
    }
};
```
```
<AoBlock-Label>: aoblock@aoblock-index { 
    ... 
} 
}; 
};
```

<table>
<thead>
<tr>
<th>Node</th>
<th>&lt;parameter&gt;</th>
<th>Description</th>
</tr>
</thead>
</table>
| `<reset-type>`-`<Ao-command-index>` | `<&AoBlock-Label>` | • `<reset-type>` specifies the reset type and must have one of the values. watchdog5, watchdog4, sc7, sc8, soft-reset, sensor-aotag, vfsensor, or hsm.  
• `<AO-command-index>` is the index of the AO command and can have values 0, 1 or 2. Each reset path can point to maximum three aoblocks  
• `<AoBblock-Label>` is the label given to one of the Ao Blocks |
| aoblock@aoblock-index | command-retries-count | Specifies the number of command attempts allowed for AO-block with `<aoblock-index>` |
| | delay-between-command-us | Specifies the delay (in microseconds), in between different commands. The delay is calculated as $1 << n$ microseconds where $n$ is provided by this parameter. |
| | wait-before-start-bus-clear-us | Specifies the wait timeout (in microseconds), before issuing the bus clear command for given AO block. The wait time is calculated as $1 << n$ microseconds where $n$ is provided by this parameter. |
| block@<block-index> | `<command-type>/td>` | `<command-type>` can only be one value - i2c-controller. That is the only one supported |
| | count | Specifies the number of commands in the block `<block-index>` |
| | i2c-controller-id | I2C controller instance |
| | slave-addr | 7-bit I2C slave address |
| | reg-data-size | Register size in bits. Valid values are 0[1-byte], 8[1-byte] and 16[2-byte], |
| | reg-addr-size | Register address size in bits. Valid values are 0[1-byte], 8[1-byte] and 16[2-byte] |
NEW DTS example of BootROM reset configuration file:

```
/dts-v1/
/
/dts-v1/
/

reset {
  // Each reset path can point to up to three aoblocks
  // This is a map of reset paths to aoblocks
  // <reset-path>-<index-pointer> = <aoblock-id>
  // index-number should be 0, 1 or 2
  // aoblock-id is the id of the one of the blocks
  // mentioned above sensor-aotag-1 = &aoblock0;
  sc7-1 = &aoblock0;

  aoblock0: aoblock0 {
    command-retries-count = <1>;
    delay-between-commands-us = <1>;
    wait-before-start-bus-clear-us = <1>;

    block0 {
      i2c-controller;
      slave-add = <0x3c>; // 7Bit:0x3c reg-data-size = <8>;
      reg-add-size = <8>;
      commands {
        command0 {
          reg-addr = <0x42>; value = <0xda>;
        };
        command1 {
          reg-addr = <0x41>;
          value = <0xf8>;
        };
      };

      // Shutdown: Set MAX77620
      // Register ONOFFCNFG2, bit SFT_RST_WK = <0>
      // Register ONOFFCNFG1, bit SFT_RST
```

<table>
<thead>
<tr>
<th>Node</th>
<th>&lt;parameter&gt;</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>commands</td>
<td>List of &lt;Address Value&gt; pairs where value to be written to the I2c slave register address &lt;reg-addr&gt; for the command indexed by &lt;command-index&gt;.</td>
</tr>
</tbody>
</table>
= <1> aoblock1: aoblock@1 {
  // Shutdown: Set MAX77620
  // Register ONOFFCNFG2, bit SFT_RST_WK = <0>
  // Register ONOFFCNFG1, bit SFT_RST = <1>
  command-retries-count = <1>
  delay-between-commands-us = <1>
  wait-before-start-bus-clear-us = <1>;
  block@0 {
    i2c-controller;
    slave-add = <0x3c>; // 7Bit:0x3c reg-data-size
    // = <8>
    reg-add-size = <8>
    commands {
      command@0 {
        reg-addr = <0x42>; value = <0x5a>;
      }
      command@1 {
        reg-addr = <0x41>; value = <0xf8>;
      }
    };
  };
};
}
// SC7 exit
// Clear PMC_IMPL_DPD_ENABLE_0[ON]=0
during SC7 exit aoblock2: aoblock@2 {
  command-retries-count = <1>
  delay-between-commands-us = <256>
  wait-before-start-bus-clear-us = <1>
  block@0 {
    mmio;
    command
    s {
      command@0 {
        reg-addr = <0x0c360010>;
        value = <0x0>
      }
    };
  };
};
OLD CFG format

/ CFG Version 1.0
// This contains the BOOTROM commands in MB1 for
multiple reset paths. reset.major = 1;
reset.minor = 0;

// Automatic power cycling: Set MAX77620
// Register ONOFFCNFG2, bit SFT_RST_WK = 1 (default is "0" after cold boot),
// Register ONOFFCNFG1, bit
SFT_RST = 1
reset.aoblock[0].command-
retries-count = 1;
reset.aoblock[0].delay-between-commands-us = 1;
reset.aoblock[0].wait-before-start-bus-clear-us = 1;
reset.aoblock[0].block[0].type = 1; // I2C
Type reset.aoblock[0].block[0].slave-add =
0x3c; // 7Bit:0x3c
reset.aoblock[0].block[0].reg-data-size =
8;
reset.aoblock[0].block[0].reg-add-size =
8;
reset.aoblock[0].block[0].commands[0].0x4
2 = 0xda;
reset.aoblock[0].block[0].commands[1].0x4
1 = 0xf8;

// Shutdown: Set MAX77620
// Register ONOFFCNFG2, bit SFT_RST_WK = 0
// Register ONOFFCNFG1, bit
SFT_RST = 1
reset.aoblock[1].command-
retries-count = 1;
reset.aoblock[1].delay-between-commands-us = 1;
reset.aoblock[1].wait-before-start-bus-clear-us = 1;
reset.aoblock[1].block[0].type = 1; // I2C
Type reset.aoblock[1].block[0].slave-add =
0x3c; // 7Bit:0x3c
reset.aoblock[1].block[0].reg-data-size =
8;
reset.aoblock[1].block[0].reg-add-size =
8;
reset.aoblock[1].block[0].commands[0].0x4
2 = 0x5a;
reset.aoblock[1].block[0].commands[1].0x4
1 = 0xf8;
// SC7 exit
// Clear PMC_IMPL_DPD_ENABLE_0[ON]=0
during SC7 exit
reset.aoblock[2].command-retries-count = 1;
reset.aoblock[2].delay-between-commands-us = 256;
reset.aoblock[2].wait-before-start-bus-clear-us = 1;

reset.aoblock[2].block[0].type = 0; // MMIO Type
reset.aoblock[2].block[0].commands[0].0x0C360010 = 0x0;

// Shutdown in sensor/ao-tag
// no commands for other
case reset.sensor-aotag.aocommand[0] = 1;
reset.sc7.aocommand[0] = 2;
Chapter 11. Miscellaneous Configuration

The different settings that do not fit into the other categories are documented in the miscellaneous configuration file.

### 11.1 MB1 Feature Fields

<table>
<thead>
<tr>
<th>Field</th>
<th>Descriptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>disable_spe</td>
<td>• 0: Enables load of SPE-FW by MB1.</td>
</tr>
<tr>
<td></td>
<td>• 1: Enables load of SPE-FW by MB1.</td>
</tr>
<tr>
<td>enable_dram_page_blacklisting</td>
<td>• 0: Disables DRAM ECC page blacklisting feature.</td>
</tr>
<tr>
<td></td>
<td>• 1: Enables DRAM ECC page blacklisting feature.</td>
</tr>
<tr>
<td>disable_sc7</td>
<td>• 0: Enables SC7-entry/exit support.</td>
</tr>
<tr>
<td></td>
<td>• 1: Enables SC7-entry/exit support.</td>
</tr>
<tr>
<td>disable_fuse_visibility</td>
<td>Certain fuses cannot be read or written by default because they are not visible.</td>
</tr>
<tr>
<td></td>
<td>• 0: Keeps the default visibility of fuses.</td>
</tr>
<tr>
<td></td>
<td>• 1: Enables visibility of such fuses.</td>
</tr>
<tr>
<td>enable_vpr_resize</td>
<td>• 0: VPR is allocated based on McVideoProtectSizeMb and McVideoProtect, WriteAccess fields of SDRAM config file.</td>
</tr>
<tr>
<td></td>
<td>• 1: VPR-resize feature is enabled (for example, no VPR carveout is allocated by MB1 and TZ write access to VPR carveout is enabled ).</td>
</tr>
<tr>
<td>l2_mss_encrypt_regeneration</td>
<td>On L2 RAMDUMP reset, regenerate MSS encryption keys for the carveouts.</td>
</tr>
<tr>
<td></td>
<td>This is a bit-field with the bit-mapping:</td>
</tr>
<tr>
<td></td>
<td>• 1: TZDRAM</td>
</tr>
<tr>
<td></td>
<td>• 2: VPR</td>
</tr>
<tr>
<td></td>
<td>• 3: GSC</td>
</tr>
<tr>
<td>se_ctx_save_tz_lock</td>
<td>Restrict SE context save and SHA_CTX_INTEGRITY operation to TZ.</td>
</tr>
<tr>
<td>disable_mb2_glitch_protection</td>
<td>Disable checks on DCLS faults, TCM parity error, TCM and cache ECC.</td>
</tr>
<tr>
<td>Field</td>
<td>Descriptions</td>
</tr>
<tr>
<td>----------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
</tbody>
</table>
| enable_dram_error_injection| • 0: Disable DRAM error injection tests  
                            • 1: Enable DRAM error injection tests                                                                                                    |
| enable_dram_staged_scrubbing| • 0: If DRAM ECC is enabled, scrub entire DRAM  
                            • 1: If DRAM ECC is enabled, scrub DRAM in stages - each BL responsible for the DRAM portions that it uses. |
| wait_for_debugger_connection | • 0: Don’t wait for debugger connection at end of MB1  
                            • 1: Spin in a while(1) loop at end of MB1 for debugger connection                                                                         |
| limit_l1_boot_client_freq   | 0: Keep boot client frequencies (BPMP, SE, CBB, etc) same for L0 and L1 reset                                                               |

### 11.1.1 Clock Data

These fields allow certain clock-related customization.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
</table>
| bpmp_cpu_nic_divide_r | Controls BPMP CPU frequency  
                          0: Skip programming
                          CLK\_SOURCE\_BPMP\_CPU\_NIC[BPMP\_CPU\_NIC\_CLK\_DI, VISOR]  
                          non-zero: 1 + Value to be programmed in
                          CLK\_SOURCE\_BPMP\_CPU\_NIC[BPMP, CPU\_NIC\_CLK\_DIVISOR] |
| bpmp_apb_divider    | Controls BPMP APB frequency  
                          • 0: Skip programming of
                          CLK\_SOURCE\_BPMP\_APB[BPMP\_APB\_CLK\_DIVISOR] non-zero:  
                          • 1 + Value to be programmed in
                          CLK\_SOURCE\_BPMP\_APB[BPMP\_APB, _CLK\_DIVISOR]       |
| axi_cbb_divider     | Controls CBB (control backbone) frequency  
                          • 0: Skip programming of CLK\_SOURCE\_AXI\_CBB[AXI\_CBB\_CLK\_DIVISOR] non-zero:  
                          • 1 + Value to be programmed in
                          CLK\_SOURCE\_AXI\_CBB[AXI\_CBB\_CLK, K\_DIVISOR]     |
| se_divider          | Controls SE (security engine) frequency  
                          • 0: Skip programming of CLK\_SOURCE\_SE[SE\_CLK\_DIVISOR] non-zero:  
                          • 1 + Value to be programmed in
                          CLK\_SOURCE\_SE[SE\_CLK\_DIVISOR]                    |
| aon_cpu_nic_divider | Controls AON/SPE CPU frequency  
                          • 0: Skip programming of
                          CLK\_SOURCE\_AON\_CPU\_NIC[AON\_CPU\_NIC\_CLK\_DI, VISOR] non-zero:  
                          • 1 + Value to be programmed in
                          CLK\_SOURCE\_AON\_CPU\_NIC[AON\_C, PU\_NIC\_CLK\_DIVISOR] |
### Miscellaneous Configuration

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aon_apb_divider</td>
<td>Controls AON APB frequency</td>
</tr>
<tr>
<td></td>
<td>• 0: Skip programming of CLK_SOURCE_AON_CPU_NIC[AON_CPU_NIC_CLK_DI←, VISOR]</td>
</tr>
<tr>
<td></td>
<td>• non-zero: 1 + Value to be programmed in CLK_SOURCE_AON_CPU_NIC[AON_C, PU_NIC_CLK_DIVISOR]</td>
</tr>
<tr>
<td>aon_can0_divider</td>
<td>Controls AON CAN1 frequency</td>
</tr>
<tr>
<td></td>
<td>• 0: Skip programming of CLK_SOURCE_CAN1[CAN1_CLK_DIVISOR]</td>
</tr>
<tr>
<td></td>
<td>• non-zero: 1 + Value to be programmed in CLK_SOURCE_CAN1[CAN1_CLK_DIVISOR]</td>
</tr>
<tr>
<td>aon_can1_divider</td>
<td>Controls AON CAN2 frequency</td>
</tr>
<tr>
<td></td>
<td>• 0: Skip programming of CLK_SOURCE_CAN2[CAN2_CLK_DIVISOR]</td>
</tr>
<tr>
<td></td>
<td>• non-zero: 1 + Value to be programmed in CLK_SOURCE_CAN2[CAN2_CLK_DIVISOR]</td>
</tr>
<tr>
<td>osc_drive_strength</td>
<td>Oscillator drive strength</td>
</tr>
<tr>
<td>pllao_divn</td>
<td>DIVN value of PLLAON</td>
</tr>
<tr>
<td></td>
<td>• 0: Use PLLAON_DIVN = 30, PLLAON_DIVM = 1, PLLAON_DIVP = 2</td>
</tr>
<tr>
<td></td>
<td>• non-zero: 1 + Value to be programmed in PLLAON_BASE[PLLAON_DIVN]</td>
</tr>
<tr>
<td>pllao_divm</td>
<td>DIVM value of PLLAON (ignored when clock.pllao_divn = 0)</td>
</tr>
<tr>
<td></td>
<td>1 + Value to be programmed in PLLAON_BASE[PLLAON_DIVM]</td>
</tr>
<tr>
<td>pllao_divp</td>
<td>DIVP value of PLLAON (ignored when clock.pllao_divn = 0)</td>
</tr>
<tr>
<td></td>
<td>1 + Value to be programmed in PLLAON_BASE[PLLAON_DIVP]</td>
</tr>
<tr>
<td>pllao_divn_frac</td>
<td>Value to be programmed</td>
</tr>
</tbody>
</table>

### 11.1.2 AST Data

MB1/MB2 uses the address-translation (AST) module to map the DRAM carveout for different firmware in the 32bit virtual address space of the various auxiliary processor clusters.

### 11.1.3 MB1 AST Data

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mb2_va</td>
<td>Virtual address for MB2 carveout in BPMP-R5 address-space.</td>
</tr>
<tr>
<td>spe_fw_va</td>
<td>Virtual address for SPE-FW carveout in AON-R5 address-space.</td>
</tr>
<tr>
<td>misc_carveout_va</td>
<td>Virtual address for MISC carveout in SCE-R5 address-space.</td>
</tr>
<tr>
<td>rcm_blob_carveout_va</td>
<td>Virtual address for RCM-blob carveout in SCE-R5 address-space.</td>
</tr>
<tr>
<td>temp_map_a_carveout_va</td>
<td>Virtual address for temporary mapping A used while loading binaries</td>
</tr>
<tr>
<td>temp_map_a_carveout_size</td>
<td>Size for temporary mapping A used while loading binaries</td>
</tr>
</tbody>
</table>
### 11.1.4 Carveout Configuration

Although "SDRAM Configuration" on page 59 has MC carveout’s preferred base, size and permissions, it does not have all information required to allocate the carveouts by MB1. This additional information is specified using miscellaneous configuration file.

For carveouts that are not protected by MC, all information, including size and preferred base address, is specified by using the miscellaneous configuration file.

Each MC carveout configuration parameter is of the following form:

```plaintext
/ {
    misc {
        carveout {
            <carveout-type> { 
                <parameter> = <value>;
            };
        };
    };
};
```

where:

- `<carveout-type>` identifies the carveout and is one of the following:

<table>
<thead>
<tr>
<th>carveout-type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>gsc[1-31]</td>
<td>GSC carveout for various purposes</td>
</tr>
<tr>
<td>mts</td>
<td>MTS/CPU-uCode carveout</td>
</tr>
<tr>
<td>vpr</td>
<td>VPR carveout</td>
</tr>
<tr>
<td>tzdram</td>
<td>TZDRAM carveout used for SecureOS</td>
</tr>
<tr>
<td>os</td>
<td>OS carveout used for loading OS kernel</td>
</tr>
<tr>
<td>rcm</td>
<td>RCM carveout used for loading RCM-blob during RCM mode [temporary boot carveout]</td>
</tr>
</tbody>
</table>
<parameter> is one of the following parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pref_base</td>
<td>Preferred base address of carveout</td>
</tr>
<tr>
<td>size</td>
<td>Size of carveout (in bytes)</td>
</tr>
<tr>
<td>alignment</td>
<td>Alignment of base address of carveout (in bytes)</td>
</tr>
<tr>
<td>ecc_protected</td>
<td>When DRAM region-based ECC is enabled and there are non-ECC protected DRAM regions, whether to allocate the carveout from ECC protected region 0: Allocate from non-ECC protected region 1: Allocate from ECC protected region</td>
</tr>
<tr>
<td>bad_page_tolernant</td>
<td>When DRAM page blacklisting is enabled, whether it is ok to have bad pages in the carveout (only possible for very large carveouts and which are handled complete by component that can avoid bad pages using SMMU/MMU) 0: No bad pages allowed for the carveout 1: Bad pages allowed for the carveout (allocation can be done without filtering bad pages)</td>
</tr>
</tbody>
</table>

The valid combination of the carveouts and their parameters are specified in the following table:

<table>
<thead>
<tr>
<th>Supported carveout-type</th>
<th>pref_base</th>
<th>size</th>
<th>alignment</th>
<th>ecc_protected</th>
<th>bad_page_tolerant</th>
</tr>
</thead>
<tbody>
<tr>
<td>gsc-[1-31], mts, vpr</td>
<td>N/A</td>
<td>N/A</td>
<td>YES</td>
<td>YES</td>
<td>YES</td>
</tr>
<tr>
<td>tzdram, mb2, cpubl, misc, os, rcm</td>
<td>YES</td>
<td>YES</td>
<td>YES</td>
<td>YES</td>
<td>YES</td>
</tr>
</tbody>
</table>

11.1.5 Coresight Data

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cfg_system_ctl</td>
<td>Value to be programmed to CORESIGHT_CFG_SYSTEM_CTL</td>
</tr>
<tr>
<td>cfg_csiee_mc_wr_ctl</td>
<td>Value to be programmed to CORESIGHT_CFG_CSITE_MC_WR_CTRL</td>
</tr>
<tr>
<td>cfg_csiee_mc_rd_ctl</td>
<td>Value to be programmed to CORESIGHT_CFG_CSITE_MC_RD_CTRL</td>
</tr>
<tr>
<td>cfg_etrmc_wr_ctl</td>
<td>Value to be programmed to CORESIGHT_CFG_ETR_MC_WR_CTRL</td>
</tr>
<tr>
<td>cfg_etrmc_rd_ctl</td>
<td>Value to be programmed to CORESIGHT_CFG_ETR_MC_RD_CTRL</td>
</tr>
<tr>
<td>cfg_csite_cbb_wr_ctl</td>
<td>Value to be programmed to CORESIGHT_CFG_CSITE_CBB_WR_CTRL</td>
</tr>
<tr>
<td>cfg_csite_cbb_rd_ctl</td>
<td>Value to be programmed to CORESIGHT_CFG_CSITE_CBB_RD_CTRL</td>
</tr>
</tbody>
</table>
11.1.6 Firmware Load and Entry Configuration

Firmware configuration is specified as follows:

```c
/{
  misc {
    ...
    ...
    firmware {
      <firmware-type> {
        <parameter> = <value>;
      }
    }
  }
};
```

Where `<firmware-type>` is one of the mb2 or tzram-el3 and `<parameter>` is specified in below table

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>load-offset</td>
<td>Offset in <code>&lt;firmware</code> carveout where <code>&lt;firmware</code> binary is loaded.</td>
</tr>
<tr>
<td>entry-offset</td>
<td>Offset of <code>&lt;firmware</code> entry point in <code>&lt;firmware</code> carveout.</td>
</tr>
</tbody>
</table>

11.1.7 CPU Configuration

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ccplex_platform_features</td>
<td>CPU platform features [should be 0]</td>
</tr>
<tr>
<td>clock_mode.clock_burst_policy</td>
<td>CPLEX clock burst policy</td>
</tr>
<tr>
<td>clock_mode.max_avfs_mode</td>
<td>Highest CPLEX AVFS mode</td>
</tr>
<tr>
<td>nafll_cfg2/fll_init</td>
<td>CPLEX NAFLL CFG2 [Fll Init]</td>
</tr>
<tr>
<td>nafll_cfg2/fll_ldmem</td>
<td>CPLEX NAFLL CFG2 [Fll Ldmem]</td>
</tr>
<tr>
<td>nafll_cfg2/fll_switch_ldmem</td>
<td>CPLEX NAFLL CFG2 [Fll Switch Ldmem]</td>
</tr>
<tr>
<td>nafll_cfg3</td>
<td>CPLEX NAFLL CFG3</td>
</tr>
<tr>
<td>nafll_ctrl1</td>
<td>CPLEX NAFLL CTRL1</td>
</tr>
<tr>
<td>nafll_ctrl2</td>
<td>CPLEX NAFLL CTRL2</td>
</tr>
<tr>
<td>lut_sw_freq_req/sw_override_ndiv</td>
<td>SW override for CPLEX LUT frequency request</td>
</tr>
<tr>
<td>lut_sw_freq_req/ndiv</td>
<td>NDIV for CPLEX LUT frequency request</td>
</tr>
<tr>
<td>lut_sw_freq_req/vfgain</td>
<td>VFGAIN for CPLEX LUT frequency request</td>
</tr>
<tr>
<td>lut_sw_freq_req/sw_override_vfgain</td>
<td>VFGAIN override for CPLEX LUT frequency request</td>
</tr>
<tr>
<td>nafll_coeff/mdiv</td>
<td>MDIV for NAFLL coefficient</td>
</tr>
<tr>
<td>nafll_coeff/pdiv</td>
<td>PDIV for NAFLL coefficient</td>
</tr>
<tr>
<td>nafll_coeff/fll_frug_main</td>
<td>FLL frug main for NAFLL coefficient</td>
</tr>
</tbody>
</table>
### Miscellaneous Configuration

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nafll_coeff/fll_fru_fast</td>
<td>FLL frug fast for NAFLL coefficient</td>
</tr>
<tr>
<td>adc_vmon.enable</td>
<td>Enable CPLEX ADC voltage monitor</td>
</tr>
<tr>
<td>min_adc_fuse_rev</td>
<td>Minimum ADC fuse revision</td>
</tr>
<tr>
<td>pllx_base/divm</td>
<td>PLLX DIVM</td>
</tr>
<tr>
<td>pllx_base/divn</td>
<td>PLLX DIVN</td>
</tr>
<tr>
<td>pllx_base/divp</td>
<td>PLLX DIVP</td>
</tr>
<tr>
<td>pllx_base/enable</td>
<td>Enable PLLX</td>
</tr>
<tr>
<td>pllx_base/bypass</td>
<td>PLLX Bypass Enable</td>
</tr>
</tbody>
</table>

### 11.1.8 Other Configuration

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aocluster.evp_reset_add</td>
<td>AON/SPE reset vector (in SPE BTCM)</td>
</tr>
<tr>
<td>carveout_alloc_direction</td>
<td>Carveout allocation direction 0:</td>
</tr>
<tr>
<td></td>
<td>End of DRAM</td>
</tr>
<tr>
<td></td>
<td>1: End of 2GB DRAM (32b address space)</td>
</tr>
<tr>
<td></td>
<td>2: Start of DRAM</td>
</tr>
<tr>
<td>se_oem_group</td>
<td>Value to be programmed to SE0_OEM_GROUP_0 / SE_RNG1_RNG1_OEM_GROUP_0 [NVHS / NVLS / Lock bits are forced set]</td>
</tr>
<tr>
<td>i2c-freq</td>
<td>List of &lt; &lt;i2c controller instance&gt; &lt;Frequency (in KHz) of I2C controller Instance &gt; where, I2C controller instance has valid values 0 to 8.</td>
</tr>
</tbody>
</table>

### 11.1.8.1 MB1 Soft Fuse Configurations

There are certain platform-specific configurations or decisions in MB1 that are required before the storage is initialized and MB1-BCT is read (for example, debug port details, certain boot-mode controls, behavior in case of failure, and so on.). To address this requirement, an array of 64 fields has been added to signed section of BR-BCT that is opaque to BR and will be consumed by MB1. In recovery mode, BR does not read BR-BCT, so this array is kept part of RCM message and is copied by BR to the location that BR would have kept it in coldboot as part of BR-BCT. This array is called MB1 software fuses (soft-fuses).
11.1.8.2 Debug Controls

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
</table>
| verbosity       | Controls verbosity of debug logs on UART (verbosity increases with increasing value):  
|                 | • 0: UART logs disabled  
|                 | • 1: Critical prints only  
|                 | • 2: Error  
|                 | • 3: Warn  
|                 | • 4: Info  
|                 | • 5: Debug  
| uart_instance   | UART controller number where debug logs are spewed:  
|                 | • 0: UARTA (only for simulation platforms)  
|                 | • 2: UARTC (open-box debug)  
|                 | • 5: UARTF (closed-box debug, USB-Type C over DP_AUX pins)  
|                 | • 7: UARTH (closed-box debug, USB-Type C over USB-OTG)  
| usb_2_nvtag      | On-chip controller connected to the USB2 pins:  
|                 | • 0: ARMJTAG  
|                 | • 1: NVJTAG  
| swd_usb_port_sel| USB2 port over which SWD should be configured:  
|                 | • 0: USB2 Port0  
|                 | • 1: USB2 Port1  
| uart8_usb_portsel| USB2 port over which UART should be configured:  
|                 | • 0: USB2 Port0  
|                 | • 1: USB2 Port1  
| wdt_enable      | Enable BPMP WDT 5th-expiry during execution of MB1/MB2 (boolean).  
| wdt_period_secs | BPMP WDT period (in secs) per expiry.  

11.1.8.3 Boot Failure Controls

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
</table>
| switch_bootchain        | Switch boot chain in case of failure (boolean)  
| reset_to_recovery       | Trigger L1 RCM reset on failure (boolean)  
| bootchain_switch_mechanism | Used if switch_bootchain is set to 1:  
|                         | • 0: Use BR-based boot-chain switching  
|                         | • 1: Use Android A/B based boot-chain switching  
| bootchain_retry_count   | Maximum number of retries for single boot-chain (0-15).  

11.1.8.4 On/Off IST Mode Controls

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>platform_detection_flow</td>
<td>In RCM mode, enter platform detection flow (boolean)</td>
</tr>
<tr>
<td>enable_tegrashell</td>
<td>In RCM mode, enter tegrashell mode (Allowed only if FUSE_SECURITY_MODE_0 is not blown) (boolean)</td>
</tr>
<tr>
<td>enable_IST</td>
<td>Enable Key ON/OFF IST boot mode (boolean).</td>
</tr>
<tr>
<td>enable_L0_IST</td>
<td>Enable Key ON (L0) IST boot mode (boolean). Used only if EnableIST is set to 1.</td>
</tr>
<tr>
<td>enable_dgpu_IST</td>
<td>Enable dGPU IST during IST boot modes (boolean).</td>
</tr>
</tbody>
</table>

11.1.8.5 Frequency Monitor Controls

These should be documented for OEMs after review with security team.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>vrefRO_calib_override</td>
<td>Override VrefRO Calibration Override based on SoftFuse instead of fuse (boolean)</td>
</tr>
<tr>
<td>vrefRO_min_rev_threshold</td>
<td>Program VrefRO frequency adjustment target based on FUSE_VREF_CALIB_0 if [FUSE_ECO_RESERVE_1[3:0] &gt; vrefRO_min_rev_threshold] (Used if vrefRO_calib_override is set to 0)</td>
</tr>
<tr>
<td>vrefRO_calib_val</td>
<td>VrefRO Calibration Value used to program VrefRO frequency adjustment target (Used if vrefRO_calib_override is set to 1)</td>
</tr>
<tr>
<td>osc_threshold_low</td>
<td>Lower Threshold of FMON counter for OSC clock</td>
</tr>
<tr>
<td>osc_threshold_high</td>
<td>Upper Threshold of FMON counter for OSC clock</td>
</tr>
<tr>
<td>pmc_threshold_low</td>
<td>Lower Threshold of FMON counter for 32K clock</td>
</tr>
<tr>
<td>pmc_threshold_high</td>
<td>Upper Threshold of FMON counter for 32K clock</td>
</tr>
</tbody>
</table>

The miscellaneous configuration files are in the bct/t23x/misc/ directory.

NEW DTS example of miscellaneous configuration file:

```dts-v1;
/
{
  misc {
    disable_spe = <0>;
    enable_vpr_resize = <0>;
    disable_sc7 = <1>;
    disable_fuse_visibility = <0>;
    disable_mb2_glitch_protection = <0>;
  }
} /
```
carveout_alloc_direction = <2>;

// Specify i2c bus speed in KHz
i2c_frequency = <4 918>;

// Soft fuse configurations
uart_instance = <2>;
emulate_prod_flow = <0>; // 1: emulate prod flow. For
pre-prod only switch_bootchain = <0>; // 1: Switch to
alternate Boot Chain on failure
bootchain_switch_mechanism = <1>; // 1: use a/b boot
reset_to_recovery = <1>; // 1: Switch to Forced Recovery on failure
platform_detection_flow = <0>; // 0: Boot in RCM flow for platform
detection
nv3p_checkSum = <1>; // 1 Check=Sum enabled for Nv3P
command
vrefRO_calib_override = <0>; // 1: program VrefRO as per soft fuse VrefRO
calibration value
vrefRO_calib_val = <0>;
osc_threshold_low =
<0x30E>;
osc_threshold_high =
<0x375>;
pmc_threshold_low =
<0x59E>;
pmc_threshold_high =
<0x64E>;
bootchain_retry_count = <0>; // Specifies the max count to retry a particular
boot chain, Max value is

cpu {
    ////////// cpu variables
    ////////// nafll_cfg3 =
    <0x38000000>; nafll_ctrl1 =
    <0x0000000C>; nafll_ctrl2 =
    <0x22250000>;
    adc_vmon.enable = <0x0>;
    min_adc_fuse_rev = <1>;
    ccplex_platform_features =
    <0x0000000>;
    clock_mode {
        clock_burst_policy = <15>;
        max_avfs_mode = <0x2E>;
    }
};

lut_sw_freq_req {
    sw_override_ndiv =
    <3>;
    ndiv = <102>;
    vfgain = <2>;
    sw_override_vfgain = <3>;
};

nafll_coeff {
    mdiv =
Miscellaneous Configuration

```
<pdiv = <0x1>
fill_frug_main = <0x9>
fill_frug_fast = <0xb>
};
nafl1 cfg2 {
  fill_init = <0xd>
  fill_ldmem = <0xc>
  fill_switch_ldmem = <0xa>
};
pllx_base {
  divm = <2>
  divn = <104>
  divp = <2>
  enable = <1>
  bypass = <0>
};
wp {
  wp {
    rails_shorted = <1>
    vsense0_cg0 = <1>
  }
};
/*******/ sw carveout variables
/*******/ carveout {
  misc {
    size = <0x800000>; // 8MB alignment = <0x800000>; // 8MB
  }
  os {
    size = <0x08000000>; //128MB
    pref_base = <0x08000000>; alignment = <0x20000000>
  }
  cpubl {
    alignment = <0x20000000>
    size = <0x040000000>; // 64MB
  }
  rcm {
    size = <0x0>
    alignment = <0>
  }
  mb2 {
    size = <0x01000000>; //16MB
  }
```
```c
alignment = <0x01000000>; //16MB
}
tzdram {
    size = <0x01000000>; //16MB
    alignment = <0x00100000>; //1MB
}

////////// mc carveout alignment
////////// vpr {
    alignment = <0x10000000>; //1GB
}
gsc@6 {
    alignment = <0x400000>; //4MB
}
gsc@7 {
    alignment = <0x100000>; //1MB
}
gsc@8 {
    alignment = <0x100000>; //1MB
}
gsc@9 {
    alignment = <0x800000>; //8MB
}
gsc@10 {
    alignment = <0x100000>; //1MB
}
gsc@12 {
    alignment = <0x100000>; //1MB
}
gsc@17 {
    alignment = <0x200000>; //2MB
}
gsc@19 {
    alignment = <0x2000000>; //2MB
}
gsc@24 {
    alignment = <0x200000>; //2MB
}
gsc@27 {
    alignment = <0x200000>; //2MB
}
gsc@28 {
    alignment = <0x200000>; //2MB
}
gsc@29 {
    alignment = <0x200000>; //2MB
}
}

/////////// mb1 ast va //////////
```
ast{
    mb2_va = <0x52000000>;
    misc_carveout_va =
        <0x70000000>;
    rcm_blob_carveout_va =
        <0x60000000>;
    temp_map_a_carveout_va =
        <0x80000000>;
    temp_map_a_carveout_size =
        <0x40000000>;
    temp_map_b_carveout_va =
        <0xc0000000>;
    temp_map_b_carveout_size =
        <0x20000000>;
};

///////////// clock variables
/////////// clock {
    pllaon_divp =
        <0x3>;
    pllaon_divn =
        <0x1F>;
    pllaon_divm =
        <0x1>;
    pllaon_divn_frac = <0x03E84000>;
    // For bpmp_cpu_nic, bpmp_apb, axi_cbb and se,
    // specify the divider with PLLP_OUT0 (408MHz) as source.
    // For aon_cpu_nic, aon_can0 and aon_can1,
    // specify the divider with PLLAON_OUT as source.
    // In both cases, BCT specified value = (1 + expected divider value).
    bpmp_cpu_nic_divider = <1>;
    bpmp_apb_divider = <1>;
    axi_cbb_divider = <1>;
    se_divider = <1>;
};

///////////// aotag variables
/////////// aotag {
    boot_temp_threshold = <97000>;
    cooldown_temp_threshold = <87000>;
    cooldown_temp_timeout = <30000>;
    enable_shutdown = <1>;
};

///////////// aocluster data
/////////// aocluster {
    evp_reset_addr = <0xc480000>;
};

OLD CFG format
disable_spe = 0;
enable_vpr_resize = 0;
disable_sc7 = 1;
disable_fuse_visibility = 0;
disable_mb2_glitch_protection = 0;
carveout_alloc_direction = 2;
/////////// cpu variables //////////
cpu.ccplex_platform_features = 0x000000;
cpu.clock_mode.clock_burst_policy = 15;
cpu.clock_mode.max_avfs_mode = 0x2E;
cpu.lut_sw_freq_req.sw_override_ndiv = 3;
cpu.lut_sw_freq_req.ndiv = 102;
cpu.lut_sw_freq_req.vfgain = 2;
cpu.lut_sw_freq_req.sw_overide_vfgain = 3;
cpu.nafll_coeff.mdiv = 3;
cpu.nafll_coeff.pdiv = 0x1;
cpu.nafll_coeff.fll_frug_main = 0x9;
cpu.nafll_coeff.fll_frug_fast = 0xb; cpu.nafll_cfg2.fll_init = 0xd; cpu.nafll_cfg2.fll_ldmem = 0xc;
cpu.nafll_cfg2.fll_switch_ldmem = 0xa; cpu.nafll_cfg3 = 0x38000000; cpu.nafll_ctrl1 = 0x0000000C;
cpu.nafll_ctrl2 = 0x22250000;
cpu.adc_vmon.enable = 0x0; cpu.pllx_base.divm = 2;
cpu.pllx_base.divn = 104;
cpu.pllx_base.divp = 2;
cpu.pllx_base.enable = 1;
cpu.min_adc_fuse_rev = 1;
wp.waypoint0.rails_shorted = 1;
wp.waypoint0.vsense0_cg0 = 1;
/////////// sw_carveout variables
/////////// carveout.misc.size = 0x8000000; // 8MB
 CARVEOUT.misc.alignment = 0x8000000; // 8MB
 carveout.os.size = 0x08000000; //128MB carveout.os.pref_base = 0x80000000;
carveout.os.alignment = 0x200000;
carveout.cpubl.alignment = 0x200000;
firmware.cpubl_load_offset = 0x600000; carveout.cpubl.size = 0x04000000; // 64MB
carveout.rcm.size = 0x0;
carveout.rcm.alignment = 0;
carveout.mb2.size = 0x01000000;
//16MB carveout.mb2.alignment =
0x01000000; //16MB

carveout.tzdram.size =
0x01000000; //16MB

carveout.tzdram.alignment = 0x00100000; //1MB

///// mc carveout alignment
///// carveout.vpr.alignment =
0x100000; carveout.gsc[6].alignment =
0x400000; carveout.gsc[7].alignment =
0x800000; carveout.gsc[8].alignment =
0x100000; carveout.gsc[9].alignment =
0x100000; carveout.gsc[10].alignment =
0x800000; carveout.gsc[12].alignment =
0x100000; carveout.gsc[17].alignment =
0x100000; carveout.gsc[19].alignment =
0x200000; carveout.gsc[24].alignment =
0x2000000; carveout.gsc[27].alignment =
0x200000; carveout.gsc[28].alignment =
0x200000; carveout.gsc[29].alignment =
0x200000;

///// mb1 ast va
///// ast.mb2_va =
0x52000000;

ast.misc_carveout_va =
0x70000000;
ast.rcm_blob_carveout_va =
0x60000000;
ast.temp_map_a_carveout_va =
0x80000000;

ast.temp_map_a_carveout_size =
0x40000000;

ast.temp_map_b_carveout_va =
0xc0000000;

ast.temp_map_b_carveout_size =
0x20000000;

///// MB2 AST VA /////


carveout.bpmp_ast_va = 0x50000000;
carveout.ape_ast_va = 0x80000000;
carveout.apr_ast_va = 0xC0000000;
carveout.sce_ast_va = 0x70000000;
carveout.rce_ast_va = 0x70000000;
carveout.camera_task_ast_va =
0x78000000;

///// clock variables
///// clock.pllaon_divp =
0x3; clock.pllaon_divn = 0x1F;
clock.pllaon_divm = 0x1;
clock.pllaon_divn_frac =
0x03E84000;
// For bpmp_cpu_nic, bpmp_apb, axi_cbb and se,
// specify the divider with PLLP_OUT0 (408MHz) as source.
// For aon_cpu_nic, aon_can0 and aon_can1,
// specify the divider with PLLAON_OUT as source.
// In both cases, BCT specified value = (1 + expected
divider value). clock.bpmp_cpu_nic_divider = 1;
clock.bpmp_apb_divider = 1;
clock.axi_cbb_divider = 1;
clock.se_divider = 1;

////////// aotag variables //////////
aotag.boot_temp_threshold = 97000;
aotag.cooldown_temp_threshold = 87000;
aotag.cooldown_temp_timeout = 30000;
aotag.enable_shutdown = 1;
//Specify i2c bus speed
in KHz i2c.4 = 918;

////////// accluster data //////////
aocluster.evp_reset_addr = 0xc480000;

////////// mb2 feature flags
////////// enable_sce = 1;
enable_rce = 1;
enable_ape = 1;
enable_combined_uart = 1;
spe_uart_instance = 2;
Chapter 12. SDRAM Configuration

DTS format for SDRAM configuration:

```
/
{
    sdram {
        mem_cfg_<N>: mem-cfg@<N> {
            <parameter> = <value>;
        };
    };
}

&mem_cfg_<N> {
    #include "<mem_override_dts >"
};
```

where

- `<N>` is number starting from 0 representing different memory configurations
- `<parameter>` is the SDRAM parameter. Usually, this corresponds to MC/EMC register.
- `<value>` is the 32bit value of the corresponding register.
- `<mem_override_dts>` is override file for SDRAM parameters which will be applied to all the configs

Format of override dts will be:

```
<parameter> = <value>; // example McVideoProtectBom = <0x00000000>;
```

Memory/MSS HW team will use a tool provided by SW to convert legacy configuration format to above DTS format.
12.1 Carveouts

The following hardware carveouts are available:

- GSC carveouts
- Non-GSC carveouts

Important carveout fields in BCT cfg.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>McGeneralizedCarveout&lt;N&gt;Bom and McGeneralizedCarveout&lt;N&gt;BomHi (where N is the GSC#)</td>
<td>Preferred base address of the GSC carveout (recommended value = 0; allows MB1 to allocate on its own)</td>
</tr>
<tr>
<td>McGeneralizedCarveout&lt;N&gt;Size128kb (where N is the GSC#)</td>
<td>31:27 - Size of the generalized security carveout region with 4kb granularity&lt;br&gt;11:0 - Size of the generalized security carveout region with 128kb granularity&lt;br&gt;size = (MC_SECURITY_CARVEOUT1_SIZE_RANGE_GSE_128KB &lt;&lt; 17) / (MC_SECURITY_CARVEOUT, T1_SIZE_RANGE_4KB &lt;&lt; 12)</td>
</tr>
<tr>
<td>McGeneralizedCarveout&lt;N&gt;Access[0-5] (where N is the GSC#)</td>
<td>Client Access Register for Generalized Carveout. For bit description details refer to TRM</td>
</tr>
<tr>
<td>McVideoProtectBom and McVideoProtectBomHi</td>
<td>Preferred base address of VPR carveout (recommended value = 0; allows MB1 to allocate on its own)</td>
</tr>
<tr>
<td>McVideoProtectSizeMb</td>
<td>Specifies VPR carveout size (in MB) (see also enable_vpr_resize in &quot;Miscellaneous Configuration&quot;)</td>
</tr>
</tbody>
</table>

12.2 GSC Carveouts

The following list gives a mapping between Carveout names and its corresponding GSC numbers:

<table>
<thead>
<tr>
<th>Carveout #</th>
<th>Carveout Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>NVDEC</td>
</tr>
<tr>
<td>2</td>
<td>WPR1</td>
</tr>
<tr>
<td>3</td>
<td>WPR2</td>
</tr>
<tr>
<td>4</td>
<td>TSECA</td>
</tr>
<tr>
<td>5</td>
<td>TSECB</td>
</tr>
<tr>
<td>6</td>
<td>BPMP</td>
</tr>
<tr>
<td>7</td>
<td>APE</td>
</tr>
<tr>
<td>8</td>
<td>SPE</td>
</tr>
<tr>
<td>9</td>
<td>SCE</td>
</tr>
<tr>
<td>10</td>
<td>APR</td>
</tr>
<tr>
<td>11</td>
<td>TZRAM</td>
</tr>
<tr>
<td>12</td>
<td>IPC_SE_TSEC</td>
</tr>
<tr>
<td>13</td>
<td>BPMP_RCE</td>
</tr>
</tbody>
</table>
### Carveout # | Carveout Name
---|---
14 | BPMP_DMCE
15 | SE_SC7
16 | BPMP_SPE
17 | RCE
18 | CPUTZ_BPMP
19 | VM_ENCRYPT1
20 | CPU_NS_BPMP
21 | OEM_SC7
22 | IPC_SE_SPE_SCE_BPMP
23 | SC7_RF
24 | CAMERA_TASK
25 | SCE_BPMP
26 | CV
27 | VM_ENCRYPT2
28 | HYPERVERSOR
29 | SMMU

#### 12.3 Non-GSC Carveouts

<table>
<thead>
<tr>
<th>Carveout #</th>
<th>Carveout Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>32</td>
<td>MTS</td>
</tr>
<tr>
<td>33</td>
<td>VPR</td>
</tr>
</tbody>
</table>
Chapter 13. GPIO Interrupt Mapping Configuration

To reduce the interrupt hunt time for GPIO pins from single ISR, in T23x, each GPO controller has 8 interrupt lines to LIC. This gives opportunity to map the GPIO pin to any of 8 interrupts. The configuration for the same is specified in the GPIO interrupt configuration file.

Each entry in this configuration file is in the following form:

```plaintext
gpio-intmap {
    port@<PORT-ID> {
        pin-<N>-int-line = <INTERRUPT-NUMBER>;
    };
    port@<PORT-ID> {
        pin-<N>-int-line = <INTERRUPT-NUMBER>;
    };
}
```

where:

- `<PORT-ID>` is the port name like A, B, C., Z, AA, BB
- `<N>` is the pin id in the port. Valid values are 0-7.
- `<INTERRUPT-NUMBER>` is interrupt route for that pin. Valid values are 0-7.

The gpio-interrupt mapping configuration file are kept at hardware/nvidia/platform/t23x/<platform>/bct/

NEW DTS example of GPIO interrupt mapping configuration file:

```plaintext
/dts-v1/;
/
    gpio-intmap {
        port@B {
            pin-1-int-line = <0>; // GPIO B1 to INTO
        };
        port@AA {
```

```plaintext
```
GPIO Interrupt Mapping Configuration

```c
pin-0-int-line = <0>; // GPIO AA0 to INT0
pin-1-int-line = <0>; // GPIO AA1 to INT0
pin-2-int-line = <0>; // GPIO AA2 to INT0
```

OLD CFG format

```c
gpio-intmap.port.B.pin.1 = 0; // GPIO B1 to INT0
gpio-intmap.port.AA.pin.0 = 0; // GPIO AA0 to INT0
gpio-intmap.port.AA.pin.1 = 0; // GPIO AA1 to INT0
gpio-intmap.port.AA.pin.2 = 0; // GPIO AA2 to INT0
```
Chapter 14. MB2 BCT Misc Configuration

14.1 MB2 Feature Fields

These feature bits are boolean flags that enable or disable certain functionality in MB2.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>disable-cpu-l2ecc</td>
<td>If this property is present CPU L2 ECC id disabled. Otherwise, it is enabled.</td>
</tr>
<tr>
<td>enable-combined-uart</td>
<td>If this property is present combined uart is enabled. Otherwise, it is disabled.</td>
</tr>
<tr>
<td>spe_uart-instance</td>
<td>UART controller used for combined UART by SPE.</td>
</tr>
</tbody>
</table>

14.2 MB2 Firmware Data

Mb2 firmware configuration is specified as:

```plaintext
/ {
    mb2-misc {
        <firmware-type> {
            <parameter> = <value>;
        };
    };
};
```

Where firmware type is one of the cpubl, ape-fw, bpmp-fw, rce-fw, sce-fw, camera-task-fw, apr-fw.
<parameter> is one of the parameters from below table.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>enable</td>
<td>Enable loading of firmware from MB2, if this property is present. Otherwise disable loading of firmware from MB2.</td>
</tr>
<tr>
<td>ast-va</td>
<td>Virtual address for firmware carveout in BPMP-R5 address-space.</td>
</tr>
<tr>
<td>load-offset</td>
<td>Offset in firmware carveout where firmware binary is loaded.</td>
</tr>
<tr>
<td>entry-offset</td>
<td>Offset of firmware entry point in firmware carveout.</td>
</tr>
</tbody>
</table>

Example of an Mb2 misc DTS configuration file:

```
/dts-v1/;

/ {
  mb2-misc {
    disable-cpu-l2ecc; enable-combined-uart;
    spe-uart-instance = <0x2>;
    firmware {
      sce {
        ast-va = <0x70000000>;
     
      }
      ape {
        enable;
        ast-va = <0x80000000>;
      
      }
      rce {
        enable;
        ast-va = <0x70000000>;
      
      }
      cpubl {
        load-offset = <0x60000000>;
      
      }
      apr {
        ast-va = <0xC0000000>;
      
      }
      camera-task {
        ast-va = <0x78000000>;
      
      }
    }
  }
};
```
Chapter 15. Security Configuration

MB1 and MB2 program most of the SCRs and firewalls in T23x. The list of SCRs/firewalls, their order, and their addresses is predetermined. The values are taken from the SCR configuration file.

Each entry in this configuration file is in the following form:

```
/ {
  scr {
    reg@<index> {
      <parameter> = <value>;
    }
  }
};
```

where:

- `<index>` is the index of the SCR/firewall in the predefined list
- `<parameter>` and its `<value>` can be as follows:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>exclusion-info</td>
<td>Exclusion info is a bit field defined as follows:</td>
</tr>
<tr>
<td></td>
<td>Bit</td>
</tr>
<tr>
<td></td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>2</td>
</tr>
</tbody>
</table>

- `value` 32 bit value for the SCR register

*Note:* The values of the SCRs are programmed in increasing order of indexes and not in the order they appear in the configuration file. The scr/firewalls which are not specified in the configuration file are locked without restricting the access to the protected registers.
The scr configuration files are kept in the 
hardware/nvidia/platform/t23x/common/bct/scr directory.

NEW DTS format example of SCR config file:
/dts-v1/;
/ {
  scr {
    reg@161 {
      exclusion-info = <7>;
      value = <0x3f008080>;
    };
    reg@162 {
      exclusion-info = <4>;
      value = <0x18000303>;
    };
    reg@163 {
      exclusion-info = <4>;
      value = <0x18000303>;
    };
  };
}

OLD CFG format
scr.161.7 = 0x3f008080; //
TKE_AON_SCR_WDTSCR0_0 scr.162.4 = 0x18000303; //
DMAAPB_1_SCR_BR_INTR_SCR_0 scr.163.4 = 0x18000303; //
DMAAPB_1_SCR_BR_SCR_0
Notice

This document is provided for information purposes only and shall not be regarded as a warranty of a certain functionality, condition, or quality of a product. NVIDIA Corporation ("NVIDIA") makes no representations or warranties, expressed or implied, as to the accuracy or completeness of the information contained in this document and assumes no responsibility for any errors contained herein. NVIDIA shall have no liability for the consequences or use of such information or for any infringement of patents or other rights of third parties that may result from its use. This document is not a commitment to develop, release, or deliver any Material (defined below), code, or functionality.

NVIDIA reserves the right to make corrections, modifications, enhancements, improvements, and any other changes to this document, at any time without notice. Customer should obtain the latest relevant information before placing orders and should verify that such information is current and complete.

NVIDIA products are sold subject to the NVIDIA standard terms and conditions of sale supplied at the time of order acknowledgement, unless otherwise agreed in an individual sales agreement signed by authorized representatives of NVIDIA and customer ("Terms of Sale"). NVIDIA hereby expressly objects to applying any customer general terms and conditions with regards to the purchase of the NVIDIA product referenced in this document. No contractual obligations are formed either directly or indirectly by this document.

NVIDIA products are not designed, authorized, or warranted to be suitable for use in medical, military, aircraft, space, or life support equipment, nor in applications where failure or malfunction of the NVIDIA product can reasonably be expected to result in personal injury, death, or property or environmental damage. NVIDIA accepts no liability for inclusion and/or use of NVIDIA products in such equipment or applications and therefore such inclusion and/or use is at customer's own risk.

NVIDIA makes no representation or warranty that products based on this document will be suitable for any specified use. Testing of all parameters of each product is not necessarily performed by NVIDIA. It is customer's sole responsibility to evaluate and determine the applicability of any information contained in this document, ensure the product is suitable and fit for the application planned by customer, and perform the necessary testing for the application in order to avoid a default of the application or the product. Weaknesses in customer's product designs may affect the quality and reliability of the NVIDIA product and may result in additional or different conditions and/or requirements beyond those contained in this document. NVIDIA accepts no liability related to any default, damage, costs, or problem which may be based on or attributable to: (i) the use of the NVIDIA product in any manner that is contrary to this document or (ii) customer product designs.

No license, either expressed or implied, is granted under any NVIDIA patent right, copyright, or other NVIDIA intellectual property right under this document. Information published by NVIDIA regarding third-party products or services does not constitute a license from NVIDIA to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property rights of the third party, or a license from NVIDIA under the patents or other intellectual property rights of NVIDIA.

Reproduction of information in this document is permissible only if approved in advance by NVIDIA in writing, reproduced without alteration and in full compliance with all applicable export laws and regulations, and accompanied by all associated conditions, limitations, and notices.

THIS DOCUMENT AND ALL NVIDIA DESIGN SPECIFICATIONS, REFERENCE BOARDS, FILES, DRAWINGS, DIAGNOSTICS, LISTS, AND OTHER DOCUMENTS (TOGETHER AND SEPARATELY, "MATERIALS") ARE BEING PROVIDED "AS IS." NVIDIA MAKES NO WARRANTIES, EXPRESSED, IMPLIED, STATUTORY, OR OTHERWISE WITH RESPECT TO THE MATERIALS, AND EXPRESSLY DISCLAIMS ALL IMPLIED WARRANTIES OF NONINFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A PARTICULAR PURPOSE. TO THE EXTENT NOT PROHIBITED BY LAW, IN NO EVENT WILL NVIDIA BE LIABLE FOR ANY DAMAGES, INCLUDING WITHOUT LIMITATION ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL, PUNITIVE, OR CONSEQUENTIAL DAMAGES, HOWEVER CAUSED AND REGARDLESS OF THE THEORY OF LIABILITY, ARISING OUT OF ANY USE OF THIS DOCUMENT, EVEN IF NVIDIA HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. Notwithstanding any damages that customer might incur for any reason whatsoever, NVIDIA’s aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the Terms of Sale for the product.

Trademarks

NVIDIA, the NVIDIA logo, and High-Definition Multimedia Interface are trademarks or registered trademarks of HDMI Licensing LLC.

Arm

Arm, AMBA, and ARM Powered are registered trademarks of Arm Limited. Cortex, MPCore, and Mali are trademarks of Arm Limited. All other brands or product names are the property of their respective holders. "ARM" is used to represent ARM Holdings plc; its operating company Arm Limited; and the regional subsidiaries Arm Inc.; Arm KK; Arm Korea Limited; Arm Taiwan Limited; Arm France SAS; Arm Consulting (Shanghai) Co. Ltd.; Arm Germany GmbH; Arm Embedded Technologies Pvt. Ltd.; Arm Norway, AS, and Arm Sweden AB.

OpenCL

OpenCL is a trademark of Apple Inc. used under license to the Khronos Group Inc.

Copyright

© 2022 NVIDIA Corporation. All rights reserved.