Qualcomm Technologies, Inc.

IPQ40xx CDT Definition and Memory Configuration

Customization Guide

80-Y8950-19 Rev. J

October 5, 2016

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

Revision history

Revision	Date	Description
A	September 2015	Initial release
B	October 2015	Added SMEM Customization details in Section 3.6
C	November 2015	Updated the following: ▪ Section 2.2.1 ▪ Section 3.4 ▪ Section 3.6
D	March 2016	Updated section 3.3.1
E	March 2016	Updated section 2.2.2
F	April 2016	Added section 3.4.1
G	June 2016	Updated section 3.4.1 Added section 3.4.2
H	August 2016	Added Section 3.7
J	October 2016	Updates to Sections 2.2.1, 3.4, and 3.6

Contents

Figures

Tables

1 Introduction

1.1 Purpose

This document provides basic design overview for Configuration Data Table (CDT) and is a guide to partition table customization for an IPQ40xx-based board. It also describes the procedure to add support for new NOR or DDR devices which are not part of AVL.

Use this document as a step-by-step guide to add a new device without any source code changes.

1.2 Conventions

The function declarations, function names, type declarations, attributes, and code samples appear in a different font,

For example, `#include`.

The code variables appear in angle brackets, for example, `<number>`.

1.3 Related documents

Title	Number
Application Note: Software Configuration Data Table (CDT)	80-N3411-1
IPQ806x DDR Tuning and Flash Partition Configuration Application Note	80-Y7866-1

1.4 Acronyms and terms

Acronym or term	Definition
CDB	Configuration Data Block
NOR	Not OR (electronic logic gate)
DDR	Double data rate
AVL	Approved vendor list

2 Overview

CDT is a software table that is programmed to a Flash device on the board. It is a continuous byte array in the memory.

CDT provides platform/device-dependent data, e.g., platform ID, DDR hardware parameters. This approach reduces dependencies between hardware and software and enables configuration of the DDR controller and the DDR device based on the CDT parameters. This helps user to change the Platform ID and DDR configuration without recompiling the SBL.

2.1 General structure

The basic unit for a CDT is the CDB. Each CDB is a chunk of user-defined bytes. A CDT is essentially a table constructed by multiple CDBs and metadata (offset, size) about those CDBs.

A CDT consists of three primary sections:

- The CDT header, consisting of:
  - Magic number
  - Version number
  - Two reserved fields
- The block of data that has the offsets and sizes of the CDBs.
- Individual CDBs
  - CDB 0 – Platform ID
  - CDB 1 – DDR parameters
  - Additional CDBs – User-defined data

For each CDB there is a corresponding metadata structure, and the metadata structures are arranged in the same order as the CDBs.

Table 2-1 shows the general structure of the CDT.

Table 2-1 CDT general structure for N number of CDBs

CDT component	Attribute name	Size and type	Value	Significance
CDT header	Magic number	32 bits, constant	0x43445400 (string CDT)	Magic number represents the existence of a successfully programmed CDT
	Version number	uint16, little-endian	0x0001	CDT version If the table format changes OR If the order of data blocks changes, this version number can be incremented to keep track of the changes. CDT version is 1 as of now.
	Reserved	32 bits	0x0	Reserved for future use
	Reserved	32 bits	0x0	Reserved for future use
Metadata for CDB 0	CDB 0 offset	uint16, little-endian	Variable	Offset to the first byte of CDB 0 in CDT
	CDB 0 size	uint16, little-endian	Variable	Size of CDB 0 in bytes
Metadata for CDB 1	CDB 1 offset	uint16, little-endian	Variable	Offset to the first byte of CDB 1
	CDB 1 size	uint16, little-endian	Variable	Size of CDB 1 in bytes
Metadata for CDB N-1	Metadata rows (CDB offsets and sizes) continue until all CDBs have been entered into the CDT.			
Configuration data	CDB 0	Raw bytes	Variable	Board-specific Platform ID.
	CDB 1	Raw bytes	Variable	User-defined data based on Board-specific PCDDR3 parameters.
	Configuration data rows continue for each CDB until all CDB data rows have been entered into the CDT.			
	CDB N-1	Raw bytes	Variable	User-defined data

2.1.1 Header

Each CDT has a header that contains the following data:

- Magic number – A 4-byte value that can be used to validate the existence of the CDT; the value of the magic number is 0x43445400, which is the null-terminated ASCII string CDT
- Version number – A 2-byte value that is an unsigned and little-endian integer
- Reserved fields – Two 4-byte reserved fields that are currently not used

The following C structure represents the CDT header:

```
struct cdt_header
{
  uint32 magic;          /**< Magic number*/
  uint16 version;        /**< Version number */
  uint32 reserved1;      /**< Reserved */
  uint32 reserved2;      /**< Reserved */
/*pack the struct because the ARM compiler
  defaults at 4 bytes alignment*/
}__attribute__((packed));
```

2.1.2 Block metadata

The block metadata section comes after the CDT header (see Table 2-1). Each CDB has its own metadata structure each with two fields:

- Block offset – Defines the offset from the beginning of the CDT to the first byte of its corresponding CDB
- Block size – Defines the size of the corresponding CDB in bytes

The following C structure represents a single block metadata structure:

```
struct cdb_meta
{
  uint16 offset;
  uint16 size;
}__attribute__((packed));
```

If the size field of any block metadata structure has a zero value, the corresponding CDB does not exist.

2.1.3 CDBs

The actual CDBs appear in the last section of the CDT (see Table 2-1). Each CDB is a sequence of bytes that has the following properties:

- CDBs are user-defined.
- The CDT does not know nor care about the contents of the CDBs.
- The CDBs are packed together without any padding between them.
- The offsets and sizes of the various CDBs are contained in the first section of the CDT.
- CDBs are arranged in a fixed order and the order may change between different CDT versions.
- Since CDBs are user-defined data, each team can develop their own C structure to map to the raw CDB bytes. However, the first two CDBs have specific data requirements. Currently, the CDT version is 1:
  - CDB 0 is the platform ID information.
  - CDB 1 is the DDR parameters.

Section 2.2 provides detailed descriptions of CDB 0 and CDB 1.

2.2 CDB 0 and CDB 1 description

2.2.1 CDB 0 – platform ID

CDB 0 is fixed to be a 4-byte or 5-byte platform ID value.

- Byte 0 (the most significant byte) represents the version number. For IPQ40xx value 0x2 is used.
- Byte 1 represents the platform type.
- Byte 2 is the hardware major version number.

- Byte 3 is the hardware minor version number.
- Byte 4 is the fused platform flavor number.

Boards	Version	Platform type	Major version	Minor version	Platform subtype	Platform ID
AP.DK01.1-C1	0x2	8	1	0	0	0x0208010000
AP.DK01.1-C2	0x2	8	1	1	0	0x0208010100
AP.DK04.1-C1	0x2	8	1	0	1	0x0208010001
AP.DK04.1-C2	0x2	8	1	1	1	0x0208010101
AP.DK04.1-C3	0x2	8	1	2	1	0x0208010201
AP.DK04.1-C5	0x2	8	1	4	1	0x0208010401
AP.DK05.1-C1	0x2	8	1	0	7	0x0208010007
AP.DK06.1-C1	0x2	8	1	0	5	0x0208010005
AP.DK07.1-C1	0x2	8	1	0	6	0x0208010006
AP.DK07.1-C2	0x2	8	1	1	6	0x0208010106

2.2.2 CDB 1 – DDR parameters

CDB 1 contains PCDDR3 device-specific parameters.

NOTE:

- All parameters are unsigned int data types (4-bytes long)
- The values should be stored as decimal numbers in the xml to ensure there are no issues with endianness.

Table 2-2 CDB 1 DDR parameter attributes

Attribute name	Description	Units
version_number	Version number	Valid value 1
Magic_number	Has the value of the DDR_PARAMS_MAGIC_NUM	N/A
Checksum	Checksum of all the DDR parameters; not used as of Ver 1	N/A
num_of_device	Number of devices populated, which indicates the number of copies of DDR parameters. In case of IPQ40xx, there is only one controller; so it should be fixed to 1. Note: CDB has additional 186 Bytes of data, which is reserved and should be zero.	Valid value 1
size_of_param	Total size of parameters per CS of the DDR interface. In case of IPQ40xx this is 1.	Decimal number
Interleaved	Specifies whether bank interleaving is enabled or not.	Valid values: 0 or 1
device_name	Not used**.** Could be used by customer for Reference	N/A
manufacture_name	Not used**.** Could be used by customer for Reference	N/A
ddr_type	PCDDR3/ PCDDR3L As supported by IPQ40xx	N/A
tRFC	JEDEC timing parameter	ns value multiplied by 10

Attribute name	Description	Units
tRAS_Min	JEDEC timing parameter	ns value multiplied by 10
tRAS_Max	JEDEC timing parameter	Clock cycles
tRC	JEDEC timing parameter	ns value multiplied by 10
tREF	Not used	N/A
tREFI	JEDEC timing parameter	ns value multiplied by 10
tXSR	Not used	N/A
tXP	Not used; we use tXPDLL for going into self-refresh and coming out.	N/A
tWTR	JEDEC timing parameter	ns value multiplied by 10
tRP_AB	JEDEC timing parameter	ns value multiplied by 10
tRRD	JEDEC timing parameter	ns value multiplied by 10
tWR	JEDEC timing parameter	ns value multiplied by 10
tCKE	JEDEC timing parameter	ns value multiplied by 10
tRCD	JEDEC timing parameter	ns value multiplied by 10
tMRD	JEDEC timing parameter	Clock cycles
num_rows_cs0	Number of rows on the DDR device; to be checked with DDR device specification.	Decimal number
num_cols_cs0	Number of columns on the DDR device; to be checked with DDR device specification.	Decimal number
num_banks_cs0	Number of banks on the DDR device; to be checked with DDR device specification.	Decimal number
num_rows_cs1	Not used. Number of rows on the DDR device. It should be 0 in case of IPQ40xx.	N/A
num_cols_cs1	Not used. Number of columns on the DDR device. It should be 0 in case of IPQ40xx	N/A
num_banks_cs1	Not used. Number of banks on the DDR device. It should be 0 in case of IPQ40xx.	N/A
interface_width	Define interface width of 32, 16, or 8-bits based on the board design.	Decimal number
burst_length	Define burst length (4,8, …)	Decimal number
cas_latency	As per design recommendation this value is set as 10.	N/A
tFAW	JEDEC timing parameter	ns value multiplied by 10
tRTP	JEDEC timing parameter	ns value multiplied by 10
tZQoper	Not used	N/A
tZQCS	Not used	N/A
tXSDLL	JEDEC timing parameter	Clock cycles
tCKSRE	Not used	N/A
tCKSRX	Not used	N/A
tXPDLL	JEDEC timing parameter	ns value multiplied by 10
tAOFPD_Min	Not used	N/A
tAOFPD_Max	Not used	N/A
tMOD	JEDEC timing parameter	ns value multiplied by 10
RESERVED_0	Used for enabling the ASR bit in the MR2 register of the DDR device.	Valid values: 0 or 1

Attribute name	Description	Units
RESERVED_1	Used to program the DDRC_PHY_ODT_REG of the DDRC. Used for ODT customization.	Decimal number
RESERVED_2	Holds the value of the MR1 register of the DDR device. Used for ODT customization	Decimal number
RESERVED_3	Future use	N/A

3 Customization

The board memory topology is defined by the content of several XML files. The Qualcomm® QSDK includes several tools that convert the content of those XML files into the binary data that is included within the final flash image file.

3.1 Download packages

Qualcomm Atheros proprietary code can be downloaded from Qualcomm ChipCode™.

Refer to the *IPQ4019.ILQ.1.0 CS Release Notes* (80-Y9570-3) or latest version for instructions to download the proprietary packages that include the tools referenced in the remainder of this document.

Tools directory	**common/build/ipq/tools**
Image directory	**common/build/ipq/**

3.2 Tool directory structure

The tools directory contains required tools and the tools/config directory contains the required configuration files. To create customized images, edit the configuration files, and invoke the tools. The output images are created in the directory tools/out.

These images are combined with other images (e.g., SBLs, U-Boot, Linux) using a separate pack script and can be programmed via U-Boot.

To customize the DDR parameters and the partition table for new board type using IPQ40xx SoC, use the tools as described in the following sections.

3.3 Flash partitioning

3.3.1 NOR and NAND partition table customization

To customize the partition table, edit the XML file for the corresponding flash. The XML files for different flash configurations are available in tools directory (**common/build/ipq/tools/config**):

- NOR: nor-partition.xml
- NAND: nand-partition.xml
- NOR + NAND: nor-plus-nand-parition.xml

Add/modify new partition entry

To add or modify a new partition entry, do the following:

- Copy an existing partition entry
- Add an entry or modify the existing entry based on the following rules:
  - In the partition XML file, the size can be specified in *size_kb* option. If *size_kb* is used, the last attribute should be 0xFF.
  - If the *size_kb* option is given as 0xFFFFFFFF, it is taken as grow partition, i.e., all the remaining space is used for that particular partition. Only the last partition should be given grow partition size.
  - The partition 0:SBL1 cannot be reordered and it must always be the first entry.
- For adding multiple NAND partitions in NOR + NAND, use which_flash parameter to distinguish the entry for NOR and NAND. The "which_flash" parameter value 0 represents partition in NOR and "which_flash" value 1 represents partition in NAND. Make sure NOR and NAND partitions are grouped together with all NOR partitions at the start followed by NAND partitions.

Example

1. To add a configuration partition named CONFIG with a size of 4 MBytes and a pad size of 1 MByte (pad bytes are given only for NAND for bad block management) in the NAND, the *config/nand-partition.xml* file needs to be edited based on the following:

```
. . .
<partition>
<name length="16" type="string">0:CONFIG</name>
<size_kb length="4">4096</size_kb>
<pad_kb length="4">1024</pad_kb>
<which_flash>0</which_flash>
<attr>0xFF</attr>
<attr>0xFF</attr>
<attr>0x00</attr>
<attr>0xFF</attr>
</partition>
<partition>
. . .
```

2. To add a NAND partition in NOR + NAND, the nor-plus-nand-partition.xml file needs to be edited with which_flash set to 1 as shown below.

```
<partition>
<name length="16" type="string">0:VENDOR_DATA</name>
<size_kb length="4">128</size_kb>
<pad_kb length="2">0</pad_kb>
<which_flash length="2">1</which_flash>
<attr>0xFF</attr>
<attr>0xFF</attr>
<attr>0x00</attr>
<attr>0xFF</attr>
</partition>
```

3. After modifying the XML file, run the following commands from the tools directory.

Description	Command to rengenerate system partition	Output file
NOR BOARD	python genimg.py --partition_tool=partition_tool --mbn_gen=nand_mbn_generator.py --skip_export --image_name=NOR_IMAGES	common/build/ipq/tools/out/nor-system-partition-ipq40xx.bin
NAND BOARD	python genimg.py --partition_tool=partition_tool --mbn_gen=nand_mbn_generator.py --skip_export --image_name=NAND_IMAGES	common/build/ipq/tools/out/nand-system-partition-ipq40xx.bin
NOR + NAND BOARD	python genimg.py --partition_tool=partition_tool --mbn_gen=nand_mbn_generator.py --skip_export --image_name=NOR_PLUS_NAND_IMAGES	common/build/ipq/tools/out/norplusnand-system-partition-ipq40xx.bin

4. Copy the output file to the images directory.

NOR BOARD	cp common/build/ipq/tools/out/nor-system-partition-ipq40xx.bin common/build/ipq
NOR + NAND BOARD	cp common/build/ipq/tools/out/norplusnand-system-partition-ipq40xx.bin common/build/ipq
NAND BOARD	cp common/build/ipq/tools/out/nand-system-partition-ipq40xx.bin common/build/ipq

5. Build a single image with copied system partition as explained in section 4.

3.3.2 eMMC partition table customization

To customize the partition table, edit the **emmc-partition.xml** file available in the tools directory common/build/ipq/tools/config

Add/modify new entry

To add or modify an entry, do the following:

- Copy an existing partition entry
- Add or modify the entry based on the following rules:
  - Specify the partition name and size in KBytes, in the partition XML.
  - Type specifies the GUID of the specific partition. Type should not be changed for SBL, TZ, and APPSBL.
  - Generate a new GUID for the newly added partition. Specify the file to be flashed; if an empty partition needs to be created and leave the filename field empty.

Example

To add a configuration partition named 0: CONFIG with a size of 50 KBytes in eMMC, do the following:

1. Edit the **emmc-partition.xml** file based on the following:

```
<partition label="0:CONFIG"   size_in_kb="50" type="0CCE190E-C1E9-4CED-9E1D-590A75C5205C" bootable="false" readonly="false" filename="config.mbn"/>
```

2. After modifying the XML file, run the following command from the tools directory.

Description	Command to rengenerate GPT	Output File
EMMC BOARD	python genimg.py -–ptool=ptool.py -–msp=msp.py --skip_export --image_name=EMMC_IMAGES	common/build/ipq/tools/out/gpt_main0.bin common/build/ipq/tools/out/gpt_backup0.bin

3. Copy the output file to the images directory (**common\build\ipq)**
4. Build a single image with copied system partition as explained in section 2.

3.3.3 eMMC flash size customization

To customize the emmc flash size, edit the **config/boardconfig_premium** in case of premium build and **config/boardconfig_standard** file in case of standard build available in the tools directory (**common/build/ipq/tools/)** by doing the following:

1. Modify the **emmc_total_blocks** (sector count) as per data sheet:

```
[CB]
dirname=CB
nand_available=true
nor_available=true
emmc_available=true
spi_nand_available=true
norplusnand_available=true
norplusemmc_available=false
nand_pagesize=2048
nand_pages_per_block=64
nand_total_blocks=2048
nand_partition=nand-partition.xml
nor_pagesize=256
nor_pages_per_block=256
nor_total_blocks=512
nor_partition=nor-partition.xml
emmc_pagesize=512
emmc_blocksize=512
emmc_total_blocks=61997056
emmc_partition=emmc-partition.xml
smem_info=smem-min-cb.xml
ssd_info=none
bootconfig_info=none
```

```
nand_partition_mbn=nand-system-partition.bin
nor_partition_mbn=nor-system-partition.bin
emmc_partition_mbn=gpt_main0.bin
nand_flash_conf=nand-flash.conf
nor_flash_conf=nor-flash.conf
machid=0x8010000
norplusnand_partition=nor_and_nand_parition.xml
norplusnand_flash_conf=norplusnand-flash.conf
norplusnand_partition_mbn=norplusnand-system-partition.bin
```

2. After modifying the XML file, run the following command from the tools directory.

Description	Command to rengenerate GPT	Output file
EMMC BOARD	python genimg.py -–ptool=ptool.py -–msp=msp.py --skip_export --image_name=EMMC_IMAGES	common/build/ipq/tools/out/gpt_main0.bin common/build/ipq/tools/out/gpt_backup0.bin

3. Copy the output file to images directory (**common\build\ipq)**
4. Build a single image with copied system partition as explained in section 4.

3.4 DDR parameter customization/new device addition

The CDT XML is available in the **<common/build/ipq/tools/config>** folder.

The details of the CDT files are given below.

Boards	CDT xml
AP.DK01.1-C1	pcddr_AP.DK01.1-C1.xml
AP.DK01.1-C2	pcddr_AP.DK01.1-C2.xml
AP.DK01.1-S1	pcddr_AP.DK01.1-S1.xml
AP.DK04.1-C1	pcddr_AP.DK04.1-C1.xml
AP.DK04.1-C2	pcddr_AP.DK04.1-C2.xml
AP.DK04.1-C3	pcddr_AP.DK04.1-C3.xml
AP.DK04.1-C5	smem-AP.DK04.1-C5.xml
AP.DK04.1-S1	pcddr_AP.DK04.1-S1.xml
AP.DK05.1-C1	smem-AP.DK05.1-C1.xml
AP.DK06.1-C1	smem-AP.DK06.1-C1.xml
AP.DK07.1-C1	smem-AP.DK07.1-C1.xml
AP.DK07.1-C2	smem-AP.DK07.1-C2.xml

For DK03, the corresponding xml files of DK01 is used.

The CDT xml and corresponding DDR size is given below.

CDT xml	DDR size
pcddr_*._64M16.xml	128 MBytes
pcddr_*._128M16.xml	256 MBytes
pcddr_*._256M16.xml	512 MBytes
pcddr_*._2M256M16.xml	2 * 512 MBytes
pcddr_*..xml (Default)	256 MBytes

To customize the DDR parameters for a platform or to add support of a new DDR device, do the following:

1. Choose the appropriate XML file to be modified based on the configuration of the DDR device on the platform.
2. Modify the DDR parameters based on the values in the data sheet of the DDR device.

Example

If the board has Micron MT41K128M16JT-125 DDR3 as a single-rank configuration, edit the section CDB 1 as follows:

NOTE: The 128 Meg chip has 14 lines as row address, 10 lines as column address, and eight as the bank address.

```
<props name="num_rows_cs0" type="DALPROP_ATTR_TYPE_UINT32"> 14 </props>
<props name="num_cols_cs0" type="DALPROP_ATTR_TYPE_UINT32">  10</props>
<props name="num_banks_cs0" type="DALPROP_ATTR_TYPE_UINT32"> 8 </props>
<props name="num_rows_cs1" type="DALPROP_ATTR_TYPE_UINT32">  0 </props>
<props name="num_cols_cs1" type="DALPROP_ATTR_TYPE_UINT32">  0 </props>
<props name="num_banks_cs1" type="DALPROP_ATTR_TYPE_UINT32"> 0 </props>
<props name="interface_width" type="DALPROP_ATTR_TYPE_UINT32"> 16 </props>
```

3. Run the command from the tools directory (**common/build/ipq/tools/**)

   ```
   "python genimg.py --cdt_gen=cdt_generator.py --image_name=CDT_IMAGES"
   ```

4. Build a single image with copied system partition as explained in Section 4.

NOTE:

- The number of rows, columns, and bank should match the DDR being used and can be obtained from the DDR data sheet. If there is a mismatch in these values, it results in boot failure.
- Only DDR3 is supported using this XML file.

To change the DDR ODT setting, do the following:

1. Choose the appropriate XML file to be modified based on the configuration of the DDR device on the platform.
2. Modify the RESERVED_1 field according to the setting that is needed.

pcddr3.reserved_1 value	DDR ODT value
0xe0004444	60R
0xe0002222	80R
0xe0006666	40R

If the value is left as 0, the default value is set as 0xe0004444 i.e. 60R.

Example

Use the following configuration settings for 60R:

```
<props name="tMOD" type="DALPROP_ATTR_TYPE_UINT32"> 220 </props>
```

```
    <props name="RESERVED_0" type="DALPROP_ATTR_TYPE_UINT32"> 0 </props>
    <props name="RESERVED_1" type="DALPROP_ATTR_TYPE_UINT32"> 0xe0004444
</props>
    <props name="RESERVED_2" type="DALPROP_ATTR_TYPE_UINT32"> 0 </props>
    <props name="RESERVED_3" type="DALPROP_ATTR_TYPE_UINT32"> 0 </props>
    <!-- Start of ddr interface 1 -->
```

1. Run the command from the tools directory (**common/build/ipq/tools/**)

   `“python genimg.py --cdt_gen=cdt_generator.py --image_name=CDT_IMAGES”`

2. Build a single image with copied system partition as explained in section 4.

Reserved_0 bit description

Bits	Field Name	Description
0	ASR feature enable	0: ASR feature disable 1: ASR feature enable
1	Self refresh/Power down disable	0: Self refresh/Power down enable 1: Self refresh/Power down disable
31:2	Reserved for future use	NA

3.4.1 DDR extended temperature usage

NOTE: Refer to the DDR part datasheet for more information on the extended temperature support and guidelines.

DRAM must be refreshed externally at 2x (double refresh) when the temperature is in extended temperature range. The external refresh is attained by reducing the self-refresh period.

The self-refresh mode requires either Auto Self Refresh (ASR) or Self Refresh Temperature (SRT). IPQ40xx supports only ASR for the extended temperature. Enable ASR and reduce the refresh rate by half to enable the IPQ40xx support.

Choose the appropriate XML file to be modified based on the configuration of the DDR device on the platform.

To enable ASR, set the RESERVED_0 bit 0 as 1.

```
<props name="RESERVED_0" type="DALPROP_ATTR_TYPE_UINT32"> 1 </props>
```

To change the refresh period, modify the tREFI (Refer to the datasheet to get the appropriate value).

For example, to change the self-refresh period to 3.9 µs, modify the tREFI value as follows:

```
<props name="tREFI" type="DALPROP_ATTR_TYPE_UINT32">39000</props>
```

1. Run the command from the tools directory (common/build/ipq/tools/)

   `“python genimg.py --cdt_gen=cdt_generator.py --image_name=CDT_IMAGES”`

2. Build a single image with copied system partition as explained in Section 4.

3.4.2 Disable DDR self-refresh and power-down

To disable DDR self-refresh and power down, set the RESERVED_0 bit 1 as 1.

<props name="RESERVED_0" type="DALPROP_ATTR_TYPE_UINT32"> 2 </props>

1. Run the command from the tools directory (common/build/ipq/tools/)

   `“python genimg.py --cdt_gen=cdt_generator.py --image_name=CDT_IMAGES”`

2. Build a single image with the copied system partition as explained in Section 4.

3.5 SPI NOR device support

To add support for a new SPI NOR device, do the following:

1. Edit nor-partition.xml for NOR
   - Edit nor-plus-nand-partition.xml for NOR+NAND
   - Edit boardconfig_premium in case of premium, and boardconfig_standard in case of standard build.

   The XML files are available in tools directory <**common/build/ipq/tools/config**>

2. Update entries in partition.xml
   - Flash block size in KBytes
   - Flash density in MBytes

NOTE: Flash block size 0xFF specifies 256 KBytes.

Example 1

In this example, SPI NOR flash has a block size of 64 KBytes and density 32 MBytes.

```
<partition>
    <name length="16" type="string">0:MIBIB</name>
   <size_kb length="4">128</size_kb>
   <pad_kb length="4">0</pad_kb>
   <which_flash>0</which_flash>
   <attr>0xFF</attr>
   <!-- Specify flash block size in KB -->
   <attr>64</attr>
   <!-- Specify flash density in MB -->
   <attr>32</attr>
   <attr>0xFF</attr>
   <img_name type="string">nor-user-partition-ipq40xx.bin</img_name>
</partition>
```

Example 2

In this example, SPI NOR flash has a block size of 4 KBytes and density 2 MBytes.

```
<partition>
    <name length="16" type="string">0:MIBIB</name>
   <size_kb length="4">128</size_kb>
```

```
  <pad_kb length="4">0</pad_kb>
  <which_flash>0</which_flash>
  <attr>0xFF</attr>
  <!-- Specify flash block size in KB -->
  <attr>4</attr>
  <!-- Specify flash density in MB -->
  <attr>2</attr>
  <attr>0xFF</attr>
  <img_name type="string">nor-user-partition-ipq40xx.bin</img_name>
</partition>
```

Example 3

In this example, SPI NOR flash has a block size of 256 KBytes and density 64 MBytes.

```
<partition>
<name length="16" type="string">0:MIBIB</name>
<size_kb length="4">128</size_kb>
<pad_kb length="4">0</pad_kb>
<which_flash>0</which_flash>
<attr>0xFF</attr>
<!-- Specify flash block size in KB -->
<attr>0xFF</attr>
<!-- Specify flash density in MB -->
<attr>64</attr>
<attr>0xFF</attr>
<img_name type="string">nor-user-partition-ipq40xx.bin</img_name>
</partition>
```

3. Update the entries in boardconfig_premium/boardconfig_standard

```
nor_pagesize
nor_pages_per_block
nor_total_blocks
```

Example 1

In this example, SPI NOR flash has a block size of 64 KBytes and density 32 Mbytes.

```
nor_pagesize=256
nor_pages_per_block=256
nor_total_blocks=512
```

Example 2

In this example, SPI NOR flash has a block size of 4 KBytes and density 2 MBytes.

```
nor_pagesize=256
nor_pages_per_block=16
nor_total_blocks=512
```

Update the entries under the respective board type in addition to IPQ40xx.

4. Generate system partition.

After editing the required boardconfig and nor-partition.XML for the required flash device, run the following from the tools directory.

Description	NOR board	NOR + NAND board
Command to rengenerate system partition	python genimg.py --partition_tool=partition_tool --mbn_gen=nand_mbn_generator.py --skip_export --image_name=NOR_IMAGES	python genimg.py --partition_tool=partition_tool --mbn_gen=nand_mbn_generator.py --skip_export --image_name=NOR_PLUS_NAND_IMAGES
Output file	common/build/ipq/tools/out/nor-system-partition-ipq40xx.bin>	common/build/ipq/tools/out/norplusnand-system-partition-ipq40xx.bin

5. Copy the output file to the images directory (**common\build\ipq**)

NOR board	NOR + NAND board
cp common/build/ipq/tools/out/ nor-system-partition-ipq40xx.bin common/build/ipq cp common/build/ipq/tools/config/boardconfig common/build/ipq	cp common/build/ipq/tools/out/ norplusnand-system-partition-ipq40xx.bin common/build/ipq cp common/build/ipq/tools/config/boardconfig common/build/ipq/

6. Build a single image with the newly generated system partition as explained in section 4.

Limitations

- For flash sizes greater than 16 MBytes which fall back to default configuration, if the special commands are not supported by the new flash device added, flash operations might fail.
- Non-JEDEC flash devices are not supported for fall back.

3.6 SMEM parameter customization

The SMEM XML is available in the **<common/build/ipq/tools/config>** folder.

The details of the SMEM files are as follows:

Boards	CDT XML
AP.DK01.1-C1	smem-AP.DK01.1-C1.xml
AP.DK01.1-C2	smem-AP.DK01.1-C2.xml
AP.DK01.1-S1	smem-AP.DK01.1-S1.xml
AP.DK04.1-C1	smem-AP.DK04.1-C1.xml
AP.DK04.1-C2	smem-AP.DK04.1-C2.xml
AP.DK04.1-C3	smem-AP.DK04.1-C3.xml
AP.DK04.1-C5	pcddr_AP.DK04.1-C5.xml
AP.DK04.1-S1	smem-AP.DK04.1-S1.xml
AP.DK05.1-C1	pcddr_AP.DK05.1-C1.xml
AP.DK06.1-C1	pcddr_AP.DK06.1-C1.xml
AP.DK07.1-C1	pcddr_AP.DK07.1-C1.xml
AP.DK07.1-C2	pcddr_AP.DK07.1-C2.xml

For DK03, the corresponding xml files of DK01 is used.

To generate the SMEM partition table, do the following:

1. Choose the appropriate XML file to be modified based on the configuration of the device on the platform.
2. Modify the flash density according to the requirement by updating the SMEM_BOOT_FLASH_DENSITY field in the XML file.

Example 1

For a 16 MBytes flash size:

```
<data type="SMEM_BOOT_FLASH_TYPE">
  <flash_type>0x6</flash_type>
</data>
<data type="SMEM_BOOT_FLASH_INDEX">
  <flash_index>0x5</flash_index>
</data>
<data type="SMEM_BOOT_FLASH_CHIP_SELECT">
  <flash_chip_select>0x0</flash_chip_select>
</data>
<data type="SMEM_BOOT_FLASH_BLOCK_SIZE">
  <flash_block_size>0x10000</flash_block_size>
</data>
<data type="SMEM_BOOT_FLASH_DENSITY">
  <flash_block_size>0x1000000</flash_block_size>
</data>
```

Example 2

For a 32 MBytes flash size:

```
<data type="SMEM_BOOT_FLASH_TYPE">
    <flash_type>0x6</flash_type>
  </data>
  <data type="SMEM_BOOT_FLASH_INDEX">
    <flash_index>0x5</flash_index>
  </data>
  <data type="SMEM_BOOT_FLASH_CHIP_SELECT">
    <flash_chip_select>0x0</flash_chip_select>
  </data>
  <data type="SMEM_BOOT_FLASH_BLOCK_SIZE">
    <flash_block_size>0x10000</flash_block_size>
  </data>
  <data type="SMEM_BOOT_FLASH_DENSITY">
    <flash_block_size>0x2000000</flash_block_size>
  </data>
```

Example 3

For a 2 MBytes flash size:

```
<data type="SMEM_BOOT_FLASH_TYPE">
    <flash_type>0x6</flash_type>
  </data>
```

```
<data type="SMEM_BOOT_FLASH_INDEX">
  <flash_index>0x5</flash_index>
</data>
<data type="SMEM_BOOT_FLASH_CHIP_SELECT">
  <flash_chip_select>0x0</flash_chip_select>
</data>
<data type="SMEM_BOOT_FLASH_BLOCK_SIZE">
  <flash_block_size>0x10000</flash_block_size>
</data>
<data type="SMEM_BOOT_FLASH_DENSITY">
  <flash_block_size>0x200000</flash_block_size>
</data>
```

1. To generate the smem binary, run the following command from the **common/build/ipq/tools/** directory.

   ```
   python genimg.py --smem_gen=smem-tool.py --image_name=SMEM_IMAGES
   ```

2. The required smem binary file is now generated and present in **common/build/ipq/tools/out**

3.7 512 MByte SPI NAND support

The page size in 512 MByte SPI NAND is 4096 bytes whereas in the existing NAND, the page size is 2048 bytes. So, the boardconfig files need to be updated.

To generate a single image with the 512 MByte SPI NAND support, do the following:

1. Go to the directory: **ipq/tools/config/**.
2. Update the following parameters in boardconfig_premium and boardconfig_standard files:

   ```
   nand_pagesize=4096
   nand_total_blocks=2048
   ```

3. Copy the updated boardconfig_premium and boardconfig_standard files from **ipq/tools/config/** to **ipq/** directory.
4. Change the directory to ipq/tools/.
5. Generate system partition with the updated page size by executing the following command:

   ```
   "python genimg.py --partition_tool=partition_tool --
   mbn_gen=nand_mbn_generator.py --skip_export --
   image_name=NOR_PLUS_NAND_IMAGES"
   ```

   Generated binary file with the name **norplusnand-system-partition-ipq40xx.bin** will be in **ipq/tools/out/** directory.

6. Copy the generated norplusnand-system-partition-ipq40xx.bin file from **ipq/tools/out/** to **ipq/** directory.
7. In ipq directory, go to norplusnand-flash.conf file and change the ubi image to openwrt-ipq806x-ipq40xx-ubi-root-512MB.img.

   You can also rename the ubi image **openwrt-ipq806x-ipq40xx-ubi-root-512MB.img** to **openwrt-ipq806x-ipq40xx-ubi-root.img**.

8. Use the steps in Section 4 to create a single image with the copied system partition and boardconfig files.

4 Preparing image for flash programming

To program flash, use the tools listed in Table 4-1.

Table 4-1 Tools for programming flash

Tool	Location
pack.py	common\build\ipq\pack.py This tool combines individual binaries corresponding to each partition into a single binary image along with scripts inside.
mkimage	Standard Linux tools. For example, in Ubuntu-based system this command can be used to install the mkimage tool, "sudo apt-get install u-boot-tools"
dtc	Standard Linux tools. For example, in Ubuntu-based system this command can be used to install dtc tool: "sudo apt-get install device-tree-compiler"

1. Once the tools are in place, go to the images directory in which other images are available, e.g., **common\build\ipq**. Figure 4-1 shows the images directory.

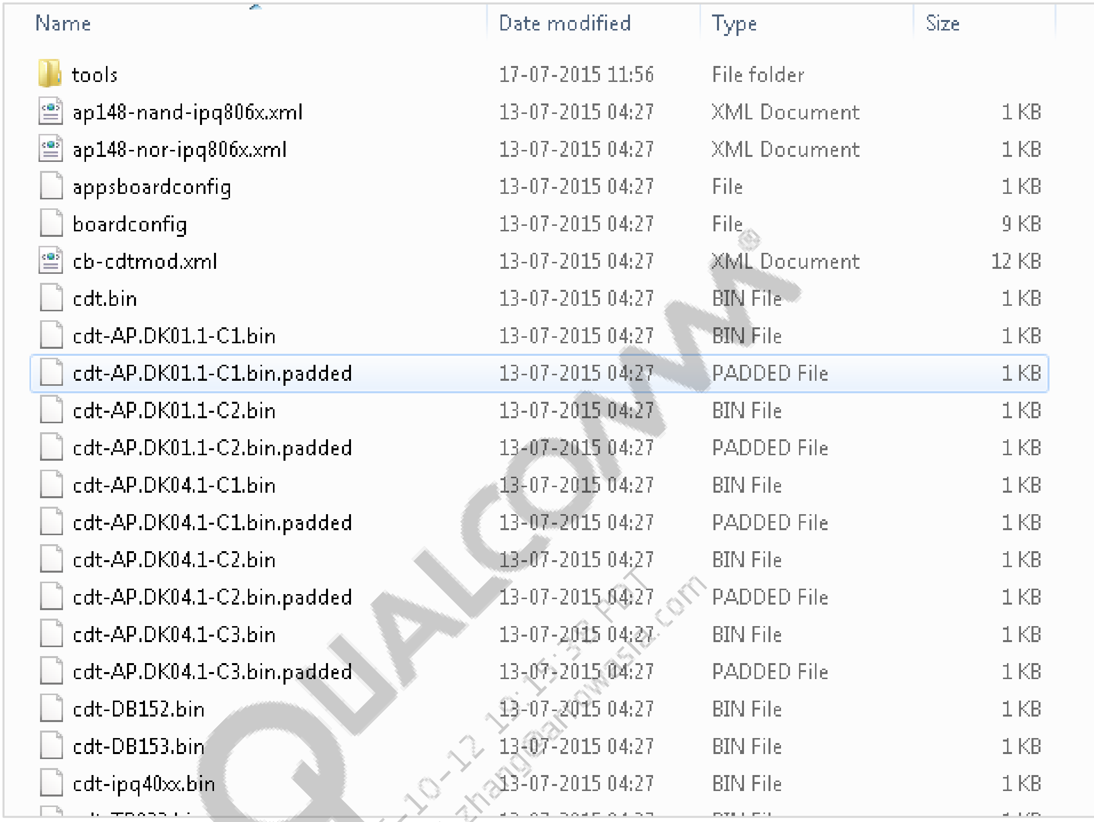

Figure 4-1 Images directory

2. Run the following commands in the images directory:

Premium profile	NOR board	python pack.py -t nor -B -F boardconfig_premium -o ipq40xx-nor.img .
	NOR + NAND board	python pack.py -t norplusnand -B -F boardconfig_premium -o ipq40xx-nornand.img .
	NAND board	python pack.py -t nand -B -F boardconfig_premium -o ipq40xx-nand.img .
	EMMC board	python pack.py -t emmc -B -F boardconfig_premium -o ipq40xx-emmc.img .
Standard profile	NOR board	python pack.py -t nor -B -F boardconfig_standard -o ipq40xx-nor.img .
	NOR + NAND board	python pack.py -t norplusnand -B -F boardconfig_standard -o ipq40xx-nornand.img .
	NAND board	python pack.py -t nand -B -F boardconfig_standard -o ipq40xx-nand.img .
	EMMC board	python pack.py -t emmc -B -F boardconfig_standard -o ipq40xx-emmc.img .

In the above commands, the last parameter '.' represents the directory that contains the image files. The output file name is mentioned after –o option in the respective commands.

5 FAQs

Question	Answer
What is the primary goal of CDT and the platform ID?	The primary goal is to have one identical software build work across different hardware platforms and form factors and their different hardware revisions and variations.
What information is stored as part of the CDT in the platform ID?	CDT contains platform-specific information. It currently holds platform ID data structure (PlatformInfoMemType) DDR device type, mode (interleaved or noninterleaved), density, and JEDEC spec default timing data; this DDR configuration may or may not exist, depending on the target
Will the hardware platform without a platform ID installed boot up? What are the consequences/risks involved?	It may or may not boot. Boot loaders default the hardware platform as unknown in this case. Also, the DDR driver defaults to the JEDEC specifications. If drivers or services packaged within the boot loaders or later in the system/HLOS/kernel have a hard dependency on the hardware platform information, then the behavior is undefined. Also, if the DDR configuration in the hardware platform is different from the built-in default, it fails to boot.
Who benefits from this platform ID?	Abstracting the DDR type and configuration information to the external storage device enables customers to support different memory vendors easily without requiring to recompile the SBL.
What are the steps to add support for a new SPI NOR to SBL?	See section 3.4.2 for the steps to add Nor flash support in SBL.
What is the command to link a single image to a specific DDR configuration file?	In the common\build\ipq> folder > nor-flash.conf file >, change the filename to point to the specific image file. **Note**: Platform ID must be the same as in the entry. For example: [ddr-AP-DK01.1-C1] partition = 0:CDT filename = cdt-AP.DK01.1-C1.bin if_machid = 0x8010000
Is it necessary to add flash device ID and other NOR flash parameter in the nor-partition XML file?	It is not necessary to add the flash device ID in the XML file. However, the device density and block size can be updated in the XML file. See section 3.4.2 for more details.
At present, the SBL1 image contains the detailed SPI NOR configuration for each type # in the Qualcomm AVL. Why not store this data in the CDT partition and provide a way to change the type # via XML file. This would enable customers to customize the SPI NOR configuration parameters to fit their board/component choice.	Only some of the parameters can be configured using Partition XML file and SPI NOR configuration parameters is not one of the values that can be customized or configured using this file.

Question	Answer
Can default parameters be used if the SPI NOR type # is not defined on the current AVL?	Yes. The default parameters can be used. See section 3.4.2 for the steps to configure SPI NOR details.
What is the GUID field used for and by whom?	GUID is used by the Qualcomm proprietary code to determine the partition instead of the partition name. **Note**: It is advisable to retain the GUID of the existing EMMC partition.
Can the value of the cdt.bin in the filename field of the emmc_partition.xml file be updated?	Only the flash configuration details can be updated in the emmc_partition XML file. None of the other details can be changed.
What is the maximum number of partitions that are specified in the emmc_partition.xml file?	Maximum number of allowed partitions is 32.
What is the procedure to get GUID to add a partition?	Use any 128-bits random number that is generated by using GUID generator or random 128-bit generator.
Is 64 M flash supported?	Yes. There is no limitation in SBL.
Does the 4 or 5 bytes value in ‘CDB0’ correspond to the U-Boot ‘machid’ environment variable?	The last 4-bytes of the platform ID are used by U-Boot as ‘machid’ to differentiate between boards.