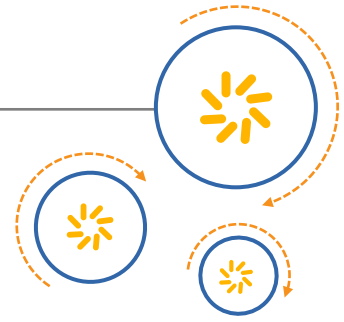




Qualcomm Technologies, Inc.



Qualcomm[®] Snapdragon[™] 600E Processor APQ8064E

DSI Programming Guide

September 2016

© 2015-2016 Qualcomm Technologies, Inc. All rights reserved

Qualcomm Snapdragon is a product of Qualcomm Technologies, Inc. Other Qualcomm products referenced herein are products of Qualcomm Technologies, Inc. or its other subsidiaries.

DragonBoard, Qualcomm, and Snapdragon are trademarks of Qualcomm Incorporated, registered in the United States and other countries. Other product and brand names may be trademarks or registered trademarks of their respective owners.

This technical data may be subject to U.S. and international export, re-export, or transfer ("export") laws. Diversion contrary to U.S. and international law is strictly prohibited.

Use of this document is subject to the license set forth in Exhibit 1.

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

Revision history

Revision	Date	Description
C	September 2016	Update to 'E' part
B	February 24, 2016	<ul style="list-style-type: none">Section 4.1 Setting up DSI panel-related GPIO reset pins: Changed mach-apq to mach-msm: Information about setting up DSI panel-related GPIO reset pins can be found at \Linux\android\kernel\arch\arm\mach-msm\board-8064-display.c.Section 5.3: Use dsi_timing_program to calculate DSI PHY register Explained how to open the DB_APQ8064_DSI_Timing_Program.xlsm file
A	June 1, 2015	Initial release

Contents

1 Introduction	5
1.1 Purpose	5
1.2 Conventions	5
1.3 Terms and acronyms	5
2 DSI Overview	7
2.1 Command and Video modes	7
2.1.1 Command mode	8
2.1.2 Video mode	8
2.2 Clock	8
2.2.1 Type of DSI clock	8
2.2.2 Clock relation requirements	9
3 Design Consideration	10
3.1 Performance consideration	10
4 MIPI DSI Driver Migration Guide	11
4.1 Setting up DSI panel-related GPIO reset pins	11
4.1.1 Vsync GPIO	11
4.1.2 Initialization GPIO high/low/high location	11
4.1.3 Example	12
4.2 DSI host initialization-related code	14
4.2.1 Initialization sequence call flow	14
4.2.2 DSI clock enable	15
4.2.3 mipi_dsi_clk_div_config() – DSI core clock and DSI PCLK	15
4.2.4 DSI PHY configuration	17
4.3 DSI client-side-related code change	18
4.3.1 Panel driver	18
4.3.2 Panel-specific initialization sequence	20
5 Panel Bring-up	21
5.1 LCD parameter fill	21
5.2 PHY table modify inside panel driver	21
5.3 Use dsi_timing_program to calculate DSI PHY register	22
6 Panel Bring-up Checkpoints	26
6.1 No DSI clock output	26
6.1.1 Check function mipi_dsi_clk_ctrl ()	26
6.1.2 Check DSI_related clocks by ADB command	26
6.1.3 Check DSI_BITCLK output signal by scope	27
6.1.4 Possible file changes	28
6.2 Tearing on DSI Command mode LCD	29
7 Unstable/Locked Bitclk Issue	30
7.1 VCO and bitclk setting formula on APQ8064E	30
7.2 Verification by scope capture	31

7.2.1 Normal waveform	31
EXHIBIT 1	32

Figures

Figure 2-1 Examples for Video and Command modes.....	7
Figure 5-1 The DSI and MDP registers worksheet.....	24
Figure 5-2 DSIPHY_TIMING_CTRL.....	25
Figure 6-1 DSI_BITCLK = 366 MHz	27
Figure 6-2 DSI_bit clock = 500 MHz	27
Figure 6-3 DSI_bit clock = 600 MHz	28
Figure 6-4 Frame package transfer vs. TE signal	29
Figure 7-1 MIPI DSI bitclk = 578 MHz normal waveform.....	31

Tables

Table 1-1 Terms and acronyms	5
Table 2-1 Clock relation requirements	9
Table 3-1 Lane configuration recommendation for various resolutions	10
Table 4-1 Parameter descriptions	20
Table 5-1 Example of bridge IC PHY settings	22

1 Introduction

1.1 Purpose

This document presents an application usage of the Display Serial Interface (DSI) panel bring-up for the Android OS for Qualcomm® Snapdragon™ 600E processor (APQ8064E). This document also provides sample code and PLL calculations regarding the DSI Mobile Industry Processor Interface (MIPI) panel bring-up.

NOTE: This document provides a description of chipset capabilities. Not all features are available, nor are all features supported in the software.

NOTE: Enabling some features may require additional licensing fees.

1.2 Conventions

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

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

Shading indicates content that has been added or changed in this revision of the document.

1.3 Terms and acronyms

[Table 1-1](#) defines terms and acronyms that may be used throughout this document.

Table 1-1 Terms and acronyms

Term	Definition
720P HD	720p, 720 progressive scan or non-interlaced horizontal lines high definition
APQ	Qualcomm® Application-only Processor
BLLP	Banking or Low-Power Interval
DCS	Display Command Set
DSI	Display Serial Interface
EMI	Electromagnetic Interference
ESC	Electronic Speed Controller

Term	Definition
fps	Frames per second
FWVGA	Full-width VGA
GPIO	General Purpose Input/Output
HFP	Horizontal-sync front porch
HSA	Horizontal Sync Active
HVGA	Half-size Video Graphics Array
IRQ	Interrupt Request
LCD	Liquid Crystal Display
LCDC	Liquid Crystal Display Controller
LCM	Liquid Crystal Module
LP	Low Power
MDP	Mobile Display Processor
MIPI	Mobile Industry Processor Interface
MMSS	Multimedia Subsystem
PLL	Phase Lock Loop
PCLK	Pixel Clock
PHY	Physical Layer
QHD	960x540 resolution
QVGA	Quarter Video Graphics Array, 320 x 240 image resolution
RAM	Random Access Memory
RGB	Red-Green-Blue
SVGA	Super Video Graphics Array (1024 x 768)
TCXO	Temperature-Compensated Crystal Oscillator
TE	Tearing Effect
VCO	Voltage-Controlled Oscillator
VE	Vertical Sync End
VGA	Video Graphics Array (640 x 480 image resolution)
VS	Vertical Sync Start
WQVGA	Wide Quarter Video Graphics Array, 320 x 240 image resolution
WSVGA	Wide Super VGA, 1024 x 576/600
WVGA	800 x 480 resolution
WXGA, WXGA+	Wide Extended Graphics Array, 1280 x 768, 1440 x 900
XGA	Extended Graphics Array, 1024 x 768

2 DSI Overview

The DSI is a specification by the MIPI and is targeted at reducing the cost of the display subsystem in mobile and embedded-computing devices. It defines a serial bus and communication protocol between the host and the device (client). The bus includes one high-speed clock lane and one or more data lanes. Each lane is carried on two wires and uses low voltage, differential signaling.

See the following:

- MIPI D-PHY Specification v01-00-00
- DSI Specification v01-02-00
- MIPI DCS Specification v01-02-00

at <http://www.mipi.org/specifications> for further details on the MIPI DSI.

2.1 Command and Video modes

There are two modes of operation for DSI-compliant peripherals, Command and Video. [Figure 2-1](#) shows examples of these two modes.

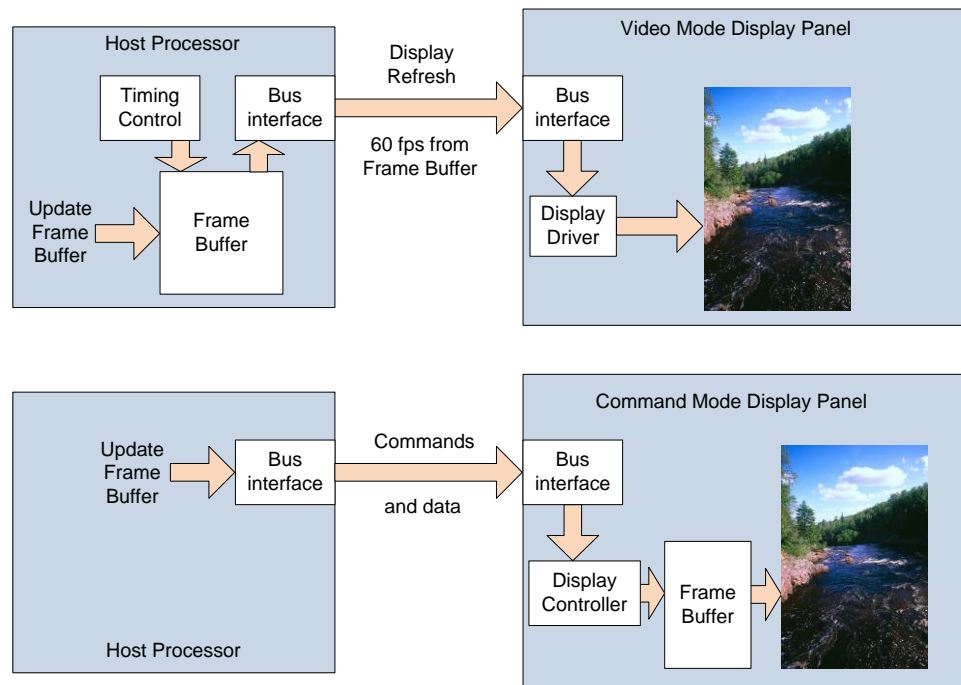


Figure 2-1 Examples for Video and Command modes

2.1.1 Command mode

Command mode refers to transactions taking the form of sending commands and data to a peripheral, that is, the LCD driver IC. This mode is typically used for the smart panel with external RAM out of the APQ and external LCDC, which can self-refresh in the case of a static image update. The APQ can go to TCXO shutdown and save more power, but there is an extra cost on the external RAM and LCDC.

The signal flow of information is bidirectional in Command mode so the host can write or read data to or from a peripheral. The host can also synchronize the flow of data using the Tearing Effect (TE) signal (Vsync) from the panel to avoid TEs.

2.1.2 Video mode

Video mode refers to transactions taking the form of a real-time pixel stream. The DSI host inside the processor must refresh the image data continuously and, typically, this is used for the dumb panel without external RAM. The host provides video data, that is, pixel values, and synchronization information, that is, Vsync, Hsync, data enable, and the pixel clock.

Video mode behavior is similar to the RGB interface but costs only a couple of pins (so there is also less EMI and radio desensing).

2.2 Clock

The DSI core receives several clocks to drive the different logic blocks. However, users can focus on the bitclk based on the panel specification. The actual clock can be calculated from the configuration, such as targeting the fps, LCD resolution, number of lanes, etc.

2.2.1 Type of DSI clock

2.2.1.1 DSI bit clock

The DSI bit clock is the DSI clock that goes out from the host processor to the LCD driver. The DSI bit clock is used as the source-synchronous bit clock for capturing the serial data bit in the receiver PHY. This clock shall be active while data is transferred.

The maximum frequency is 1 GHz.

2.2.1.2 DSI byte clock

During HS transmission, each byte of data is accompanied by a byte clock. The DSI byte clock is used in the lane management layer for HS data transmission. Like the DSI bit clock, the byte clock shall be active while data is transferred.

The maximum frequency is 125 MHz.

2.2.1.3 DSI clock

This clock is the core clock of the DSI controller.

The maximum frequency is 500 MHz.

2.2.1.4 DSI ESC clock

The DSI ESC clock source can be PXO or DSI byte clock. If it is PXO, then the source is 27 MHz. But the value of the ESC clock changes depending on the value of the divider used. It is controlled by the DSI1_ESC_NS register (0x011C).

In the latest source code, the DSI byte clock is used as the source of the ESC clock. The value of the DSI ESC clock can be changed using the `pinfo→mipi.esc_byte_ratio` parameter in the panel file. The value of this ratio should be DSI byte clock/Expected DSI ESC clk.

2.2.1.5 DSI pixel clock

The pixel clock is required to always run when transmitting data over the DSI link. However, this requirement applies only when DSI functionality is enabled. Therefore, the `clkon` signal is the output of the DSI control register to enable the functionality.

The maximum frequency is 166.67 MHz.

2.2.2 Clock relation requirements

- $H\text{-total} = \text{HorizontalActive} + \text{HorizontalFrontPorch} + \text{HorizontalBackPorch} + \text{HorizontalSyncPulse} + \text{HorizontalSyncSkew}$
- $V\text{-total} = \text{VerticalActive} + \text{VerticalFrontPorch} + \text{VerticalBackPorch} + \text{VerticalSyncPulse} + \text{VerticalSyncSkew}$
- $\text{Total pixel} = H\text{-total} \times V\text{-total} \times 60 \text{ (Hz)}$
- $\text{Bitclk} = \text{Total pixel} \times \text{bpp (byte)} \times 8/\text{lane number}$
- $\text{Byteclk} = \text{bitclk}/8$
- $\text{Dsiclk} = \text{Byteclk} \times \text{lane number}$
- $\text{Dsipclk} = \text{dsiclk}/\text{bpp (byte)}$

Table 2-1 shows the clock relation requirements.

Table 2-1 Clock relation requirements

Clock relation	Frequency ratio
Bit clock to byte clock	8 – 1
Byte clock to DSI clock	1 – #lanes
DSI clock to pclk (Video mode operation)	Video mode pixel depth – 1
DSI clock to pclk (no Video mode operation)	DSI clk <= pclk x Command mode pixel depth

3 Design Consideration

3.1 Performance consideration

Table 3-1 shows the recommended lane configuration to guarantee 60 fps performance with respect to LCD resolution. Consider this condition to meet the specification. Also ensure that the client can support the required clock to meet the targeted performance.

Table 3-1 Lane configuration recommendation for various resolutions

Resolution	Width	Height	Lane	Pclk (MHz)	Dsclk (MHz)	Byteclk (MHz)	Bitclk (MHz)	Link data rate (Mbps)
QVGA	320	240	1	6	18	18	146	146
WQVGA	432	240	1	8	25	25	197	197
HVGA	480	320	1	12	37	37	293	293
QHD	640	360	2	18	55	27	219	439
VGA	640	480	2	24	73	37	293	585
WVGA	800	480	2	30	91	46	366	731
SVGA	800	600	3	38	114	38	305	914
FWVGA	864	480	2	33	99	49	395	790
¾ HD	960	540	3	41	123	41	329	987
WSVGA	1024	540	3	44	132	44	351	1053
WSVGA	1024	600	3	49	146	49	390	1170
XGA	1024	768	4	62	187	47	374	1498
XGA	1280	720	4	73	219	55	439	1755
720P HD	1280	768	4	78	234	59	468	1872
WXGA	1280	800	4	81	244	61	488	1950
WXGA+	1440	900	4	103	309	77	617	2468

4 MIPI DSI Driver Migration Guide

The display driver includes a reference MIPI DSI driver that is tested in-house. The display driver supports both Command and Video mode. However, since a different driver IC/panel can be used, you are responsible for modifying the driver to light up the panel.

Since the MIPI DSI host (including PHY) is inside the APQ, almost all of the registers are set to facilitate a display update through the MIPI DSI interface. However, some of the registers should be changed to fit the driver IC/panel specification. Proper implementation should be performed to initialize the driver IC/panel.

Panel initialization is dependent upon the driver IC/panel type. This chapter explains how to set the MDP/MIPI DSI register and how to change the MIPI DSI driver to apply those register changes.

This chapter explains how to get the value to set the MDP/DSI registers related to the MIPI DSI interface directly from the user specification. It also explains what you need to change for the Android MIPI DSI driver.

4.1 Setting up DSI panel-related GPIO reset pins

Information about setting up DSI panel-related GPIO reset pins can be found at `\Linux\android\kernel\arch\arm\mach-msm\board-8064-display.c`.

4.1.1 Vsync GPIO

MDP_Vsync input must use GPIO0 and set the alternative function as mdp_vsync input.

```
#define MDP_VSYNC_GPIO 0
```

4.1.2 Initialization GPIO high/low/high location

In the `mipi_dsi_platform_data mipi_dsi_pdata` structure, `.dsi_power_save = mipi_dsi_panel_power`.

Add a specific function call for the OEM panel init on the bottom of the function `mipi_dsi_panel_power()`.

```
static struct mipi_dsi_platform_data mipi_dsi_pdata = {
    .vsync_gpio = MDP_VSYNC_GPIO,
    .dsi_power_save = mipi_dsi_panel_power,
};
```

4.1.3 Example

NOTE: **Red boldface** indicates code to be **added**.

The following example shows how to set up DSI panel-related GPIO reset pins.

1. Define the specific GPIO pin.

```
#define MIPI_LCD_RST      (75)
#define AVDD3V           (77)
#define IOVDD            (83)
```

2. Create a new structure for the OEM panel.

```
#ifdef CONFIG_FB_MSM_MIPI_DSI
static struct platform_device mipi_dsi_toshiba_panel_device = {
    .name = "mipi_toshiba",
    .id = 0,
    .dev = {
        .platform_data = &toshiba_pdata,
    }
};

static struct platform_device mipi_dsi_video_sharp_qHD_panel_device = {
    .name = "dsi_video_sharp_qHD",
    .id = 0,
    .dev = {
        .platform_data = &toshiba_pdata,
    }
};

if (machine_is_msm8960_liquid())
    ptr = &mipi_dsi2lvds_bridge_device;
elseif
    ptr = &mipi_dsi_toshiba_panel_device;
ptr = &mipi_dsi_video_sharp_qHD_panel_device; //add your data
structure here
```

3. Add the specific power function.

```
static int dsi_panel_power(int on)
{
    mipi_sharp_panel_power(on);
    if (machine_is_msm8960_liquid())
        ret = mipi_dsi_liquid_panel_power(on);
    else
        ret = mipi_dsi_cdp_panel_power(on);
    return 0;
}
```

4.1.3.1 Panel power-on function example

```
static void mipi_sharp_panel_power(int on)
{

    ret = gpio_request(AVDD3V, "AVDD3V");
    if (ret)
        printk(KERN_ERR "%s: RESET gpio %d request""failed\n", __func__, AVDD3V);
    gpio_direction_output(AVDD3V, 0);
    ret = gpio_request(IOVDD, "IOVDD");
    if (ret)
        printk(KERN_ERR "%s: RESET gpio %d request""failed\n", __func__, IOVDD);
    gpio_direction_output(IOVDD, 0);
    ret = gpio_request(MIPI_LCD_RST, "lcd_reset");
    if (ret)
        printk(KERN_ERR "%s: RESET gpio %d request""failed\n",
            __func__ MIPI_LCD_RST);
    gpio_direction_output(MIPI_LCD_RST, 0);

    ,
    gpio_set_value(AVDD3V, 1);
    mdelay(4);
    //AVDD OFF
    gpio_set_value(AVDD3V, 0);
    mdelay(50);
    //AVDD ON
    gpio_set_value(AVDD3V, 1);
    //IOVDD ON
    gpio_set_value(IOVDD, 1);
    mdelay(1);
    gpio_set_value(MIPI_LCD_RST, 1);
    mdelay(1);
    gpio_set_value(MIPI_LCD_RST, 0);
    mdelay(1);
    gpio_set_value(MIPI_LCD_RST, 1);
    mdelay(60);
}
```

4.1.3.2 Panel power-off function example

```
mdelay(120);
gpio_set_value(MIPI_LCD_RST, 1);
mdelay(10);
//AVDD OFF
gpio_set_value(AVDD3V, 0);
mdelay(10);
//IOVDD OFF
gpio_set_value(IOVDD, 0);
```

4.2 DSI host initialization-related code

4.2.1 Initialization sequence call flow

- mipi_dsi.c
 - mipi_dsi_probe()
 - Binding IRQ – Request_irq(DSI_IRQ, mipi_dsi_isr, IRQF_DISABLED, "MIPI_DSI", 0);
 - Assign vsync – vsync_gpio = mipi_dsi_pdata->vsync_gpio;
 - Binding data chain
 - pdata = mdp_dev->dev.platform_data;
 - pdata->on = mipi_dsi_on;
 - pdata->off = mipi_dsi_off;
 - pdata->next = pdev;
 - Pull panel info
 - Auto-calculate DSI core clock/PCLK/bitclk : mipi_dsi_clk_div_config()
 - Auto-calculate D-PHY by mipi_dsi_phy_pll_config()
 - mipi_dsi_on()
 - mipi_dsi_pdata->dsi_power_save(1); call dsi_panel_power() in board-8064-display.c
 - Enable DSI core clock – mipi_dsi_clk(&dsicore_clk, 1);
 - Enable DSI PCLK – mipi_dsi_pclk(&dsi_pclk, 1);
 - DSI PHY setting – mipi_dsi_phy_init()
 - Set up DSI host – mipi_dsi_host_init()
 - Clean up error status report from DSI client – mipi_dsi_cmd_bta_sw_trigger()
 - Start next panel on function for FB1 if it appears in the link list – panel_next_on()
 - Change GPIO AF as mdp_vsync if hardware Vsync is enabled
 - Send out TE enable command if hardware Vsync – mipi_dsi_set_tear_on()

- `msm_dss_io_8960.c`
 - Include all DSI/PHY clock setup/calculate functions

4.2.2 DSI clock enable

- `mipi_dsi_ahb_ctrl()`
 - `clk_enable(amp_pclk)` – Clock for AHB-master to AXI
 - `clk_enable(dsi_m_pclk)` – 0x0008 AHB_EN bit 9:DSI_M_AHB_CLK_EN
 - `clk_enable(dsi_s_pclk)` – 0x0008 AHB_EN bit 18 DSI_S_AHB_CLK_EN
- `mipi_dsi_clk_enable()`
 - `mipi_dsi_pclk_ctrl()` – Enable DSI pixel clock
 - `mipi_dsi_clk_ctrl()` – Enable DSI core clock
 - `clk_enable(dsi_byte_div_clk)` – Enable DSI byte clock
 - `clk_enable(dsi_esc_clk)` – Enable DSI esc clock
 - `mipi_dsi_clk()` – Turn on DSI core clock and change M:N/D value based on `mipi_dsi_clk_div_config()` result in `DSI_probe`
 - `mipi_dsi_pclk()` – Turn on DSI pixel clock and change M:N/D value based on `mipi_dsi_clk_div_config()` result in `DSI_probe`

4.2.3 `mipi_dsi_clk_div_config()` – DSI core clock and DSI PCLK

This major function can auto-adjust the DSI core clock/pixel clock/PHY PLL control clock. The following instructions show how the driver auto-calculates these clocks.

Reference table in source code in `mipi_dsi.h`.

```
struct dsi_clk_mnd_table {
    uint8 lanes;
    uint8 bpp;
    uint8 dsiclk_div;
    uint8 dsiclk_m;
    uint8 dsiclk_n;
    uint8 dsiclk_d;
    uint8 pclk_m;
    uint8 pclk_n;
    uint8 pclk_d;
};

#define PREF_DIV_RATIO 27
static struct dsiphy_pll_divider_config pll_divider_config;
static const struct dsi_clk_mnd_table mnd_table[] = {
    { 1, 2, 8, 1, 1, 0, 1, 2, 1},
    { 1, 3, 8, 1, 1, 0, 1, 3, 2},
    { 2, 2, 4, 1, 1, 0, 1, 2, 1},
    { 2, 3, 4, 1, 1, 0, 1, 3, 2},
}
```

```
{ 3, 2, 1, 3, 8, 4, 3, 16, 8},
{ 3, 3, 1, 3, 8, 4, 1, 8, 4},
{ 4, 2, 2, 1, 1, 0, 1, 2, 1},
{ 4, 3, 2, 1, 1, 0, 1, 3, 2},
```

Two clocks must be modified inside `msm_dss_io_8960.c`:

- DSI_PCLK – 0x0130 DSI_PIXEL_CC, 0x0134 DSI_PIXEL_MD, and 0x0138 DSI_PIXEL_NS in the MMSS section
- DSI_clock – 0x004C DSI_CC, 0x0050 DSI_MD, and 0x0054 DSI_NS in the MMSS section

```
0x0138 DSI_PIXEL_NS
```

```
If((pclk_d == 0) || (pclk_m == 1)) - mnd_mode = 0 (Bypass mode)
```

```
2:0 SRC_SEL: always = 0x3 - select source as "dsi_phy_pll0_src"
```

```
15:12 PRE_DIV_FUNC = pclk_n - 1 : reference to matrix mnd_table[]
```

```
Else
```

```
2:0 SRC_SEL: always = 0x3 - select source as "dsi_phy_pll0_src"
```

```
23:12:
```

```
val = pclk_n - pclk_m;
```

```
data = (~val) & 0x0ff;
```

```
data <= 24;
```

```
0x0130 DSI_PIXEL_CC
```

```
If((pclk_d == 0) || (pclk_m == 1)) - mnd_mode = 0 (Bypass mode)
```

```
Bit 2 ROOT_EN =1 , bit 0 CLK_EN =1
```

```
Else
```

```
Bit 7:6 MND_MODE= 2 (10 : Dual-edge mode)
```

```
Bit 5 MND_EN = 1
```

```
Bit 2 ROOT_EN =1
```

```
Bit 0 CLK_EN =1
```

```
0x0134 DSI_PIXEL_MD
```

```
If((pclk_d == 0) || (pclk_m == 1)) - mnd_mode = 0 (Bypass mode)
```

```
Don't need fill any data since it's disabled
```

```
Else
```

```
val = pclk_d * 2;
```

```
data = (~val) & 0x0ff;
```

```
data |= pclk_m << 8;
```

```
0x0054 DSI_NS
```

```
If((dsiclk_d== 0) || (dsiclk_m== 1)) - mnd_mode = 0 (Bypass mode)
```

```
15:14 PRE_DIV_FUNC = dsiclk_n - 1
```

```
2:0 SRC_SEL: always = 0x3 - select source as "dsi_phy_pll0_src"
```

```
Else
```

```
2:0 SRC_SEL: always = 0x3 - select source as "dsi_phy_pll0_src"
```

```
23:12:
```



```

val = dsiclk_n - dsiclk_m;
data = (~val) & 0x0ff;
data <<= 24;

0x004C DSI_CC
If((dsiclk_d== 0) || (dsiclk_m== 1)) - mnd_mode = 0 (Bypass mode)
Bit 2 ROOT_EN =1 , bit 0 CLK_EN =1 , bit 8 PMXO_SEL = 0 , bit 7:6 MND_MODE
= 0
Else
Bit 8 PMXO_SEL = 0 , 7:6 MND_MODE = 2 (10 : Dual-edge mode) , bit 5 MND_EN
= 1 , Bit 2 ROOT_EN =1 , bit 0 CLK_EN =1
0x0050 DSI_MD
If((dsiclk_d== 0) || (dsiclk_m== 1)) - mnd_mode = 0 (Bypass mode)
There is no need to fill any data, since it is disabled.
Else
val = pclk_d * 2;
data = (~val) & 0x0ff;
data |= pclk_m << 8;

```

4.2.4 DSI PHY configuration

There are five DSI PHY-related registers.

1. DSIPHY_REGULATOR_CTRL_0-4 – Start from 0x500
2. DSIPHY_TIMING_CTRL_0-11 – Start from 0x0440
 - Mapping to DSI PHY-related timing parameter, such as T2TM/T3/T4, T6/T7/T8... (MIPI_D-PHY_Specification_v1.0).
For descriptions of T2, T3, etc., see example table [Table 5-1](#).
3. DSIPHY_CTRL_0-3 – Start from 0x0470
4. DSIPHY_STRENGTH_CTRL_0-3 – Start from 0x0480
5. DSIPHY_PLL_CTRL_0-20 – Start from 0x0200
 - Decide how to generate DSI_bit clock/byte clock/pixel clock

mipi_dsi_phy_init()

- Reset DSI PHY first
- Reference to dsi_cmd_mode_phy_db[] in panel driver
- Regulator – Do not change
- DSI PHY timing control – Reference to dsi_cmd_mode_phy_db[] in panel driver
 - Default value is middle of DSI PHY specification between min/max
- DSI PHY control – Do not change
- Change DSIPHY_PLL_CTRL_1/2/3/8/9/10 again in mipi_dsi_phy_pll_config()
 - Get related value from function mipi_dsi_clk_div_config()

4.3 DSI client-side-related code change

4.3.1 Panel driver

See the API of `mipi_cmd_novatek_blue_qhd_pt_init()` in `\LINUX\android\kernel\drivers\video\msm\mipi_novatek_video_qhd_pt.c`.

The following parameters must be modified based on the user specification. Most of the values were also used in `mipi_dsi_phy_pll_config()` for auto-calculation purposes.

- `pinfo.xres` – Panel resolution width
- `pinfo.yres` – Panel resolution height
- `pinfo.lcdc.xres_pad` – Dummy lines that are larger than LCD resolution. The main purpose is to keep horizontal line clock status in HS mode. Note that this parameter is not always needed and the parameter is used depending on the panel requirement.
- `pinfo.lcdc.yres_pad` – Dummy lines that larger than LCD resolution. The major purpose is to keep vertical line clock status in HS mode. Note that this parameter is not always needed and the parameter is used depending on the panel requirement.
- `pinfo.wait_cycle` – Set as 0, since there should not be any delay between frame and frame
- `pinfo.bpp` – Color depth
- `pinfo.lcdc.h/v_back_porch/front_porch/pulse_width` – LCD porch value from LCM spec
- `pinfo.lcdc.border_clr` – Defines border (inactive) area color for LCD display; 0 x 0 = black
- `pinfo.lcdc.underflow_clr` – Sets underrun interrupt color; 0xff – blue, 0xff00 – green, 0xff0000 – red
- `pinfo.lcdc.hsync_skew` – Skew value for LCD
- `pinfo.bl_max` – Backlight level 0 to `bl_max` with rounding
- `pinfo.bl_min` – Backlight level minimum level (Android define 0~255)
- `pinfo.fb_num` – How many frame buffers for flip
- `pinfo.clk_rate` – Bit clock rate for panel
- `hw vsync` – `pinfo.lcd.vsync_enable/pinfo.lcd.hw_vsync_mode` – Enable or do not enable hardware TE signal
- `pinfo.lcd.refx100` – Hardware Vsync parameter adjust 0x0100 DP_SYNC_CONFIG_P bit 18:0 VSYNC_COUNT value
- `pinfo.mipi.mode` – Command mode or Video mode
- `pinfo.mipi.pulse_mode_hsa_he` – No HSA and HE following VS/VE packet or send HSA and HE following VS/VE packet
- `pinfo.mipi.hfp_power_stop` – Power mode during Horizontal Front Porch (HFP) period – Send blanking packets in HS mode or LP Stop mode
- `pinfo.mipi.hbp_power_stop` – Power mode during Horizontal Back Porch (HBP) period – Send blanking packets in HS mode or LP Stop mode

- `pinfo.mipi.hsa_power_stop` – Power mode during HSA period – Send blanking packets in HS mode or LP Stop mode
- `pinfo.mipi.eof_bllp_power_stop` – Power mode for Blanking or LP (BLLP) interval of last line of a frame. Send blanking packets during BLLP in HS mode and block Command mode packets or LP Stop mode (LP-11), or let Command mode engine send packets in HS or LP mode.
- `pinfo.mipi.bllp_power_stop` – DSI Power mode for packets sent during BLLP period. Send blanking packets during BLLP in HS mode and block Command mode packets or LP Stop mode (LP-11), or let Command mode engine send packets in HS or LP mode.
- `pinfo.mipi.traffic_mode` – DSI Video mode traffic sequence
- `pinfo.mipi.tx_eot_append` – `TX_EOT_APPEND` – Specify whether the EOT packet must be appended at the end of each forward HS data burst
- `pinfo.mipi.dst_format` – Panel color format
- `pinfo.mipi.vc` – MIPI virtual channel number
- `pinfo.mipi.rgb_swap` – Set as `DSI_RGB_SWAP_BGR`, swap RGB to BRG
- `pinfo.mipi.data_lane0-3` – Lane numbers; set TRUE to enable
- `pinfo.mipi.t_clk_post` – T14 value (MIPI_D-PHY_Specification_v065 – Figure 21)
- `pinfo.mipi.t_clk_pre` – T15 time; need fill based on LCM vendor specification; if vendor does not provide; check with vendor and get value
- `pinfo.mipi.stream` – Register 0x0080 `DSI_TRIG_CTRL` bit 8
`COMMAND_MODE_DMA_STREAM_SEL`, set as 0 for DMAP
- `pinfo.mipi.mdp_trigger/pinfo.mipi.dma_trigger` – Set both as software trigger
- `pinfo.mipi.te_sel` – Set as 1 for GPIO AF function as `mdp_vsync`
- `pinfo.mipi.insert_dcs_cmd` – Insert DCS command as the first byte of payload of the pixel data packet or not
- `pinfo.mipi.interleave_max`
 - Maximum number of Command mode RGB packets to send within one horizontal blanking period of Video mode frame (software must ensure that the package number can fit in one BLLP period)
- `pinfo.mipi.wr_mem_continue/pinfo.mipi.wr_mem_start` – In register 0x0040
`DSI_COMMAND_MODE_MDP_DCS_CMD_CTRL` bit 0~15
 - DSI spec `wr_mem_continue` and `wr_mem_start` command

4.3.2 Panel-specific initialization sequence

Panel driver on.

- Mipi_novatek.c function mipi_novatek_lcd_on()
- Panel init command sequence on structure

```
static struct dsi_cmd_desc novatek_cmd_on_cmds[] = {
    {DTYPE_DCS_WRITE, 1, 0, 0, 50,
    sizeof(sw_reset), sw_reset},
    {DTYPE_DCS_WRITE, 1, 0, 0, 10,
    sizeof(exit_sleep), exit_sleep},
    {DTYPE_DCS_WRITE, 1, 0, 0, 10,
    ...

```

- mipi_dsi_cmd_bta_sw_trigger – Clean up ack_err_status that DSI client reports to host
- mipi_novatek_manufacture_id() reads manufacture ID

4.3.2.1 dsi_cmd_desc

This section describes the dsi_cmd_desc structure.

```
struct dsi_cmd_desc {
    int dtype;
    int last;
    int vc;
    int ack;
    int wait;
    int dlen;
    char payload
};

```

Table 4-1 Parameter descriptions

Parameter	Description
dtype	Data type of the command
last	Configures whether the command will be sent one at a time or clubbing commands together
vc	The virtual channel ID
ack	Asks for ack from peripheral
Wait	The amount of time needed to wait after the command is sent
dlen	The total length of the command in bytes
payload	Pointer to the start of payload in the command packet

5 Panel Bring-up

You must prepare the following items for the DSI panel bring-up on the APQ8064E for Android. Without these items, support cannot help you in bringing-up the panel:

- Panel specification, which includes the panel parameter information in Section [4.3.1](#)
- Panel power-on sequence and signal duration for GPIO pins, for example, RESET/IOVDD
- Panel DSI initial command sequence and duration information in Section [4.3.2](#)
- bitclk, which the panel needs in order to reach the fps target number

5.1 LCD parameter fill

Fill panel parameters, as discussed in Section 4.3.1, into the panel driver, `mipi_(vendor name)_(cmd/video)_(resolution)_(pt).c`, e.g., `mipi_himax_video_720p_pt.c`.

5.2 PHY table modify inside panel driver

The `mipi_dsi_phy_ctrl` structure, including the PHY setting is shown. Note that the *DB_APQ8064E_DSI_Timing_Program.xlsm* file (attached to this document) is necessary to generate the bolded values. Section [5.3](#) describes how to use the .xlsm file.

```
static struct mipi_dsi_phy_ctrl dsi_video_mode_phy_db = {
/* DSIPHY_REGULATOR_CTRL */
.regulator = {0x03, 0x0a, 0x04, 0x00, 0x20}, /* common 8064 */
/* DSIPHY_CTRL */
.ctrl = {0x5f, 0x00, 0x00, 0x10}, /* common 8064 */
/* DSIPHY_STRENGTH_CTRL */
.strength = {0xff, 0x00, 0x06, 0x00}, /* common 8064 */
/* DSIPHY_TIMING_CTRL */
.timing = { 0xB6, 0x8D, 0x1E, /* panel specific */
0, /* DSIPHY_TIMING_CTRL_3 = 0 */
0x21, 0x95, 0x21, 0x8F, 0x21, 0x03, 0x04}, /* panel specific */
/* DSIPHY_PLL_CTRL */
 pll = { 0x00, /* common 8064 */
/* VCO */
0xC6, 0x01, 0x19, /* panel specific */
0x00, 0x50, 0x48, 0x63,
```

```

0x77, 0x88, 0x99, /* Auto update by dsi-mipi driver */
0x00, 0x14, 0x03, 0x00, 0x02, /* common 8064 */
0x00, 0x20, 0x00, 0x01 }, /* common 8064 */
};

```

5.3 Use dsi_timing_program to calculate DSI PHY register

To register:

1. Check with the bridge IC vendor to obtain a bridge IC PHY timing spec, as listed in [Table 5-1](#).

Table 5-1 Example of bridge IC PHY settings

Item	Symbol	Unit	Test condition	Min	Typical	Max
Time to drive LP-00 to prepare for HS trans.	T7	ns	IOVCC = 1.65 V ~ 3.30v DPHYVCC = 1.65 V ~3.30 V	40 ns + 4*T1	—	85 ns + 6*T1
T7 + Time to drive HS-0 before the Sync sequence	T7 + T8	ns	IOVCC = 1.65 V ~ 3.30v DPHYVCC = 1.65 V ~3.30 V	145 ns + 10*T1	—	—
Time to drive flipped differential state after last payload data bit of a HS trans. burst	T9	ns	IOVCC = 1.65 V ~ 3.30v DPHYVCC = 1.65 V ~3.30 V	Max (n*8* T1, 60 ns + n*4*T1)	—	—
Time to drive LP-11 after HS burst	Init	ns	IOVCC = 1.65 V ~ 3.30v DPHYVCC = 1.65 V ~3.30 V	100	—	—
Time to drive LP-00 after Turnaround Req.	T12		IOVCC = 1.65 V ~ 3.30v DPHYVCC = 1.65 V ~3.30 V	4*tlptx		
Timeout before new Tx side starts drive	T13		IOVCC = 1.65 V ~ 3.30v DPHYVCC = 1.65 V ~3.30 V	1*tlptx	—	2*tlptx
Time to drive LP-00 by new Tx	T14		IOVCC = 1.65 V ~ 3.30v DPHYVCC = 1.65 V ~3.30 V	5*tlptx		
Length of any low-power state period	T2	ns	IOVCC = 1.65 V ~ 3.30v DPHYVCC = 1.65 V ~3.30 V	50	—	—
Ratio of T1(MASTER) /T1(SLAVE) between Master and Slave Side	Ratio T2		IOVCC = 1.65 V ~ 3.30v DPHYVCC = 1.65 V ~3.30 V	2/3	—	3/2
Time that the transmitter continues to send HS clock after the last associated Data Lane has transitioned to LP Mode	T17	T1	IOVCC = 1.65 V ~ 3.30v DPHYVCC = 1.65 V ~3.30 V	60 ns + 52 T1	—	—
T4 + time for lead HS-0 drive period before stating Clock	T4+ T5	ns	IOVCC = 1.65 V ~ 3.30v DPHYVCC = 1.65 V ~3.30 V	300	—	—

Item	Symbol	Unit	Test condition	Min	Typical	Max
Time that the HS clock shall be driven by the transmitter prior to any associated Data Lane beginning the transition from LP to HS mode	T18	T1	IOVCC = 1.65 V ~ 3.30v DPHYVCC = 1.65 V ~3.30 V	8	–	–
Time to drive LP-00 to prepare for HS clock trans.	T4	ns	IOVCC = 1.65 V ~ 3.30v DPHYVCC = 1.65 V ~3.30 V	38	–	95
Time to drive HS differential state after last payload clock bit of a HS clock trans.	T6	ns	IOVCC = 1.65 V ~ 3.30v DPHYVCC = 1.65 V ~3.30 V	60	–	–
Time from start of T7 period to start of LP-11 state	T15, T16		IOVCC = 1.65 V ~ 3.30v DPHYVCC = 1.65 V ~3.30 V	–	–	105 ns + n*12*UI
Length of low-power Tx period in case of using DSI clock	Tx1	T1	IOVCC = 1.65 V ~ 3.30v DPHYVCC = 1.65 V ~3.30 V	–	32	–
Length of low-power Tx period in case of using internal OSC clock	Tx2	ns	IOVCC = 1.65 V ~ 3.30v DPHYVCC = 1.65 V ~3.30 V	–	1/fosc	–

T0 (ESC clock) – 13.5 MHz

T3 – 30-85% rise and fall time

T10 – Request of protocol for high-speed transmission

T11 – Time that drives LP-11 following HS burst

- Based on the required bitclk panel, calculate the value listed in [Table 5-1](#):

- $T1 = 10^3/\text{bitclk (MHz)}$
- $T2 = 10^3/T0 (13.5 \text{ MHz})$
This calculation gives T2 in ns. T0 should be in MHz

- Map the following DSI spec name to the APQ8064E software interface register:

- T4
- T5
- T6
- T7
- T8
- T9
- T10
- T11
- T12


- T13
- 14
- T15, T16
- T17
- T18

This maps the APQ8064E software interface register description from 0x0440 DSI1_DSIPHY_TIMING_CTRL_0 to 0x0468 DSI1_DSIPHY_TIMING_CTRL_10.

NOTE: Only the **yellow highlighted** parameters must be changed. Do not change any other value in this matrix.

NOTE: Different APQs have different matrix values so do not apply the APQ8064E table to any other APQ.

4. Calculate DSI PHY registers using dsi_timing_program:

- a. Open *DB_APQ8064E_DSI_Timing_Program.xlsm* and look at the DSI and MDP registers worksheet. Click the paperclip icon  (attachment button) on the left side of the PDF to open the xlsm file.
- b. Enter the values in blue cells on this worksheet as shown in [Figure 5-1](#).

Micros

File Home Insert Page Layout Formulas Data Review View Developer Add-Ins Acrobat PRISM

Cut Copy Paste Format Painter Clipboard Font Alignment Number

Arial 10 A Wrap Text General Normal 2 Neutral

B I U Merge & Center \$ % +.00 .00

Conditional Formatting as Table

H37

DB_APQ8064_DSL_Timing_Program

	A	B	C	D	E	F	G	H	I	J	K	
1		Enter requirements (Enter values in blue)										
2		frame rate	60	frame per sec								
3		lane config	4	lanes								
4		pixel format BPP	3	bytes/pixel								
5												
6		Display Width	720	pixels	(including reqd. border fill)							
7		Display Height	1280	lines	(including reqd. border fill)							
8		Active Width	720	pixels	(active image region)							
9		Active Height	1280	lines	(active image region)							
10												
11		Hsync Pulse Width	3	pcilks	ok							
12		Hori. Back Porch	45	pcilks	ok							
13		Hori. Back Porch + hsync pulse width	48	pcilks								
14		Hori. Front Porch	156	pcilks	ok							
15												
16		Vsync Pulse Width	4	lines								
17		Vert. Back Porch	3	lines								
18		Vert. Back Porch + Vsync pulse width	7	lines								
19		Vert. Front Porch	9	lines								
20												
21		DSI PLL reference clock (mxo = 27MHz or pxo = 24MHz)	27	MHz								
22		Esck source (mxo = 27MHz or pxo = 24MHz)	27	MHz								
23		% within allowable PHY timing range	10	%								
24												
25												
26		To run macro that verifies the video parameters with respect to the DSI video programming rules, press Ctrl-L										
27		Check done!										
28												

Figure 5-1 The DSI and MDP registers worksheet

The correct DSIPHY_TIMING_CTRL values appear, as shown in [Figure 5-2](#).

2. DSI PHY registers

PHY Registers (address)	value in hex
DSIPHY_TIMING_CTRL_0 (0x260)	7B
DSIPHY_TIMING_CTRL_1 (0x264)	1B
DSIPHY_TIMING_CTRL_2 (0x268)	12
DSIPHY_TIMING_CTRL_4 (0x270)	40
DSIPHY_TIMING_CTRL_5 (0x274)	49
DSIPHY_TIMING_CTRL_6 (0x278)	17
DSIPHY_TIMING_CTRL_7 (0x27C)	1E
DSIPHY_TIMING_CTRL_8 (0x280)	1E
DSIPHY_TIMING_CTRL_9 (0x284)	3
DSIPHY_TIMING_CTRL_10 (0x288)	4

Figure 5-2 DSIPHY_TIMING_CTRL

5. Place those values into the matrix `mipi_dsi_phy_ctrl` timing part.

```
/* timing */
{0xae, 0x3c, 0x1b, 0x00, 0x54, 0x48, 0x1D,
0x40, 0x2f, 0x03, 0x04},
```

NOTE: DSIPHY_TIMING_CTRL_3 is 0x00. DSIPHY_TIMING_CTRL_11 is used for the DSI secondary display, which does not exist on the device. Therefore, it is not necessary to modify it if there is no DSI secondary panel.

NOTE: DSI PHY timing parameters can be calculated using the standard D-PHY specification listed in [Table 5-1](#). Based on the input values given by the customer and % within allowable PHY timing range, the Excel sheet (attached to this document) calculates the DSI PHY timings. But if you must modify any specific timing, you can do it and the Excel sheet will update accordingly.

6 Panel Bring-up Checkpoints

6.1 No DSI clock output

6.1.1 Check function `mipi_dsi_clk_ctrl ()`

Read 0x4c/0x50/0x54 of the DSI register value and determine if 0 appears.

```
printk(KERN_ERR "mmss_cc_base + 0x004c =%x \n", (uint)MIPI_INP_SECURE(cc));
printk(KERN_ERR "mmss_cc_base + 0x0050 =%x \n", (uint)MIPI_INP_SECURE(md));
printk(KERN_ERR "mmss_cc_base + 0x0054 =%x \n", (uint)MIPI_INP_SECURE(ns));
```

If 0 appears, the DSI host-related clock did not turn on successfully.

6.1.2 Check DSI_related clocks by ADB command

6. Adb root
7. Adb remount
8. Adb shell
9. # mount -t debugfs none /sys/kernel/debug
10. # cd /sys/kernel/debug/clk/dsi1_byte_clk
11. #cat measure

Then get the DSI byte clock as shown:

```
root@android:/sys/kernel/debug/clk/dsi1_byte_clk # cat measure
cat measure
56750912
```

NOTE: The DSI bitclk = DSI byte clock x8.

6.1.3 Check DSI_BITCLK output signal by scope

Since the DSI_BITCLK is a differential signal, the actual frequency is twice the time of the measured data.

Figure 6-1, Figure 6-2, and Figure 6-3 show examples.

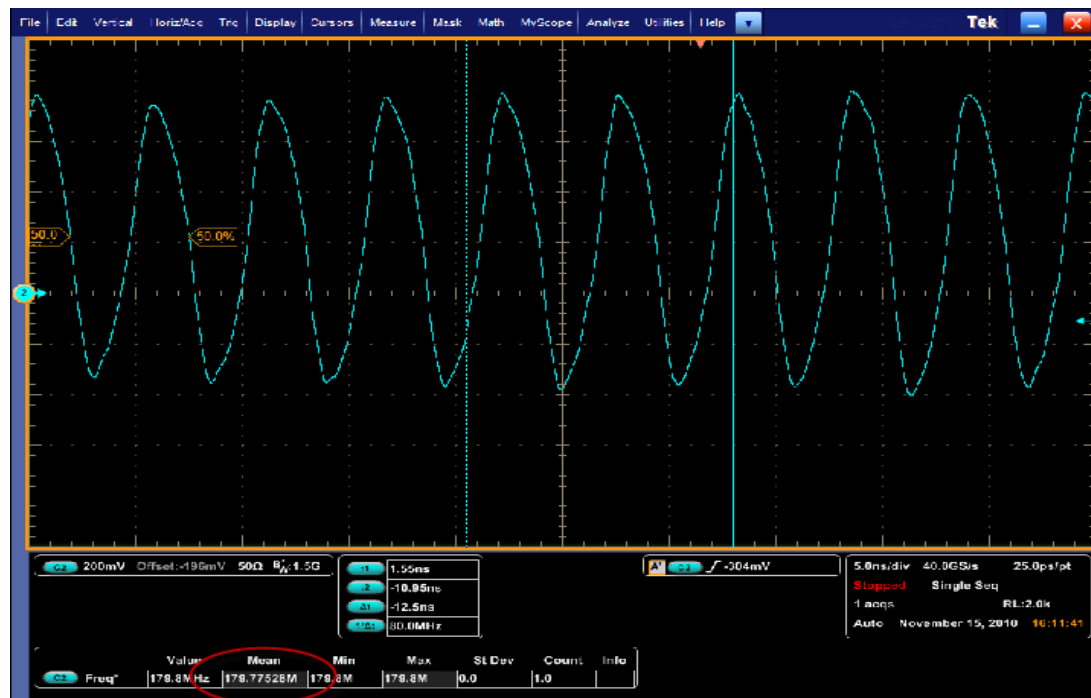


Figure 6-1 DSI_BITCLK = 366 MHz

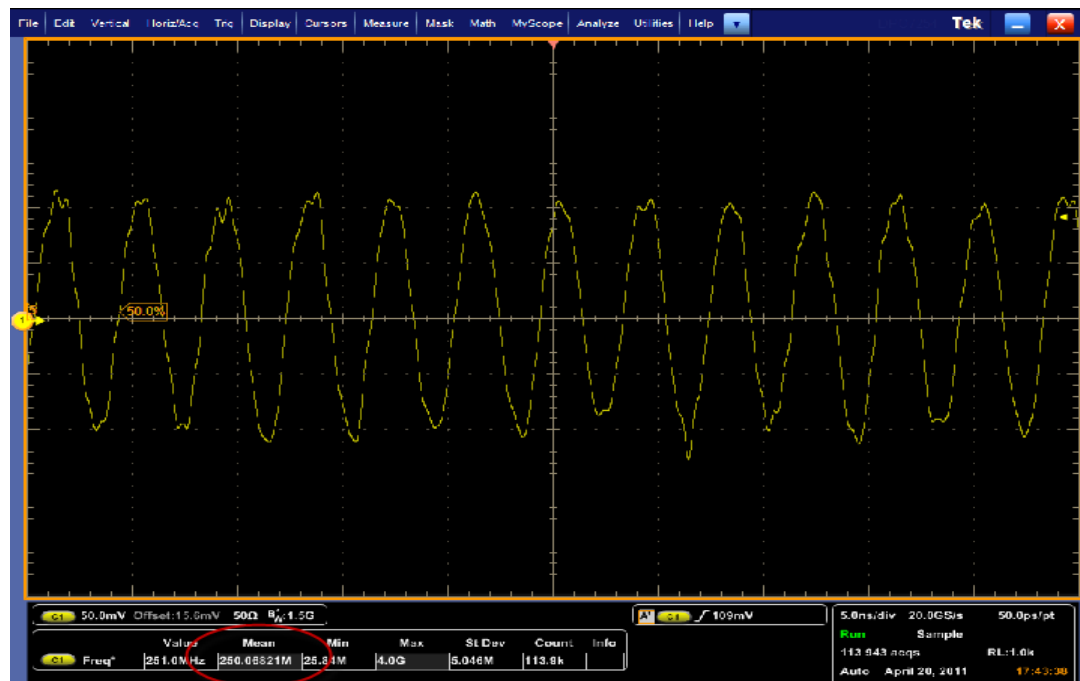


Figure 6-2 DSI_bit clock = 500 MHz

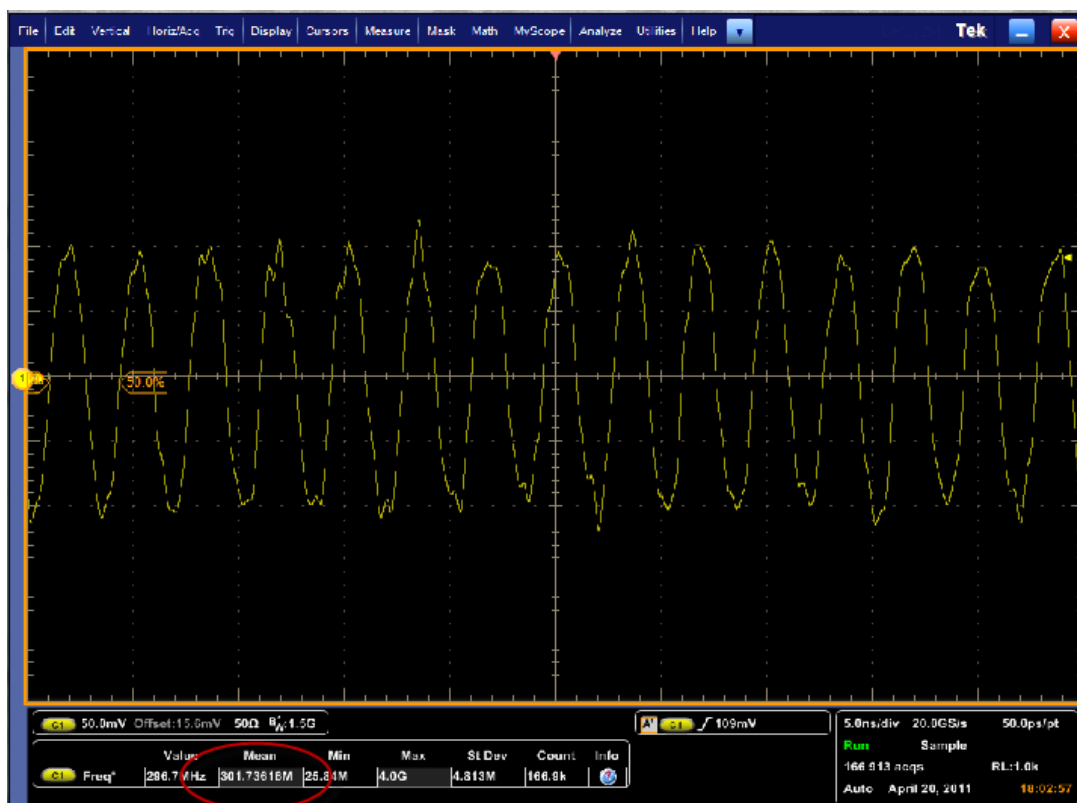


Figure 6-3 DSI_bit clock = 600 MHz

6.1.4 Possible file changes

If your display needs more pins than the default APQ8064E-based SYS6640 development platform display, then you may need changes to the following files:

- board-8064-regulator.c
- board-8064-pmic.c
- board-8064
- -gpiomux
- .c

6.2 Tearing on DSI Command mode LCD

Figure 6-4 shows the frame package transfer vs. TE signal.

1. Ensure DSI bitclk is high enough to update one frame package during 16.6 ms (60 Hz).
2. Ensure the frame updates bitclk on the scope as expected. Since bitclk is a differential signal, the frequency number on the scope x2 will match the bitclk rate exactly.

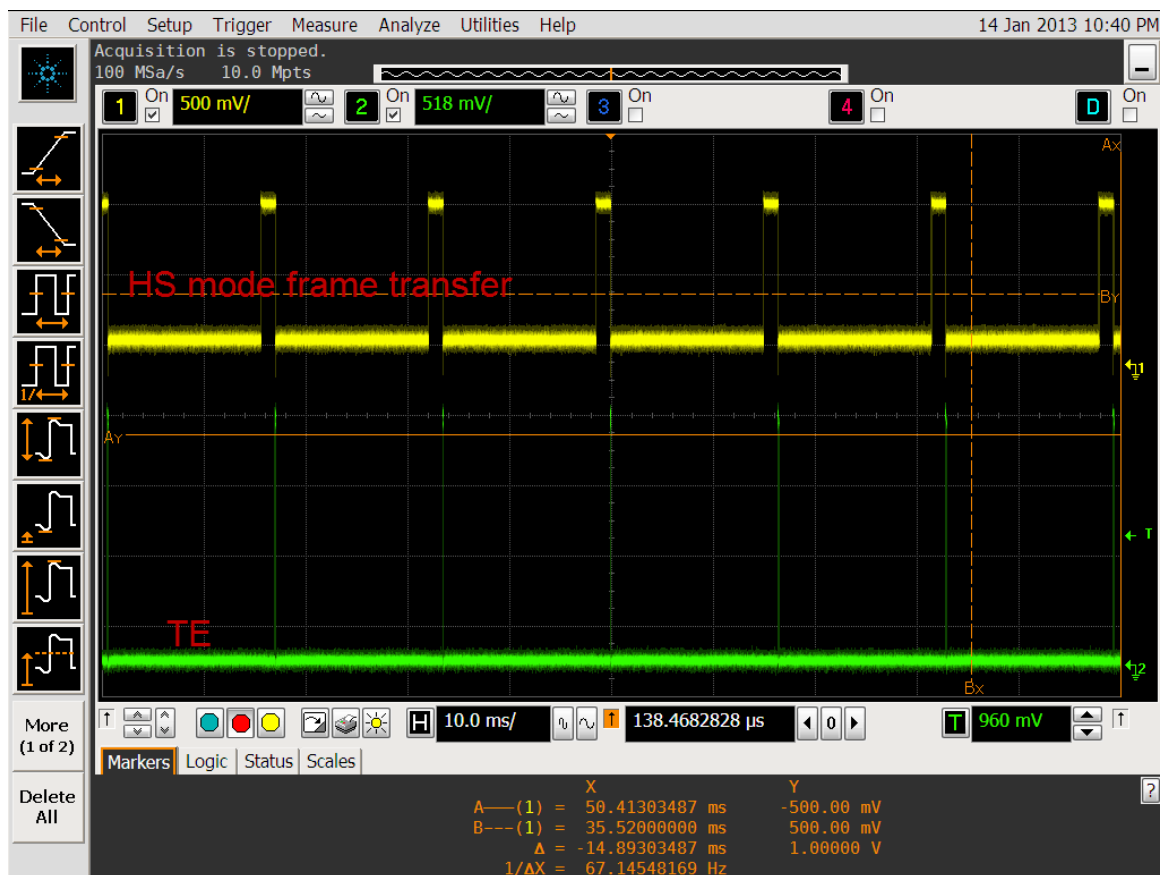


Figure 6-4 Frame package transfer vs. TE signal

7 Unstable/Locked Bitclk Issue

In APQ8064E, the bitclk may be unstable or locked if the VCO frequency is less than 600 MHz. The unstable bitclk may make the LCD unable to light up.

An abnormal low-power status and unstable bitclk can be measured by scope in this case. The bitclk locked can be checked with the MIPI_DSI_1_DSII_CLK_STATUS bit16 register, DSIPLL_UNLOCKED = 1.

The APQ8064E VCO clock range is 600 MHz ~ 1.2 GHz, which is guaranteed over process, voltage, temperature (PVT) variations testing.

7.1 VCO and bitclk setting formula on APQ8064E

The DSI regulator, strength, and DSII_DSIPHY_TIMING_CTRL do not need to change and only the PHY PLL change is necessary.

To change DSII_DSIPHY_PLL_CTRL, do the following:

1. If the bitclk rate > 600 MHz, program the VCO clock to 1*bitclk.
2. If the bitclk rate < 600 MHz, program the VCO clock to 2* data rate and the programmed VCO clock is still < 1.2 GHz.

In the DSI PHY PLL, DSII_DSIPHY_PLL_CTRL_1/2/3/8/9/10 needs to be changed, and the software code change as compared with the earlier version of the processor is shown below:

```
diff -git a/drivers/video/msm/msm_dss_io_8960.c
b/drivers/video/msm/msm_dss_io_8960.c
old mode 100644
new mode 100755
index 3b000ec..5e5d3fb
--- a/drivers/video/msm/msm_dss_io_8960.c
+++ b/drivers/video/msm/msm_dss_io_8960.c
@@ -328,7 +328,7 @@ int mipi_dsi_clk_div_config(uint8 bpp, uint8 lanes,
} else if (rate < 250) {
vco = rate * 4;
div_ratio = 4;
-     } else if (rate < 500) {
+     } else if (rate < 600) {
vco = rate * 2;
div_ratio = 2;
} else {
```

7.2 Verification by scope capture

Normal waveforms are verified by scope capture.

7.2.1 Normal waveform

A normal waveform is shown in [Figure 7-1](#).

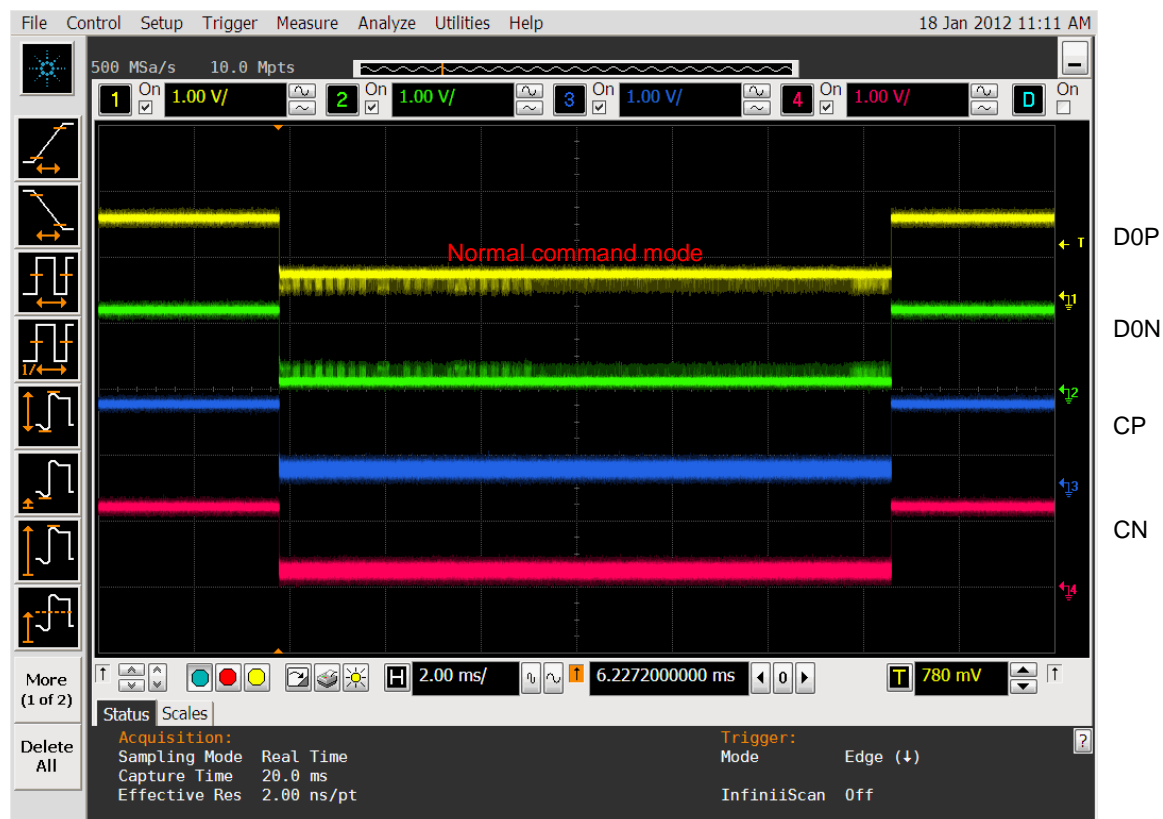


Figure 7-1 MIPI DSI bitclk = 578 MHz normal waveform

EXHIBIT 1

PLEASE READ THIS LICENSE AGREEMENT ("AGREEMENT") CAREFULLY. THIS AGREEMENT IS A BINDING LEGAL AGREEMENT ENTERED INTO BY AND BETWEEN YOU (OR IF YOU ARE ENTERING INTO THIS AGREEMENT ON BEHALF OF AN ENTITY, THEN THE ENTITY THAT YOU REPRESENT) AND QUALCOMM TECHNOLOGIES, INC. ("QTI" "WE" "OUR" OR "US"). THIS IS THE AGREEMENT THAT APPLIES TO YOUR USE OF THE DESIGNATED AND/OR ATTACHED DOCUMENTATION AND ANY UPDATES OR IMPROVEMENTS THEREOF (COLLECTIVELY, "MATERIALS"). BY USING OR COMPLETING THE INSTALLATION OF THE MATERIALS, YOU ARE ACCEPTING THIS AGREEMENT AND YOU AGREE TO BE BOUND BY ITS TERMS AND CONDITIONS. IF YOU DO NOT AGREE TO THESE TERMS, QTI IS UNWILLING TO AND DOES NOT LICENSE THE MATERIALS TO YOU. IF YOU DO NOT AGREE TO THESE TERMS YOU MUST DISCONTINUE AND YOU MAY NOT USE THE MATERIALS OR RETAIN ANY COPIES OF THE MATERIALS. ANY USE OR POSSESSION OF THE MATERIALS BY YOU IS SUBJECT TO THE TERMS AND CONDITIONS SET FORTH IN THIS AGREEMENT.

1.1 **License.** Subject to the terms and conditions of this Agreement, including, without limitation, the restrictions, conditions, limitations and exclusions set forth in this Agreement, Qualcomm Technologies, Inc. ("QTI") hereby grants to you a nonexclusive, limited license under QTI's copyrights to use the attached Materials; and to reproduce and redistribute a reasonable number of copies of the Materials. You may not use Qualcomm Technologies or its affiliates or subsidiaries name, logo or trademarks; and copyright, trademark, patent and any other notices that appear on the Materials may not be removed or obscured. QTI shall be free to use suggestions, feedback or other information received from You, without obligation of any kind to You. QTI may immediately terminate this Agreement upon your breach. Upon termination of this Agreement, Sections 1.2-4 shall survive.

1.2 **Indemnification.** You agree to indemnify and hold harmless QTI and its officers, directors, employees and successors and assigns against any and all third party claims, demands, causes of action, losses, liabilities, damages, costs and expenses, incurred by QTI (including but not limited to costs of defense, investigation and reasonable attorney's fees) arising out of, resulting from or related to: (i) any breach of this Agreement by You; and (ii) your acts, omissions, products and services. If requested by QTI, You agree to defend QTI in connection with any third party claims, demands, or causes of action resulting from, arising out of or in connection with any of the foregoing.

1.3 **Ownership.** QTI (or its licensors) shall retain title and all ownership rights in and to the Materials and all copies thereof, and nothing herein shall be deemed to grant any right to You under any of QTI's or its affiliates' patents. You shall not subject the Materials to any third party license terms (e.g., open source license terms). You shall not use the Materials for the purpose of identifying or providing evidence to support any potential patent infringement claim against QTI, its affiliates, or any of QTI's or QTI's affiliates' suppliers and/or direct or indirect customers. QTI hereby reserves all rights not expressly granted herein.

1.4 **WARRANTY DISCLAIMER.** YOU EXPRESSLY ACKNOWLEDGE AND AGREE THAT THE USE OF THE MATERIALS IS AT YOUR SOLE RISK. THE MATERIALS AND TECHNICAL SUPPORT, IF ANY, ARE PROVIDED "AS IS" AND WITHOUT WARRANTY OF ANY KIND, WHETHER EXPRESS OR IMPLIED. QTI ITS LICENSORS AND AFFILIATES MAKE NO WARRANTIES, EXPRESS OR IMPLIED, WITH RESPECT TO THE MATERIALS OR ANY OTHER INFORMATION OR DOCUMENTATION PROVIDED UNDER THIS AGREEMENT, INCLUDING BUT NOT LIMITED TO ANY WARRANTY OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE OR AGAINST INFRINGEMENT, OR ANY EXPRESS OR IMPLIED WARRANTY ARISING OUT OF TRADE USAGE OR OUT OF A COURSE OF DEALING OR COURSE OF PERFORMANCE. NOTHING CONTAINED IN THIS AGREEMENT SHALL BE CONSTRUED AS (I) A WARRANTY OR REPRESENTATION BY QTI, ITS LICENSORS OR AFFILIATES AS TO THE VALIDITY OR SCOPE OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT OR (II) A WARRANTY OR REPRESENTATION BY QTI THAT ANY MANUFACTURE OR USE WILL BE FREE FROM INFRINGEMENT OF PATENTS, COPYRIGHTS OR OTHER INTELLECTUAL PROPERTY RIGHTS OF OTHERS, AND IT SHALL BE THE SOLE RESPONSIBILITY OF YOU TO MAKE SUCH DETERMINATION AS IS NECESSARY WITH RESPECT TO THE ACQUISITION OF LICENSES UNDER PATENTS AND OTHER INTELLECTUAL PROPERTY OF THIRD PARTIES.

1.5 **LIMITATION OF LIABILITY.** IN NO EVENT SHALL QTI, QTI'S AFFILIATES OR ITS LICENSORS BE LIABLE TO YOU FOR ANY INCIDENTAL, CONSEQUENTIAL OR SPECIAL DAMAGES, INCLUDING BUT NOT LIMITED TO ANY LOST PROFITS, LOST SAVINGS, OR OTHER INCIDENTAL DAMAGES, ARISING OUT OF THE USE OR INABILITY TO USE, OR THE DELIVERY OR FAILURE TO DELIVER, ANY OF THE MATERIALS, OR ANY BREACH OF ANY OBLIGATION UNDER THIS AGREEMENT, EVEN IF QTI HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. THE FOREGOING LIMITATION OF LIABILITY SHALL REMAIN IN FULL FORCE AND EFFECT REGARDLESS OF WHETHER YOUR REMEDIES HEREUNDER ARE DETERMINED TO HAVE FAILED OF THEIR ESSENTIAL PURPOSE. THE ENTIRE LIABILITY OF QTI, QTI'S AFFILIATES AND ITS LICENSORS, AND THE SOLE AND EXCLUSIVE REMEDY OF YOU, FOR ANY CLAIM OR CAUSE OF ACTION ARISING HEREUNDER (WHETHER IN CONTRACT, TORT, OR OTHERWISE) SHALL NOT EXCEED US\$10.

2. **COMPLIANCE WITH LAWS; APPLICABLE LAW.** You agree to comply with all applicable local, international and national laws and regulations and with U.S. Export Administration Regulations, as they apply to the subject matter of this Agreement. This Agreement is governed by the laws of the State of California, excluding California's choice of law rules.

3. **CONTRACTING PARTIES.** If the Materials are downloaded on any computer owned by a corporation or other legal entity, then this Agreement is formed by and between QTI and such entity. The individual accepting the terms of this Agreement represents and warrants to QTI that they have the authority to bind such entity to the terms and conditions of this Agreement.

4. **MISCELLANEOUS PROVISIONS.** This Agreement, together with all exhibits attached hereto, which are incorporated herein by this reference, constitutes the entire agreement between QTI and You and supersedes all prior negotiations, representations and agreements between the parties with respect to the subject matter hereof. No addition or modification of this Agreement shall be effective unless made in writing and signed by the respective representatives of QTI and You. The restrictions, limitations, exclusions and conditions set forth in this Agreement shall apply even if QTI or any of its affiliates becomes aware of or fails to act in a manner to address any violation or failure to comply therewith. You hereby acknowledge and agree that the restrictions, limitations, conditions and exclusions imposed in this Agreement on the rights granted in this Agreement are not a derogation of the benefits of such rights. You further acknowledges that, in the absence of such restrictions, limitations, conditions and exclusions, QTI would not have entered into this Agreement with You. Each party shall be responsible for and shall bear its own expenses in connection with this Agreement. If any of the provisions of this Agreement are determined to be invalid, illegal, or otherwise unenforceable, the remaining provisions shall remain in full force and effect. This Agreement is entered into solely in the English language, and if for any reason any other language version is prepared by any party, it shall be solely for convenience and the English version shall govern and control all aspects. If You are located in the province of Quebec, Canada, the following applies: The Parties hereby confirm they have requested this Agreement and all related documents be prepared in English.