QuickLogic® ArcticLink® III VX/BX CSSPs – Software Integration Guide for Qualcomm Platform, MIPI Video Mode

QuickLogic Application Note 100

Introduction

This application note explains how integrate the QuickLogic ArcticLink III VX/BX Customer Specific Standard Product (CSSP) software for use with the Qualcomm platform, beginning with how to collect panel information from the existing video driver and put it into the QuickLogic DEN-timing .xls file. This document also describes the changes that must be made to the Qualcomm platform video driver for the QuickLogic ArcticLink III VX/BX CSSP initialization, panel initialization and interface timing. The VxApp, a daemon for adapting VEE to ambient light and backlight changes, and the associated Android Java client, a graphical user interface (GUI), are also discussed.

This document includes the following sections:

NOTE: The following example code applies to VX3B2B silicon variant.

Video Driver to .xls File

In the video driver there a routine for setting up the panel information in the **`msm_panel_info`** structure. The following examples are for a MIPI input use case.

```
static struct msm_panel_info pinfo;

 static int __init mipi_video_quickvx_wvga_pt_init(void)
{
   …
        pinfo.xres = 480;
        pinfo.yres = 800;
        ….
              pinfo.type = MIPI_VIDEO_PANEL;
        pinfo.lcdc.h_back_porch = 60;
        pinfo.lcdc.h_front_porch = 20;
        pinfo.lcdc.h_pulse_width = 20;
        pinfo.lcdc.v_back_porch = 10;
        pinfo.lcdc.v_front_porch = 19;
        pinfo.lcdc.v_pulse_width = 5;
        ….
        pinfo.mipi.mode = DSI_VIDEO_MODE;
```

```
            …..
                  pinfo.mipi.vc = 0;
            pinfo.mipi.data_lane0 = TRUE;
            pinfo.mipi.data_lane1 = TRUE;
            …..
            pinfo.mipi.frame_rate = 60;
            ….
      }
```

Referring to the **Den timing-v30.xlsm**, put the data from the video driver into the correct cell.

The following use case is with RGB input. The **type** is **LCDC_PANEL** (i.e., RGB to MIPI/DPI and RGB to LVDS).

1. Input source: Select **C9** to **RGBin**.
2. PCLK: Put value in **clk_rate** to **C9**.
3. HSYNC: Put value in **lcdc.h_pulse_width** to **C16**.
4. HBP: Put value in **lcdc.h_back_porch** to **C17**.
5. HSAA: Put value in **xres** to **C18**.
6. HFP: Put Value in **lcdc.h_front_porch** to **C19**.
7. VFP: Put Value in **lcdc.v_front_porch** to **C22**.
8. VSYNC: Put Value in **lcdc.v_pulse_width** to **C23**.
9. VBP: Put Value in **lcdc.v_back_porch** to **C24**.
10. VSAA: Put Vale in **yres** to **C25**.

The following use case is with MIPI input. The **type** is **MIPI_VIDEO_PANEL** (i.e., MIPI/DPI to LVDS, MIPI/DPI to MIPI/DPI, MIPI/DBI to MIPI/DBI).

1. Input source: Select **C9** to:
   - **MIPI Rx DPI** if **mipi.mode** is **DSI_VIDEO_MODE**
   - **MIPI Rx DBI** if **mipi.mode** is **DSI_CMD_MODE**
2. HSYNC: Put value in **lcdc.h_pulse_width** to **I19**.
3. HBP: Put value in **lcdc.h_back_porch** to **I20**.
4. HSAA: Put value in **xres** to **I21**.
5. HFP: Put Value in **lcdc.h_front_porch** to **I22**.
6. VFP: Put Value in **lcdc.v_front_porch** to **I25**.
7. VSYNC: Put Value in **lcdc.v_pulse_width** to **I26**.
8. VBP: Put Value in **lcdc.v_back_porch** to **I27**.
9. VSAA: Put Value in **yres** to **I28**.
10. FrameRate: Put Value in **mipi.frame_rate** to **I9**.
11. MIPI Lanes: Put the number of **mipi.data_lanex** enabled (**TRUE**) to **I12**.
    For example in the above routine put **2** to **I12** for two MIPI lanes.
12. Video channel number: Put Value in **vc** to **I34** and **I35**.

Video Driver Porting

ArcticLink III VX/BX CSSP Initialization

The ArcticLink III VX/BX CSSP must be initialized to pass video data to the bootloader and display the boot logo. To initialize the ArcticLink III VX/BX CSSP:

1. Apply **`sysclk`** (optional).
2. Toggle the reset.
3. Initialize the ArcticLink III VX/BX CSSP registers.
4. Initialize the ArcticLink III VX/BX CSSP registers through the I^2C bus in kernel.
   a. Apply for RGB input use cases.
      - RGB to MIPI/DPI
      - RGB to LVDS
   b. Register the I^2C address. The ArcticLink III VX/BX CSSP I^2C address is 0x64 (depending on GPIO pin). Define the **`i2c_board_info`** structure and call **`i2c_register_board_info()`** with the correct bus number.

```
static struct i2c_board_info i2c_quickvx_board_info[] __initdata = {
    {
    I2C_BOARD_INFO("i2c_quickvx", 0x64),
    },
};
```

   c. Use standard I^2C driver initialization in the video driver.

```
static struct i2c_device_id i2c_quickvx_idtable[] = {
  { "i2c_quickvx", 0 },
  { }
};
MODULE_DEVICE_TABLE(i2c, i2c_quickvx_idtable);
static struct i2c_driver i2c_quickvx_driver = {
  .driver = {
    .owner = THIS_MODULE,
    .name  = "i2c_quickvx",
  },
  .id_table = i2c_quickvx_idtable,
  .probe    = i2c_quickvx_probe,
};
static int __init mipi_quickvx_lcd_init(void)
{
….
ret = i2c_add_driver(&i2c_quickvx_driver);
….
}
```

d. I²C read/write routines are provided by QuickLogic.

```
static struct i2c_client *I2cClient;
int ql_i2c_read(uint32 addr, uint32 *val, uint32 data_size);
int ql_i2c_write(long addr, long val, int data_size);
```

e. Send initialization data through the I²C interface.

```
struct chip_init_data vx3_init_data[] = {
    //from init script
    { 0x700,0x30900080},
    …..
    { 0x608,0x50F},
    };
```

f. On some platforms the I²C bus must specifically set up the GPIO. For example, on Dragon board the following must be called:

```
int msm_cam_gpio_tbl[] = {
    32,/*CAMIF_MCLK*/
    47,/*CAMIF_I2C_DATA*/
    48,/*CAMIF_I2C_CLK*/
    105,/*STANDBY*/
};

enum msm_cam_stat{
    MSM_CAM_OFF,
    MSM_CAM_ON,
};

static int config_gpio_table(enum msm_cam_stat stat)
{
    int rc = 0, i = 0;
    if (stat == MSM_CAM_ON) {
       for (i = 0; i < ARRAY_SIZE(msm_cam_gpio_tbl); i++) {
          rc = gpio_request(msm_cam_gpio_tbl[i], "CAM_GPIO");
          if (unlikely(rc < 0)) {
             pr_err("%s not able to get gpio\n", __func__);
             for (i--; i >= 0; i--)
                gpio_free(msm_cam_gpio_tbl[i]);
             break;
          }
       }
    } else {
       for (i = 0; i < ARRAY_SIZE(msm_cam_gpio_tbl); i++)
          gpio_free(msm_cam_gpio_tbl[i]);
    }
    return rc;
}
```

5. Initialize the ArcticLink III VX/BX CSSP registers through the MIPI DSI interface in kernel.

   a. Apply for MIPI input use cases.

      - MIPI/DPI to LVDS
      - MIPI/DPI to MIPI/DPI
      - MIPI/DBI to MIPI/DBI

   b. Each MIPI host has a different MIPI display driver. Find an existing MIPI display driver and modify it. Normally the **`panel_on`** routine is the place to insert VX/BX MIPI initialization codes.

      The following example is for a Qualcomm-based MIPI driver.

      QuickLogic-specific Generic Command Packet structure for register writes.

```
static char ql_csr_wr_payload[9] =
{QL_MIPI_VENDOR_ID_1, QL_MIPI_VENDOR_ID_2, QL_MIPI_COMMAND_CSR_WRITE,
0x0, 0x0,  /* address 16bits */
0x0, 0x0, 0x0, 0x0}; /* data max 32bits */

static struct dsi_cmd_desc ql_generic_csr_wr_cmd[] = {
{DTYPE_GEN_LWRITE, 1, 0, 0, 0,
sizeof(ql_csr_wr_payload), ql_csr_wr_payload}
};
```

   c. To Tx MIPI packet:

```
//send the generic write csr address and offset first
// return number of tx command.
mipi_dsi_cmds_tx(mfd, &quickvx_tx_buf, ql_generic_csr_offset_cmd,
ARRAY_SIZE(ql_generic_csr_offset_cmd));
```

   d. To Rx MIPI packet:

```
// return number of bytes.
ret = mipi_dsi_cmds_rx(mfd, &quickvx_tx_buf, &quickvx_rx_buf,
&ql_generic_csr_read_cmd, data_size);
```

   e. Send the init data through MIPI interface.

   f. Put the init data from the script into array.

```
struct chip_init_data vx3_init_data[] = {
//from init script
{ 0x700,0x30900080},
…..
{ 0x608,0x50F},
};
```

   g. Call the init routine in Panel On () function.

```
static int quickvx_lcd_on(struct platform_device *pdev)
{
….
 ql_chip_init();
….
}
```

Panel Data Structures

- MIPI-specific panel data structure

```
pinfo.mipi.mode = DSI_VIDEO_MODE;
pinfo.mipi.traffic_mode = DSI_NON_BURST_SYNCH_PULSE;
pinfo.mipi.dst_format = DSI_VIDEO_DST_FORMAT_RGB888;
pinfo.mipi.vc = 0;
pinfo.mipi.rgb_swap = DSI_RGB_SWAP_BGR;
pinfo.mipi.data_lane0 = TRUE;
pinfo.mipi.data_lane1 = TRUE;
pinfo.mipi.t_clk_post = 0x04; /* 0xc0, DSI_CLKOUT_TIMING_CTRL */
pinfo.mipi.t_clk_pre = 0x17; /* 0xc0, DSI_CLKOUT_TIMING_CTRL */
```

- Display-specific panel data structure

```
pinfo.xres = 480;
pinfo.yres = 800;
pinfo.type = MIPI_VIDEO_PANEL;
pinfo.pdest = DISPLAY_1;
pinfo.bpp = 24;
pinfo.lcdc.h_back_porch = 2; //8;
pinfo.lcdc.h_front_porch = 10; //11;
pinfo.lcdc.h_pulse_width = 2; //16;
pinfo.lcdc.v_back_porch = 11; //15;
pinfo.lcdc.v_front_porch = 18; //11;
pinfo.lcdc.v_pulse_width = 5; //1;
```

Panel Initialization

1. Initialization commands for the panels pass through the ArcticLink III VX/BX CSSP. For example:
   - MIPI/DPI to MIPI/DPI
   - MIPI/DBI to MIPI/DBI
2. Panel On – Sends out panel on commands after ArcticLink III VX/BX CSSP register initializes. Panel commands may differ.

```
static struct dsi_cmd_desc quickvx_display_on_cmds[] = {
{DTYPE_DCS_WRITE, 1, 0, 0, 150, sizeof(exit_sleep), exit_sleep},
{DTYPE_DCS_WRITE, 1, 0, 0, 0, sizeof(display_on), display_on}
};
```

3. Panel Off.

```
static struct dsi_cmd_desc quickvx_display_off_cmds[] = {
{DTYPE_DCS_WRITE1, 1, 0, 0, 10, sizeof(display_off), display_off},
{DTYPE_DCS_WRITE1, 1, 0, 0, 120, sizeof(enter_sleep), enter_sleep}
```

Backlight Control

Apply for a system that uses the QuickLogic PWM or IBC. Modify the backlight control routine in the video driver.

Suspend/Resume

If power to the ArcticLink III VX/BX CSSP stops, re-initialize the ArcticLink III VX/BX CSSP registers upon resume. The best place to put in suspend/resume handling is in the video driver suspend/resume routines.The following is a MIPI to MIPI use case example:

On suspend:

1. Send MIPI a command to Set Backlight to 0.
2. Send the MIPI panel a command (such as Set Display Off (0x28) and Enter Sleep Mode (0x10)).
3. Power down the ArcticLink III VX/BX CSSP.
4. Stop **`sysclk`** (optional).

On resume:

1. Power on the ArcticLink III VX/BX CSSP.
2. Start **`sysclk`**.
3. Toggle the reset.
4. GPIO.
5. Re-initialize the ArcticLink III VX/BX CSSP registers.
6. Send the MIPI panel a command (such as Exit Sleep Mode (0x11) or Set Display On (0x29)).
7. Set backlight to normal.

VxApp Interface

The VxApp interface is for a system that does not have I^2C connected to the ArcticLink III VX/BX CSSP (for example, MIPI/DPI to MIPI/DPI) or a system that does not support Linux kernel I^2C development interface.

To support a specific operation expose the **`Sysfs`** structure. For example, the ArcticLink III VX/BX CSSP register read/write access:

```
static struct device_attribute mipi_quickvx_attributes[] = {
    __ATTR(driver_info, 0444, show_mipi_quickvx_driver_info, NULL),
    __ATTR(read_reg, 0644, show_mipi_quickvx_read, store_mipi_quickvx_read),
    __ATTR(write_reg, 0644, NULL, store_mipi_quickvx_write),
};
```

QuickLogic provides function implementation details. For example, to displays the version number:

```
static ssize_t show_mipi_quickvx_driver_info(struct device *dev,
struct device_attribute *attr, char *buf)
{
    return snprintf(buf, PAGE_SIZE, "%s \n", QL_MIPI_DEVICE_NAME);
}
```

Example Codes

The following reference codes are located in the **driver source code** directory.

- RGB to MIPI/DPI
  - Rgb2mipi_quickvx.c
- RGB to LVDS
- MIPI/DPI to LVDS
- MIPI/DPI to MIPI/DPI - For Qualcomm msm8x60 Dragon System:
  - Mipi_quickvx.c
  - Mipi_quickvx_video_wvga_pt.c
- MIPI/DBI to MIPI/DBI

NOTE: THE FOLLOWING SECTIONS APPLY TO VX VARIANTS OF THE ARCTICLINK III VX ONLY.

VxApp

VxApp Features

The VxApp has the following features:

- Communicates with the ArcticLink III VX/BX CSSP through I^2C dev or video drivers in the system.
- Communicates with Android light sensor and backlight control services.
  - Receives events from Android on ambient light and display brightness level changes and adjusts VEE operation.
  - Changes the display brightness level through Android.
  - Polls the ambient light level at a defined interval through I^2C drivers independent of Android light sensor service.
- Implements a socket server; any Android application can talk to VxApp through a socket client interface.
- Ability to read/write registers in the ArcticLink III VX/BX CSSP using the Command line mode.
  - Good hardware debugging tool.
  - Useful during the calibration process.

VxApp Porting

Most of the device-dependent variables are defined in the device-specific header files including:

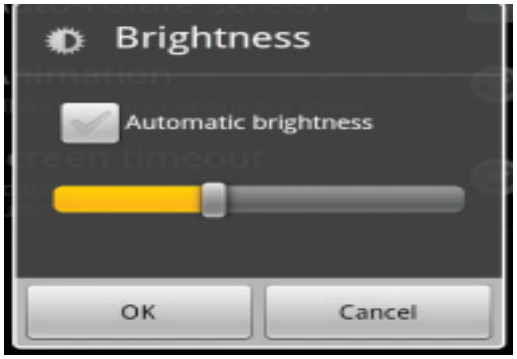
- ArcticLink III VX/BX CSSP specific – Type of ArcticLink III VX/BX CSSP in the system, such as vx3b3b, vx5a1d, etc.
- I^2C interface for ArcticLink III VX/BX CSSP communication – Linux kernel I^2C dev-interface file (bus) where the ArcticLink III VX/BX CSSP is located and the I^2C system address.
- Video driver **`sysfs`** interface for ArcticLink III VX/BX CSSP communication – Video driver **`sysfs`** path. For example **`#define MIPI_SYSFS_PATH "/sys/devices/platform/mipi_quickvx.513/"`**.
- System ambient light sensor – Ambient light sensor **`sysfs`** path, device driver direct access, or input event path (i.e., **`#define AL_INPUT_EVENT_PATH "/dev/input/event4"`**) and **`min/dark/indoor/outdoors/max lux`** value.
- System backlight control – Backlight control **`sysfs`** path and min/max value for read/write to the **`sysfs`** path.
- Display specifications information – Image width and height, Hsync front porch, width and back porch, Vsync front porch, width and back porch, MIPI video/command mode and RGB mode (RGB565/666/888)
- Calibration table – Calibrated on-site by QuickLogic.

Java Application

An Android Java application is used to enable/disable the auto brightness and VEE strength control, and to support internet connection sharing (ICS). There are two versions:

- The lite version uses the Standard Android Brightness Preference GUI. The changes to the system are minimal.
- The powerful version uses the QuickLogic GUI which offers more control, but involves more changes to the system.

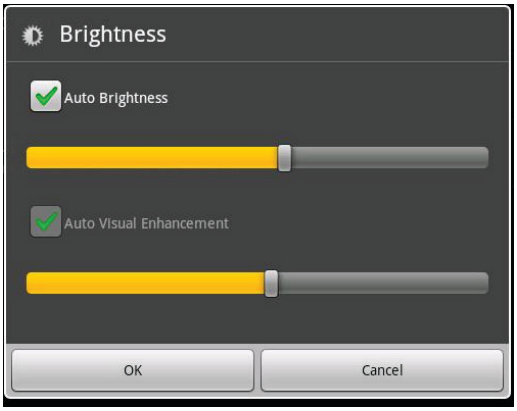
Standard Android Brightness Preference GUI – Lite Version



To use the Standard Android Brightness Preference GUI:

1. Disable the Android automatic brightness control in **`PowerManagerService.java`**.
2. Load the background application **`VEEServiceLite.java`**.
3. Monitor the Android global variable **`Settings.System.SCREEN_BRIGHTNESS_MODE`**.
4. If set, send a command to VxApp to start controlling backlight and VEE strength for different ambient light.
5. If clear, stop all backlight and VEE strength controls.

QuickLogic GUI – Powerful Version



To use the QuickLogic GUI:

The **`src`** folder contains all the files needed for the changes. Some files are for replacing an entire file in the Android source tree (**`packages/apps/Settings`**), while some are for modifying the original file.

1. Add new intents in **`AndroidManifest.xml.`**

```
<service android:enabled="true" android:name="VEEServicee">
    <intent-filter>
       <action android:name = "com.android.settings.VEEService"/>
    </intent-filter>
</service>
<uses-permission android:name="android.permission.RECEIVE_BOOT_COMPLETED" />
```

```
<receiver android:name="VEEReceiver">
    <intent-filter>
       <action android:name ="android.intent.action.BOOT_COMPLETED"/>
    </intent-filter>
</receiver>
```

2. Use the **BrightnessPreference.java** class to replace the original class
3. Use the **VeeAppControl.java** class to control communication to VxApp daemon.
4. Use the **NavtiveVxControl.java** class to do native socket communication to VxApp daemon.
5. Use the **VEEService.java** service class to receive the ambient light sensor event.
6. Use the **VEEReceiver.java** receiver for the **BOOT_COMPLETED** event to start the **VEEService** on boot.

Comparable to the Standard Android Brightness Preference GUI, the QuickLogic GUI includes the following features.

- Implementation for **settings.apk**.
  - Auto Brightness automatically adjusts the backlight based on the current ambient light.
  - Provides individual user preference setting in addition to the Auto Brightness feature, adjusting brightness even with Auto Brightness enabled.
  - Auto Visual Enhancement automatically adjusts VEE compensation based on current ambient light and backlight level.
  - Provides individual user preference setting in addition to the auto VEE compensation, adjusting VEE compensation even with Auto Visual Enhancement enabled.
  - Auto Visual Enhancement with manual backlight control can be selected (with Auto Brightness deselected). In this mode, VEE compensation is adjusted based on the current ambient light level and user's selected backlight setting.
- Implements a socket client to communicate user selections with VxApp daemon.

Contact Information

Phone: (408) 990-4000 (US)
(647) 367-1014 (Canada)
+(44) 1932-21-3160 (Europe)
+(886) 26-603-8948 (Taiwan)
+(86) 21-5179-8474 (China)
E-mail: info@quicklogic.com
Sales: America-sales@quicklogic.com
Europe-sales@quicklogic.com
Asia-sales@quicklogic.com
Japan-sales@quicklogic.com
Support: www.quicklogic.com/support
Internet: www.quicklogic.com

Revision History

Revision	Date	Originator and Comments
A	September 2012	Sunny Lai and Kathleen Bylsma

Notice of Disclaimer

QuickLogic is providing this design, product or intellectual property "as is." By providing the design, product or intellectual property as one possible implementation of your desired system-level feature, application, or standard, QuickLogic makes no representation that this implementation is free from any claims of infringement and any implied warranties of merchantability or fitness for a particular purpose. You are responsible for obtaining any rights you may require for your system implementation. QuickLogic shall not be liable for any damages arising out of or in connection with the use of the design, product or intellectual property including liability for lost profit, business interruption, or any other damages whatsoever. QuickLogic products are not designed for use in life-support equipment or applications that would cause a life-threatening situation if any such products failed. Do not use QuickLogic products in these types of equipment or applications.

QuickLogic does not assume any liability for errors which may appear in this document. However, QuickLogic attempts to notify customers of such errors. QuickLogic retains the right to make changes to either the documentation, specification, or product without notice. Verify with QuickLogic that you have the latest specifications before finalizing a product design.

Copyright and Trademark Information