QuickLogic® ArcticLink® III VX/BX CSSPs – Software Integration Guide for the MTK Platform, RGB to MIPI-DSI

QuickLogic Application Note 102

Introduction

This application note explains how integrate the QuickLogic ArcticLink III VX/BX Customer Specific Standard Product (CSSP) software for use with the MediaTek (MTK) platform. This document describes the changes that must be made to the MTK platform video driver for the QuickLogic ArcticLink III VX/BX CSSP initialization, panel initialization and interface timing. The VxApp, a daemon for adapting the QuickLogic Visual Enhancement Engine (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:

- **Video Driver Porting** on page 1
- **VxApp** on page 9
- **Java Application** on page 10

NOTE: The following example code applies to VX5A2B and VX5A3B silicon variants.

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. Make the following bootloader code changes.
   a. Add **`quickvx_init()`**.

```
static int quickvx_Init()
{
    UINT32 val = 0x8899;

        //apply clock
        mt_set_gpio_mode(GPIO_VX_CLK,GPIO_VX_CLK_MODE);
        mt_set_clock_output(GPIO_VX_CLK_PIN_CLK, GPIO_VX_PIN_FREQ,2);
        MDELAY(200);

        //reset
```

```
        mt_set_gpio_mode(GPIO_VX_RST,GPIO_VX_RST_MODE);
        mt_set_gpio_out(GPIO_VX_RST,GPIO_OUT_ONE);
        MDELAY(10);
        mt_set_gpio_out(GPIO_VX_RST,GPIO_OUT_ZERO);
        MDELAY(20);
        mt_set_gpio_out(GPIO_VX_RST,GPIO_OUT_ONE);
        MDELAY(5);

    ql_i2c_read(0x4fe, &val, 2);
    QL_DBG("VEE ID 0x%x\n", val);

    if(val != QL_ID)
        return -1;

#ifndef QL_VX_INIT_EXTERNAL
    //init code
    ql_init_table(ql_initialization_setting,
sizeof(ql_initialization_setting)/ sizeof(struct QL_VX_INIT_INFO));
#endif

    QL_DBG("quickvx_Init ok\n");

        MDELAY(10);
        return 0;
}
```

b. Call **`quickvx_init()`** in **`lcm_init()`**.

```
static void lcm_init(void)
{
    int ret = 0;
    QL_DBG("+++\n");

#ifdef BUILD_UBOOT
    ret = QL_I2C_BOOTLOADER_INIT();
    QL_DBG("Init I2C0 = %d\n",ret);
#endif

    quickvx_Init();

#ifndef QL_PANEL_INIT_EXTERNAL
//Porting: panel on command call here.
    ql_i2c_4_MIPI_panel_write(DTYPE_DCS_WRITE,
    QL_VX_LCD_VC,sizeof(lcm_init_cmd_1), lcm_init_cmd_1);
    ql_i2c_4_MIPI_panel_write(DTYPE_DCS_WRITE,
    QL_VX_LCD_VC,sizeof(lcm_init_cmd_2), lcm_init_cmd_2);
#endif
}
```

5. In some bootloader versions the I^2C clock is fixed at 1.2 MHz. If so, modify the bootloader I^2C driver to 400 kHz as follows:

```
U32 i2c0_v1_set_speed (unsigned long clock, I2C_SPD_MODE mode, unsigned long
khz)
{
```

```
…
    if (mode == HS_MODE) {
       tmp  = __raw_readw(MT_I2C_HS) & ((0x7 << 12) | (0x7 << 8));
       tmp  = (sample_cnt_div & 0x7) << 12 | (step_cnt_div & 0x7) << 8 | tmp;
       __raw_writew(tmp, MT_I2C_HS);
       I2C_SET_HS_MODE(1);
    }
    else {
       tmp  = __raw_readw(MT_I2C_TIMING) & ~((0x7 << 8) | (0x3f << 0));

//tmp  = (0/*sample_cnt_div*/ & 0x7) << 8 | (5/*step_cnt_div*/ & 0x3f) << 0 |
tmp;
       tmp  = (sample_cnt_div & 0x7) << 8 | (step_cnt_div & 0x3f) << 0 | tmp;

       __raw_writew(tmp, MT_I2C_TIMING);
       I2C_SET_HS_MODE(0);
    }
….
}
```

6. The original I²C write supports 8 bytes of data. However, the ArcticLink III VX/BX CSSP needs 10 bytes of data for register access. Make the following change:

```
U32 i2c1_v1_write (U8 chip, U8 *buffer, int len)
{
    ….
    /* polling mode : see if transaction complete */
    while (1) {
       status = I2C_INTR_STATUS;

       if ( status & I2C_TRANSAC_COMP) {
          ret = 0;
          ret_code = I2C_OK;
          break;
       }
    ….
       else if (time_out_val > 100000) {
          ret = 3;
          ret_code = I2C_WRITE_FAIL_TIMEOUT;
          printf("[i2c1_write] transaction timeout:%d\n", time_out_val);
          break;
       }
       time_out_val++;
       if( (time_out_val == 5000) && (data_left)){
          i2c1_v1_write_left_data((char *)&buffer[I2C_FIFO_SIZE],data_left);
       }
    }
….
    return ret_code;
}
```

7. Initialize the ArcticLink III VX/BX CSSP registers through the I²C bus in kernel.

   a. Register the I²C address. The ArcticLink III VX/BX CSSP I²C 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 __initdata i2c_quickvx={
I2C_BOARD_INFO("quickvx", (QL_I2C_ADDR))};
```

   b. Use standard I²C driver initialization in the **ql_vx_module_init()**.

```
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,
};
int __init ql_vx_module_init(void)
{
    int r = 0;
       QL_DBG("+++ ql_vx enter\n");

#ifndef BUILD_UBOOT
    //register i2c for kernel.
    i2c_register_board_info(QL_I2C_REGISTER_BUSNUM, &i2c_quickvx, 1);
    r= i2c_add_driver(&i2c_quickvx_driver);
    QL_DBG("i2c_add_driver ret = %d\n", r);
    if (r)
       printk(KERN_ERR
          "%s: i2c_add_driver failed!\n", __func__);
#endif
    return r;
}
module_init(ql_vx_module_init);
```

   c. 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);
```

   d. Send initialization data through the I²C interface.

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

Panel Data Structures

MIPI-specific Panel Data Structure is defined in the display driver **`lcm_get_params()`**. Use this section to change display porch values.

```
static void lcm_get_params(LCM_PARAMS *params)
{
    memset(params, 0, sizeof(LCM_PARAMS));

    params->type   = LCM_TYPE_DPI;
    params->ctrl   = LCM_CTRL_NONE;
    params->width  = FRAME_WIDTH;
    params->height = FRAME_HEIGHT;
    params->io_select_mode = 0;

//Porting: panel specification here
    params->dpi.MIPI_pll_clk_ref  = 0;
    params->dpi.MIPI_pll_clk_div1 = 44;
    params->dpi.MIPI_pll_clk_div2 = 4;
    params->dpi.dpi_clk_div       = 2;
    params->dpi.dpi_clk_duty      = 1;

    params->dpi.clk_pol           = LCM_POLARITY_RISING; //Phil, check this.
    params->dpi.de_pol            = LCM_POLARITY_FALLING;
    params->dpi.vsync_pol         = LCM_POLARITY_FALLING;
    params->dpi.hsync_pol         = LCM_POLARITY_FALLING;

    params->dpi.hsync_pulse_width = 128;
    params->dpi.hsync_back_porch  = 200;
    params->dpi.hsync_front_porch = 72;
    params->dpi.vsync_pulse_width = 6;
    params->dpi.vsync_back_porch  = 22;
    params->dpi.vsync_front_porch = 3;

    params->dpi.format            = LCM_DPI_FORMAT_RGB888;
    params->dpi.rgb_order         = LCM_COLOR_ORDER_RGB;
    params->dpi.is_serial_output  = 0;

    params->dpi.intermediat_buffer_num = 2;

    params->dpi.io_driving_current = LCM_DRIVING_CURRENT_8MA |
    LCM_DRIVING_CURRENT_4MA | LCM_DRIVING_CURRENT_2MA;
}
```

Panel MIPI Commands

For panels with MIPI commands that are sent through the ArcticLink III VX/BX CSSP in various routines (such as **lcm_init**, **lcm_suspend**, and **lcm_resume**), use the QuickLogic-provided **ql_i2c_4_MIPI_panel_write()** to send out DCS and generic commands.

For DCS commands:

```
ql_i2c_4_MIPI_panel_write(DTYPE_DCS_WRITE,
QL_VX_LCD_VC,sizeof(lcm_init_cmd_1), lcm_init_cmd_1);
```

For generic commands:

```
ql_i2c_4_MIPI_panel_write(DTYPE_GEN_WRITE,
QL_VX_LCD_VC,sizeof(lcm_init_cmd_1), lcm_init_cmd_1);
```

Backlight (Brightness) Control

To establish backlight control:

1. In the example driver, apply for system that sends MIPI commands to the panel for backlight control.

```
static void lcm_setbacklight(unsigned int level)
{
    unsigned int default_level = 0;
    unsigned int mapped_level = 0;

    //for LGE backlight IC mapping table
    if(level > 255)
          level = 255;

    if(level >0) {
          mapped_level = default_level+(level)*(255-default_level)/(255);
    }
    else
          mapped_level=0;

    // Refresh value of backlight level.
    lcm_setbacklight_cmd_1[1] = mapped_level;

    ql_i2c_4_MIPI_panel_write(DTYPE_DCS_WRITE,
QL_VX_LCD_VC,sizeof(lcm_setbacklight_cmd_1), lcm_setbacklight_cmd_1);
    ql_i2c_4_MIPI_panel_write(DTYPE_DCS_WRITE,
QL_VX_LCD_VC,sizeof(lcm_setbacklight_cmd_2), lcm_setbacklight_cmd_2);
    ql_i2c_4_MIPI_panel_write(DTYPE_DCS_WRITE,
QL_VX_LCD_VC,sizeof(lcm_setbacklight_cmd_3), lcm_setbacklight_cmd_3);

    QL_DBG("mapped_level = %d\n", mapped_level);

}
```

2. (OPTIONAL) 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.

On suspend:

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

```
static void lcm_suspend(void)
{
//disable output pixel clk
    QL_DBG("+++\n");

#ifndef BUILD_UBOOT
    ql_suspend_resume_in_progress = TRUE;
#endif

//Porting: panel sleep command send here.
    ql_i2c_4_MIPI_panel_write(DTYPE_DCS_WRITE,
QL_VX_LCD_VC,sizeof(lcm_init_cmd_1), lcm_init_cmd_1);
    ql_i2c_4_MIPI_panel_write(DTYPE_DCS_WRITE,
QL_VX_LCD_VC,sizeof(lcm_init_cmd_2), lcm_init_cmd_2);

    ql_init_table(ql_standby_setting, sizeof(ql_standby_setting) /
sizeof(struct QL_VX_INIT_INFO));

}
```

On resume:

1. Power on the ArcticLink III VX/BX CSSP. (See the QuickLogic *ArcticLink III VXxxxx/BXxxxx CSSP Data Sheet* for sequencing guidelines.)
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)).

```
static void lcm_resume(void)
{
//enable output pixel clk
    QL_DBG("+++\n");
    //if resume got call instand of lcm_init() on resume.
    lcm_init();
#ifndef BUILD_UBOOT
    ql_resume_count++;
    ql_suspend_resume_in_progress = FALSE;
#endif
}
```

VxApp Interface

The VxApp interface is for a system that does not have I²C connected to the ArcticLink III VX/BX CSSP. For example, MIPI/DPI to MIPI/DPI or a system that does not support Linux kernel I²C 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),
    __ATTR(write_bulk, 0644, NULL, store_mipi_quickvx_write_bulk),
    __ATTR(i2c_raw, 0644, show_i2c_raw, store_i2c_raw),
    __ATTR(dsi_raw, 0644, show_dsi_raw, store_dsi_raw),
    __ATTR(resume_count, 0444, show_resume_count, NULL),
    __ATTR(force_screen_update, 0644, show_force_screen_update,
store_force_screen_update),
};
```

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

Example drivers will be provided. The following sections of the example driver must be customized if it is to be used as-is.

```
//Porting: gpio details here.
#define GPIO_VX_CLKGPIO76
#define GPIO_VX_CLK_MODEGPIO_MODE_04
#define GPIO_VX_CLK_PIN_CLK CLK_OUT4
#define GPIO_VX_PIN_FREQ0x4//CLK_SRC_F26M
#define GPIO_VX_RSTGPIO50
#define GPIO_VX_RST_MODEGPIO_MODE_00

//Quicklogic routines
//i2c address
#define QL_I2C_ADDR 0x64
#define QL_I2C_BOOTLOADER_ADDR(QL_I2C_ADDR<<1)
//Porting: i2c details here.
//i2c bus number for kernel
#define QL_I2C_REGISTER_BUSNUM 0
//i2c operation for BUILD_UBOOT.
#define QL_I2C_BOOTLOADER_INIT i2c0_v1_init
#define QL_I2C_BOOTLOADER_WRITE i2c0_v1_write
#define QL_I2C_BOOTLOADER_READ i2c0_v1_read
```

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:

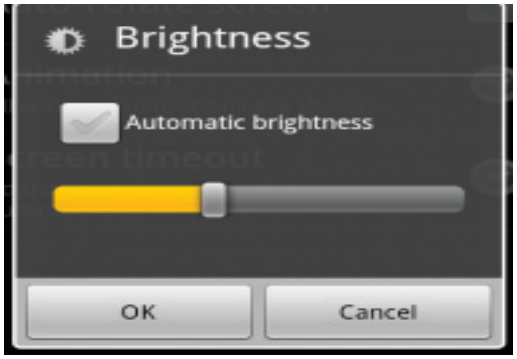
- ArcticLink III VX/BX CSSP specific – Type of ArcticLink III VX/BX CSSP in the system, such as VX3B3B, VX5A1D etc.
- I^2C interface for ArcticLink III VX/BX CSSP communication – Linux kernel I^2C dev-interface file (bus) where the ArcticLink III VX/BX CSSP is located and the I^2C system address.
- Video 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.

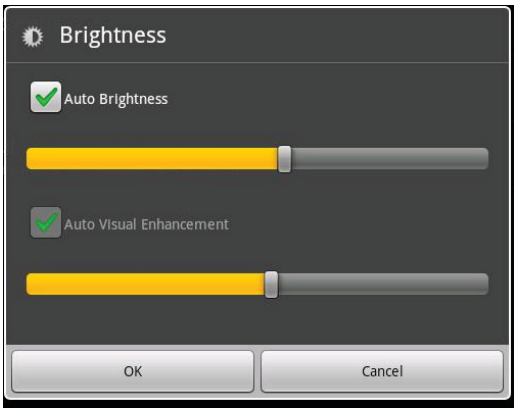
Standard Android Brightness Preference GUI – Lite Version



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="VEEService">
    <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
B	November 2012	Paul Karazuba and Kathleen Bylsma

Notice of Disclaimer

Copyright and Trademark Information