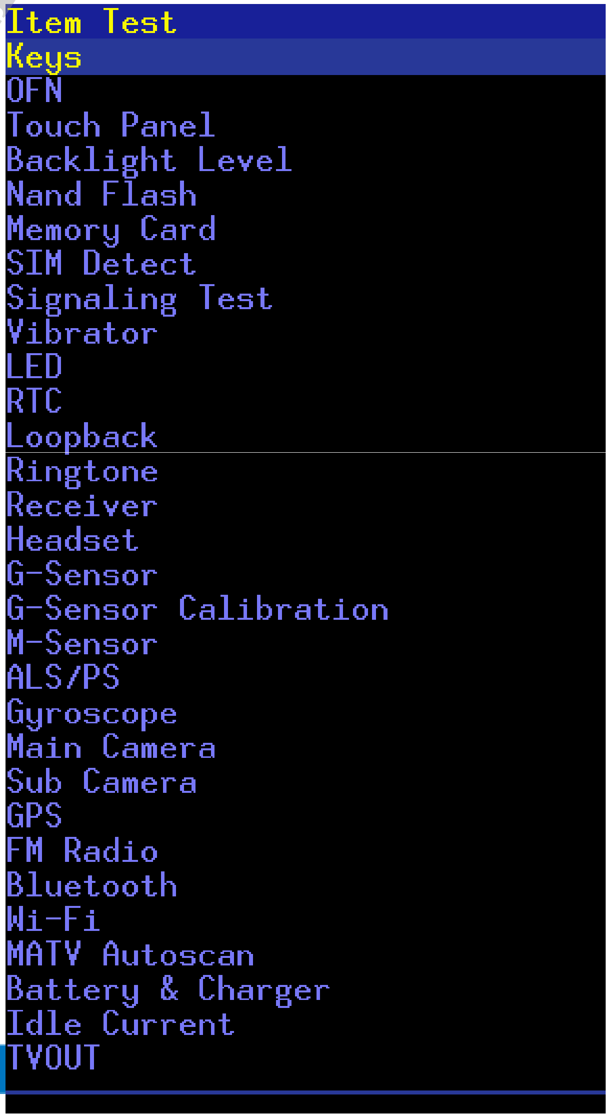
LCM ATA implement For L&KK2

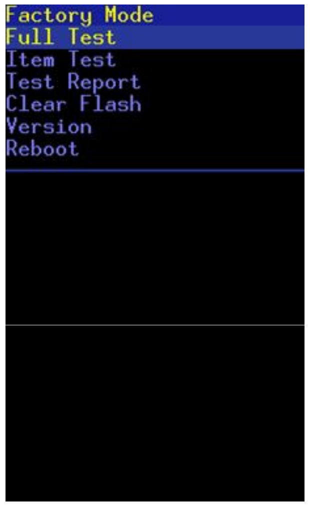
Overview

- In factory mode, add one test item:
- When tap this test item, factory mode call framebuffer driver IOCTL to do:
  - Memset() FB to Green picture and write some bytes to LCM
  - Read back the bytes that have written to LCM panel and compare these data

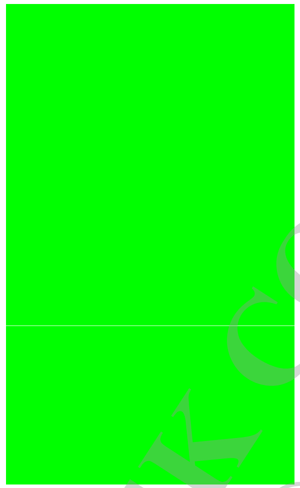


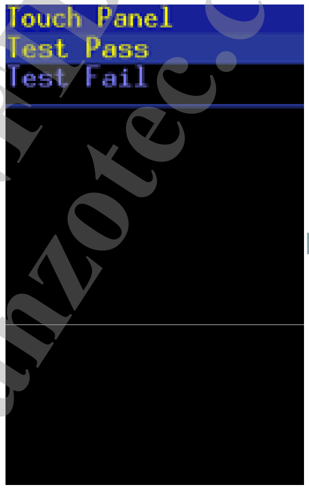
```
Item Test
Keys
OFN
Touch Panel
Backlight Level
Nand Flash
Memory Card
SIM Detect
Signaling Test
Vibrator
LED
RTC
Loopback
Ringtone
Receiver
Headset
G-Sensor
G-Sensor Calibration
M-Sensor
ALS/PS
Gyroscope
Main Camera
Sub Camera
GPS
FM Radio
Bluetooth
Wi-Fi
MATV Autoscan
Battery & Charger
Idle Current
TVOUT
```

- We also add test item to Factory mode Auto test

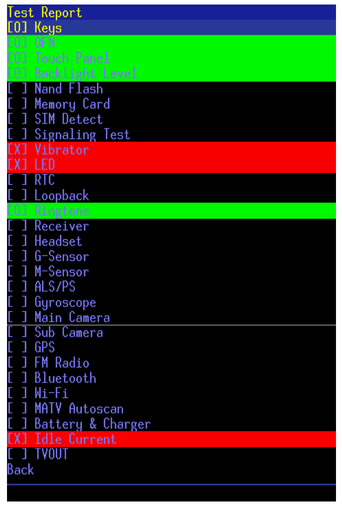


```
Factory Mode
Full Test
Item Test
Test Report
Clear Flash
Version
Reboot
```





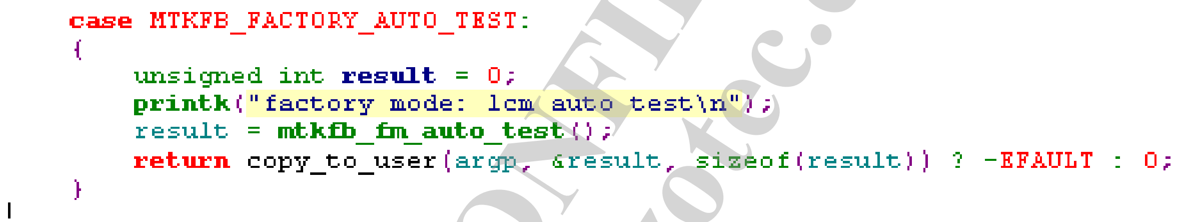
```
Touch Panel
Test Pass
Test Fail
```



```
Test Report
[O] Keys
[O] OFN
[O] Touch Panel
[O] Backlight Level
[ ] Nand Flash
[ ] Memory Card
[ ] SIM Detect
[ ] Signaling Test
[X] Vibrator
[X] LED
[ ] RTC
[ ] Loopback
[O] Ringtone
[ ] Receiver
[ ] Headset
[ ] G-Sensor
[ ] M-Sensor
[ ] ALS/PS
[ ] Gyroscope
[ ] Main Camera
[ ] Sub Camera
[ ] GPS
[ ] FM Radio
[ ] Bluetooth
[ ] Wi-Fi
[ ] MATV Autoscan
[ ] Battery & Charger
[X] Idle Current
[ ] TVOUT
Back
```

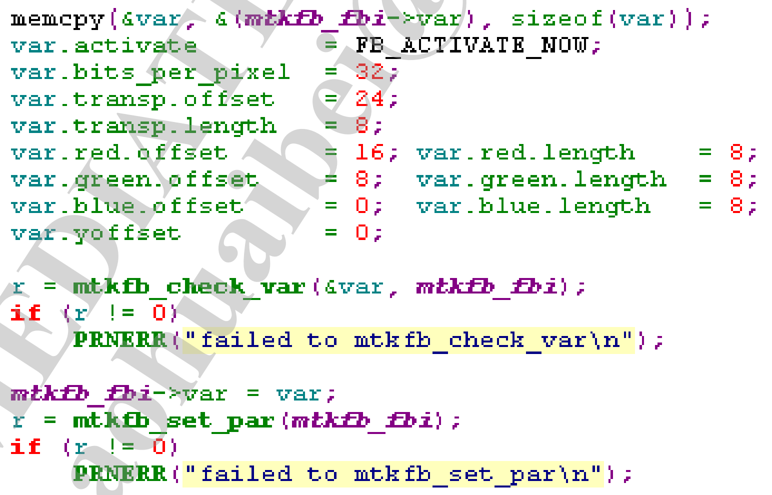
FB driver IOCTL implement

- Mtkfb.c



```
case MTKFB_FACTORY_AUTO_TEST:
{
    unsigned int result = 0;
    printk("factory mode: lcm auto test\n");
    result = mtkfb_fm_auto_test();
    return copy_to_user(argp, &result, sizeof(result)) ? -EFAULT : 0;
}
```

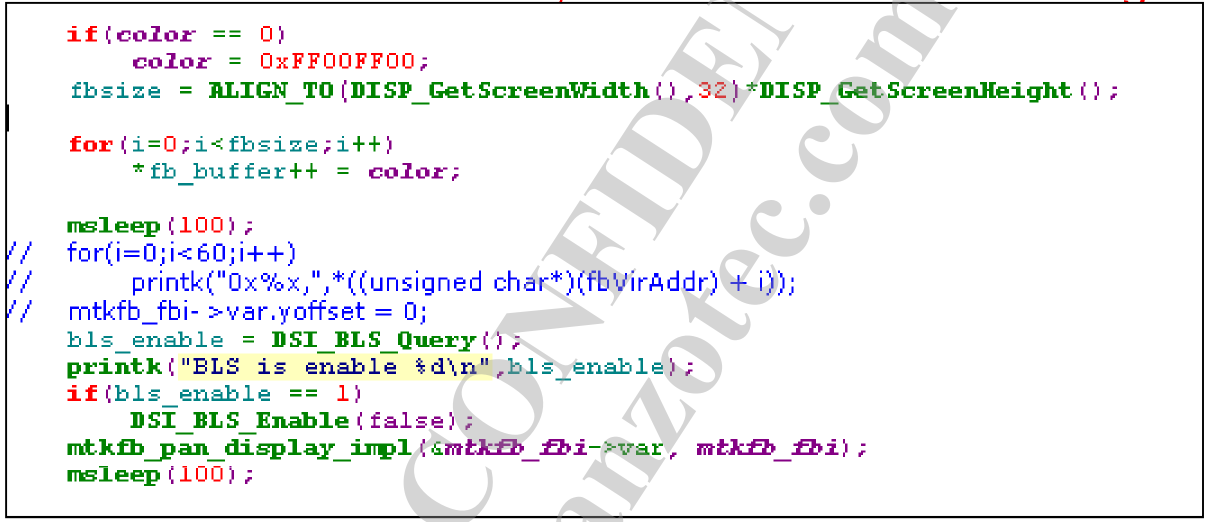
- In mtkfb_fm_auto_test(), we could divide these code into 3 parts:
  - 1: Change FB format from RGB565 to ARGB8888



```
memcpy(&var, &(mtkfb_fbi->var), sizeof(var));
var.activate          = FB_ACTIVATE_NOW;
var.bits_per_pixel    = 32;
var.transp.offset     = 24;
var.transp.length     = 8;
var.red.offset        = 16; var.red.length    = 8;
var.green.offset      = 8;  var.green.length  = 8;
var.blue.offset       = 0;  var.blue.length   = 8;
var.yoffset           = 0;

r = mtkfb_check_var(&var, mtkfb_fbi);
if (r != 0)
    PRNERR("failed to mtkfb_check_var\n");

mtkfb_fbi->var = var;
r = mtkfb_set_par(mtkfb_fbi);
if (r != 0)
    PRNERR("failed to mtkfb_set_par\n");
```

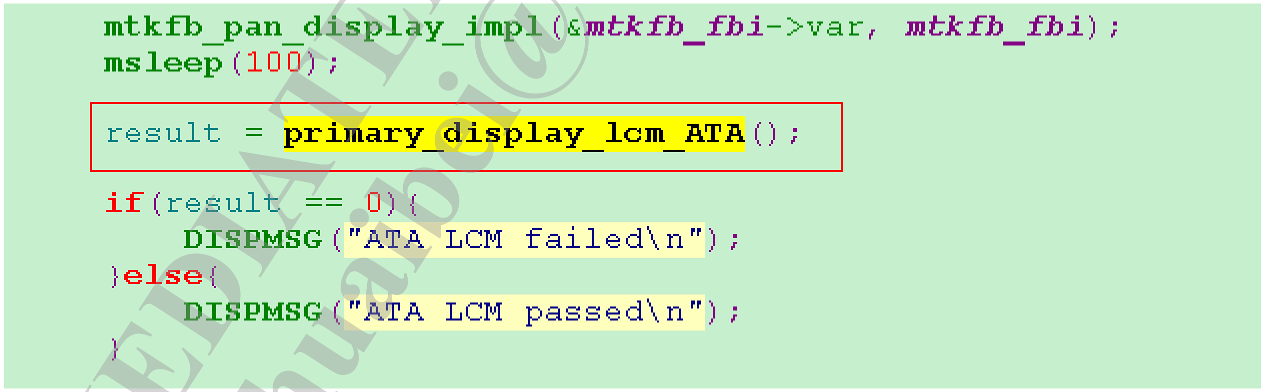
– 2: memset FB to Green picture and update to LCM panel

Note: we must disable BLS, because BLS would change FB data



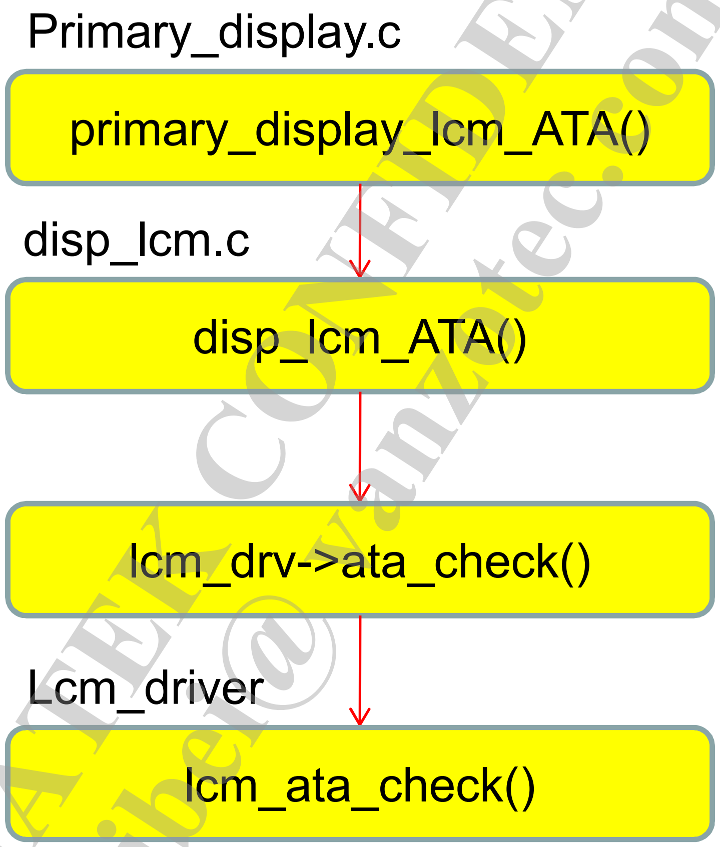
```
    if(color == 0)
        color = 0xFF00FF00;
    fbsize = ALIGN_TO(DISP_GetScreenWidth(),32)*DISP_GetScreenHeight();

    for(i=0;i<fbsize;i++)
        *fb_buffer++ = color;

    msleep(100);
//  for(i=0;i<60;i++)
//      printk("0x%x,",*((unsigned char*)(fbVirAddr) + i));
//  mtkfb_fbi->var.yoffset = 0;
    bls_enable = DSI_BLS_Query();
    printk("BLS is enable %d\n",bls_enable);
    if(bls_enable == 1)
        DSI_BLS_Enable(false);
    mtkfb_pan_display_impl(&mtkfb_fbi->var, mtkfb_fbi);
    msleep(100);
```

– 3: write and read back LCM panel data,then compare them



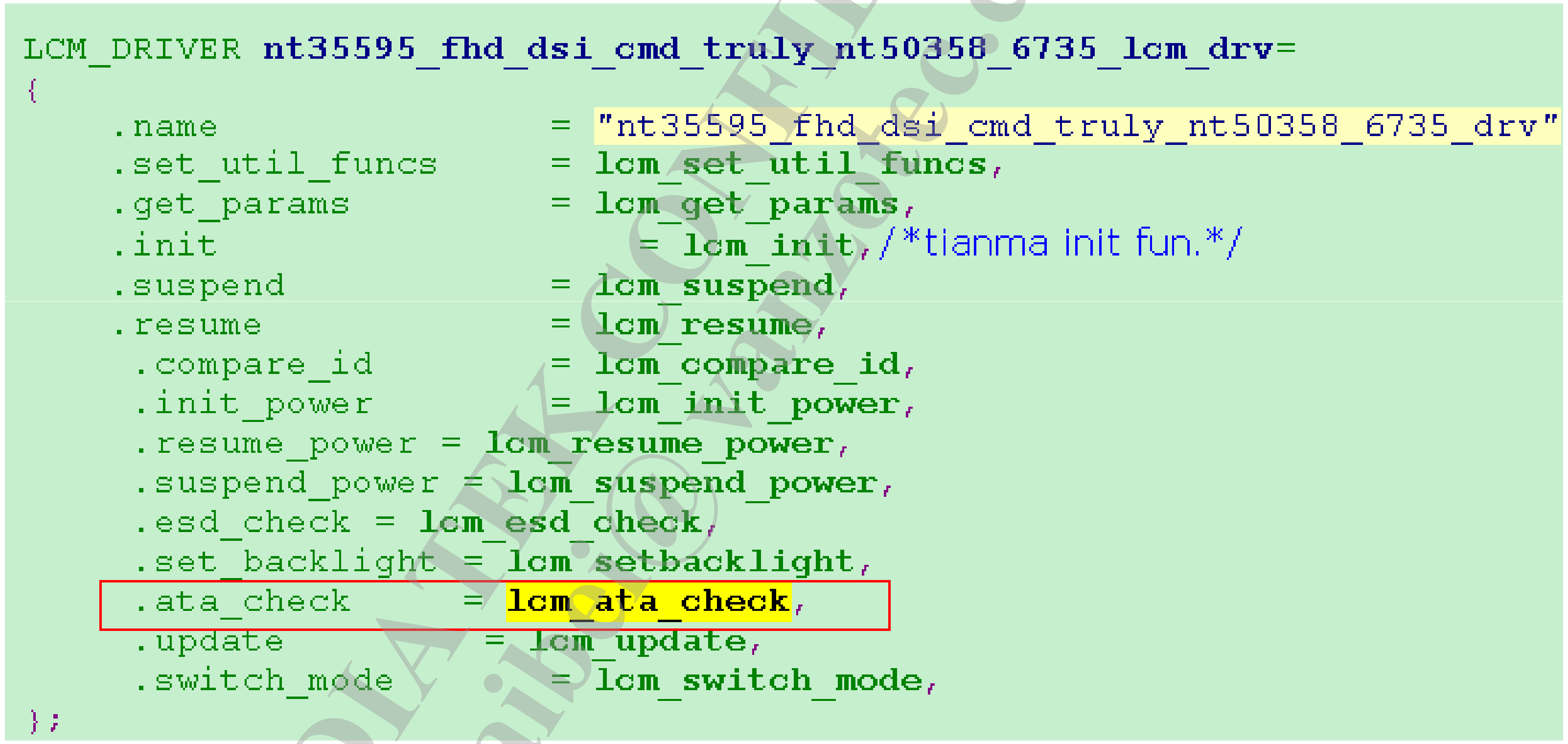
```
    mtkfb_pan_display_impl(&mtkfb_fbi->var, mtkfb_fbi);
    msleep(100);

    result = primary_display_lcm_ATA();

    if(result == 0){
        DISPMSG("ATA LCM failed\n");
    }else{
        DISPMSG("ATA LCM passed\n");
    }
```

Write and Read Back LCM panel data implement



Write and Read Back LCM panel data implement

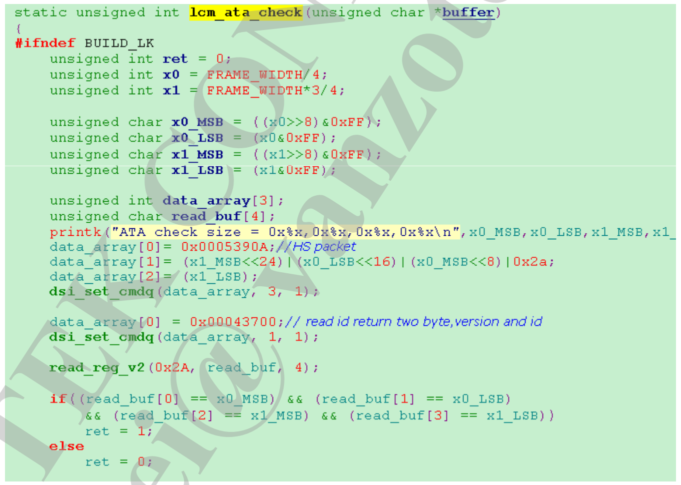
In Lcm driver, implement ata_check function



```
LCM_DRIVER nt35595_fhd_dsi_cmd_truly_nt50358_6735_lcm_drv=
{
    .name                   = "nt35595_fhd_dsi_cmd_truly_nt50358_6735_drv"
    .set_util_funcs         = lcm_set_util_funcs,
    .get_params             = lcm_get_params,
    .init                       = lcm_init,/*tianma init fun.*/
    .suspend                = lcm_suspend,
    .resume                 = lcm_resume,
     .compare_id            = lcm_compare_id,
     .init_power            = lcm_init_power,
     .resume_power = lcm_resume_power,
     .suspend_power = lcm_suspend_power,
     .esd_check = lcm_esd_check,
     .set_backlight = lcm_setbacklight,
     .ata_check        = lcm_ata_check,
     .update             = lcm_update,
     .switch_mode          = lcm_switch_mode,
};
```

DSI command mode

Write 4 bytes to lcm , and read back the 4 bytes , compare them.
If they are equal , the LCM ATA test pass . If not equal , LCM ATA test fail
command 0x2A :Set the GRAM Column Address



```
static unsigned int lcm_ata_check(unsigned char *buffer)
{
#ifndef BUILD_LK
    unsigned int ret = 0;
    unsigned int x0 = FRAME_WIDTH/4;
    unsigned int x1 = FRAME_WIDTH*3/4;

    unsigned char x0_MSB = ((x0>>8)&0xFF);
    unsigned char x0_LSB = (x0&0xFF);
    unsigned char x1_MSB = ((x1>>8)&0xFF);
    unsigned char x1_LSB = (x1&0xFF);

    unsigned int data_array[3];
    unsigned char read_buf[4];
    printk("ATA check size = 0x%x,0x%x,0x%x,0x%x\n",x0_MSB,x0_LSB,x1_MSB,x1_
    data_array[0]= 0x0005390A;//HS packet
    data_array[1]= (x1_MSB<<24)|(x0_LSB<<16)|(x0_MSB<<8)|0x2a;
    data_array[2]= (x1_LSB);
    dsi_set_cmdq(data_array, 3, 1);

    data_array[0] = 0x00043700;// read id return two byte,version and id
    dsi_set_cmdq(data_array, 1, 1);

    read_reg_v2(0x2A, read_buf, 4);

    if((read_buf[0] == x0_MSB) && (read_buf[1] == x0_LSB)
        && (read_buf[2] == x1_MSB) && (read_buf[3] == x1_LSB))
        ret = 1;
    else
        ret = 0;
```

DSI video mode

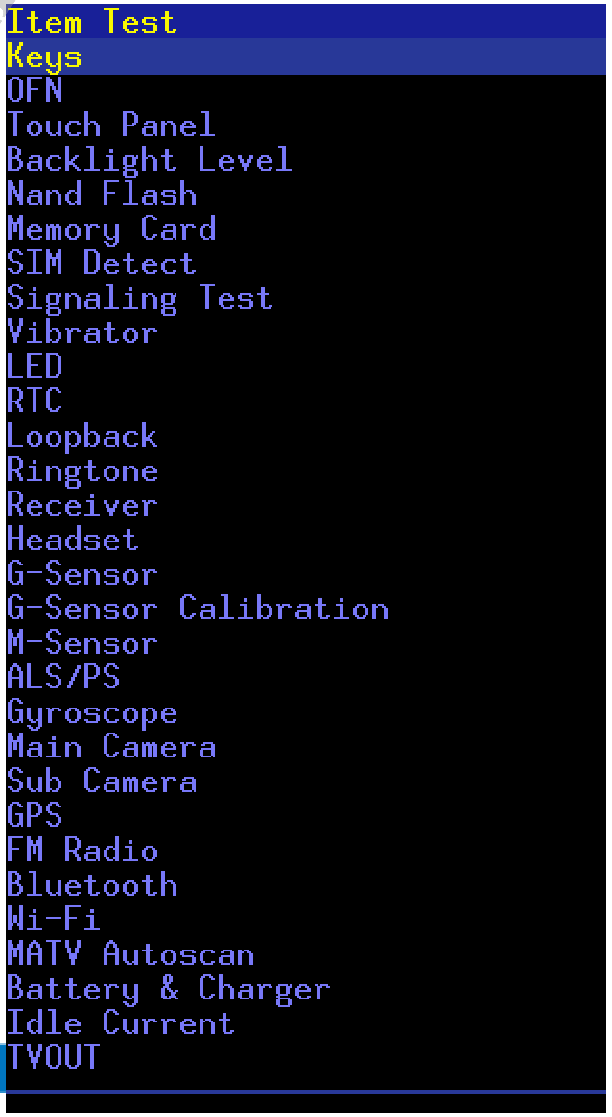
- Because video mode LCM have no GRAM , so it can’t write and read command 0x2A
- Before customizing the function lcm_ata_check() please ask driver IC FAE which register is suitable for ata check. And then implement function lcm_ata_check() like command mode

Confidential B

LCM ATA implement For JB

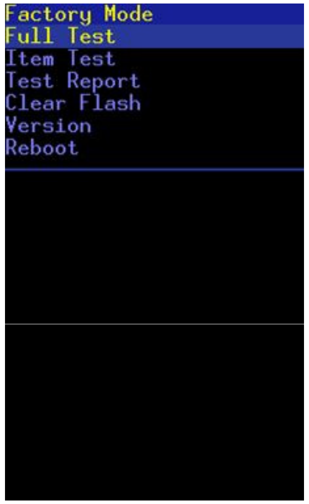
Overview

- In factory mode, add one test item:
- When tap this test item, factory mode call framebuffer driver IOCTL to do:
  - Memset() FB to Green picture and transfer to LCM panel
  - Read back 60 bytes of Gram data on LCM panel and compare these data
- Only **DSI command mode** and **DBI** is supported LCM ATA

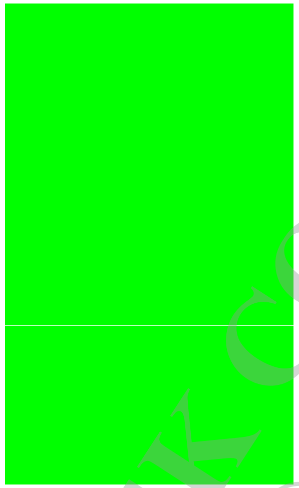


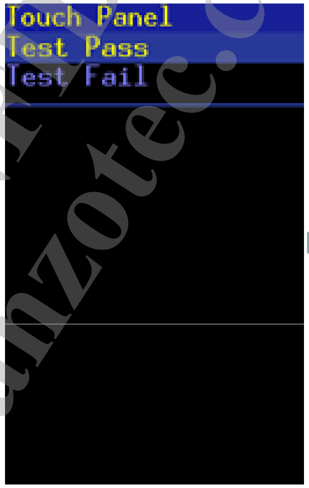
Item Test
Keys
OFN
Touch Panel
Backlight Level
Nand Flash
Memory Card
SIM Detect
Signaling Test
Vibrator
LED
RTC
Loopback
Ringtone
Receiver
Headset
G-Sensor
G-Sensor Calibration
M-Sensor
ALS/PS
Gyroscope
Main Camera
Sub Camera
GPS
FM Radio
Bluetooth
Wi-Fi
MATV Autoscan
Battery & Charger
Idle Current
TVOUT

- We also add test item to Factory mode Auto test

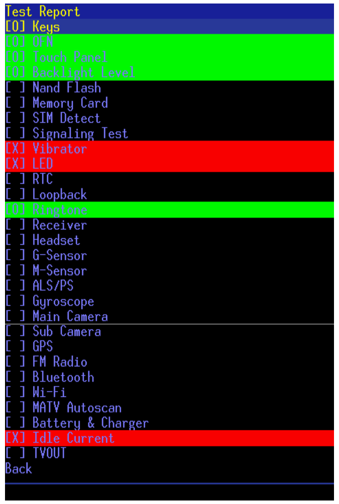


```
Factory Mode
Full Test
Item Test
Test Report
Clear Flash
Version
Reboot
```





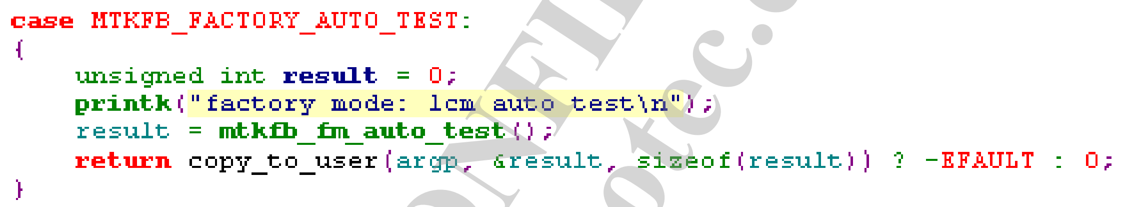
```
Touch Panel
Test Pass
Test Fail
```



```
Test Report
[O] Keys
[O] OFN
[O] Touch Panel
[O] Backlight Level
[ ] Nand Flash
[ ] Memory Card
[ ] SIM Detect
[ ] Signaling Test
[X] Vibrator
[X] LED
[ ] RTC
[ ] Loopback
[O] Ringtone
[ ] Receiver
[ ] Headset
[ ] G-Sensor
[ ] M-Sensor
[ ] ALS/PS
[ ] Gyroscope
[ ] Main Camera
[ ] Sub Camera
[ ] GPS
[ ] FM Radio
[ ] Bluetooth
[ ] Wi-Fi
[ ] MATV Autoscan
[ ] Battery & Charger
[X] Idle Current
[ ] TVOUT
Back
```

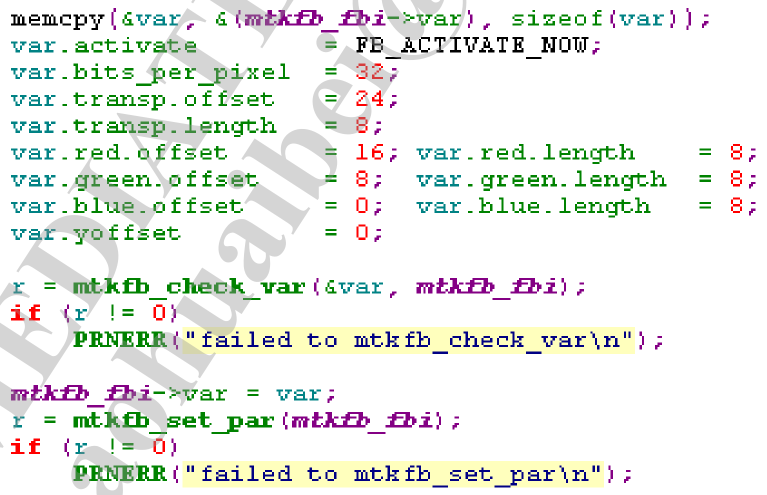
FB driver IOCTL implement

- Mtkfb.c



```
case MTKFB_FACTORY_AUTO_TEST:
{
    unsigned int result = 0;
    printk("factory mode: lcm auto test\n");
    result = mtkfb_fm_auto_test();
    return copy_to_user(argp, &result, sizeof(result)) ? -EFAULT : 0;
}
```

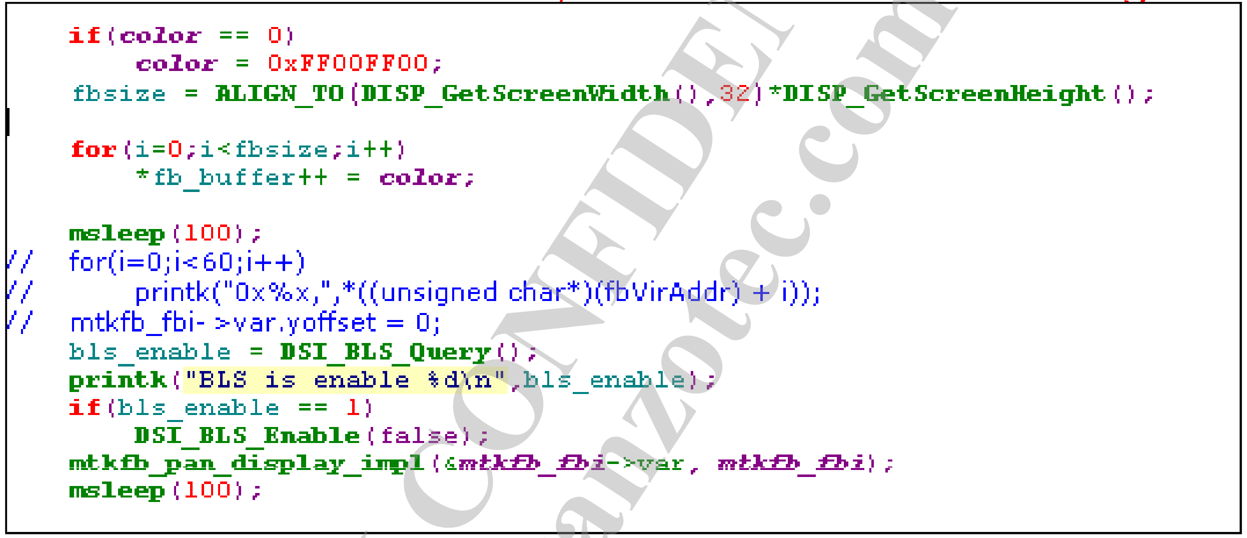
- In mtkfb_fm_auto_test(), we could divide these code into 3 parts:
  – 1: Change FB format from RGB565 to ARGB8888



```
memcpy(&var, &(mtkfb_fbi->var), sizeof(var));
var.activate        = FB_ACTIVATE_NOW;
var.bits_per_pixel  = 32;
var.transp.offset   = 24;
var.transp.length   = 8;
var.red.offset      = 16; var.red.length    = 8;
var.green.offset    = 8;  var.green.length  = 8;
var.blue.offset     = 0;  var.blue.length   = 8;
var.yoffset         = 0;

r = mtkfb_check_var(&var, mtkfb_fbi);
if (r != 0)
    PRNERR("failed to mtkfb_check_var\n");

mtkfb_fbi->var = var;
r = mtkfb_set_par(mtkfb_fbi);
if (r != 0)
    PRNERR("failed to mtkfb_set_par\n");
```

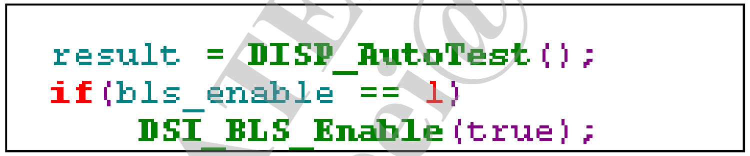
- 2: memset FB to Green picture and update to LCM panel

  Note: we must disable BLS, because BLS would change FB data



```
    if(color == 0)
        color = 0xFF00FF00;
    fbsize = ALIGN_TO(DISP_GetScreenWidth(),32)*DISP_GetScreenHeight();

    for(i=0;i<fbsize;i++)
        *fb_buffer++ = color;

    msleep(100);
//  for(i=0;i<60;i++)
//      printk("0x%x,",*((unsigned char*)(fbVirAddr) + i));
//  mtkfb_fbi->var.yoffset = 0;
    bls_enable = DSI_BLS_Query();
    printk("BLS is enable %d\n",bls_enable);
    if(bls_enable == 1)
        DSI_BLS_Enable(false);
    mtkfb_pan_display_impl(&mtkfb_fbi->var, mtkfb_fbi);
    msleep(100);
```

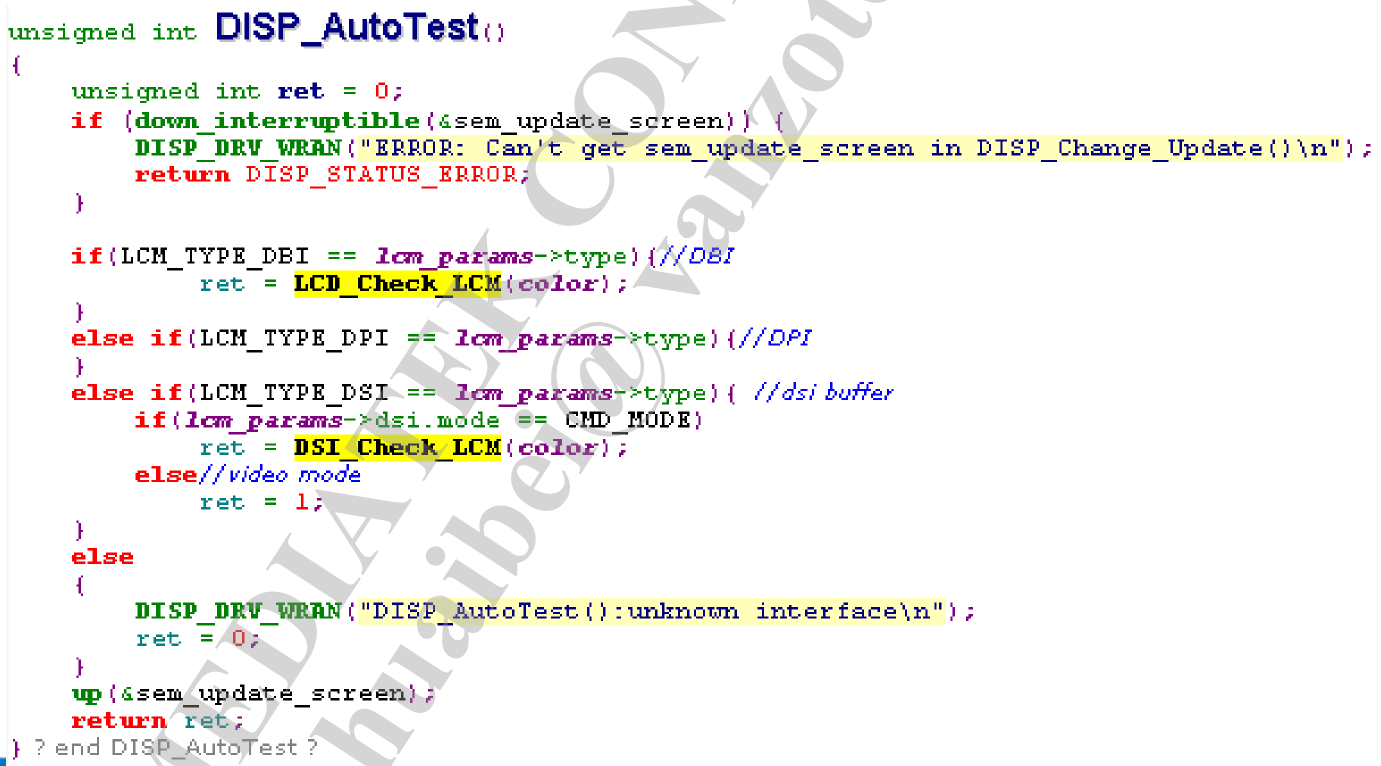
- 3: read back LCM panel data, and enable BLS if need



```
result = DISP_AutoTest();
if(bls_enable == 1)
    DSI_BLS_Enable(true);
```

Read Back LCM panel data implement

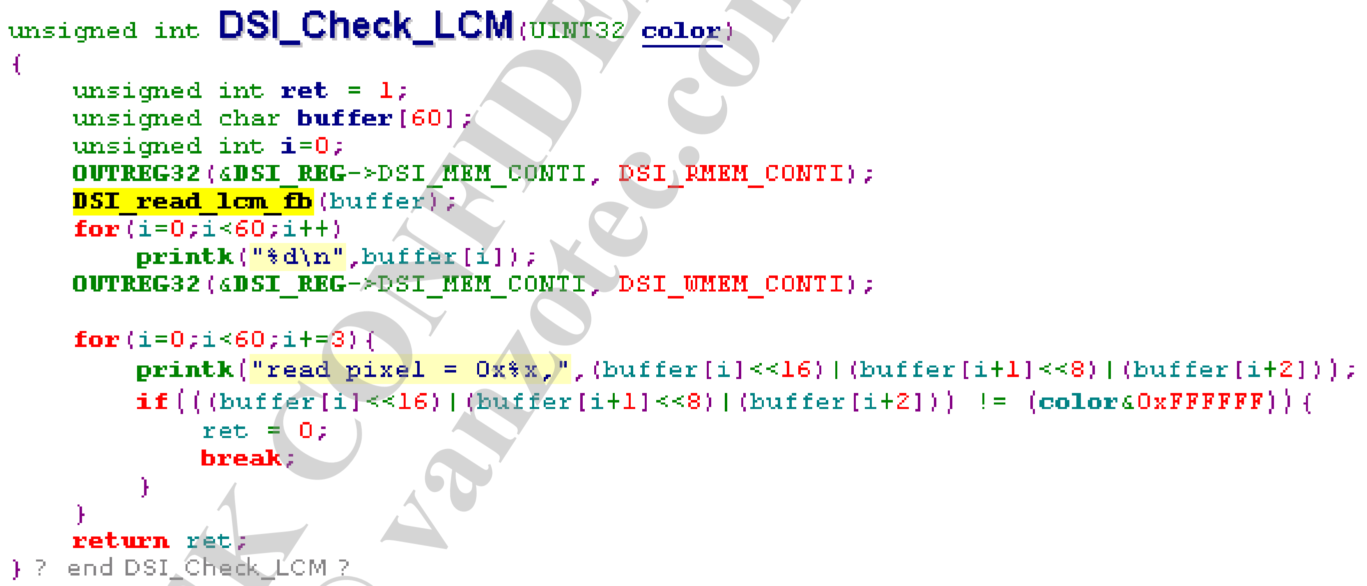
- Disp_drv.c
  - In DPI/DSI video mode case, there is no Gram on LCM panel, so can not implement this request
  - Only DSI command mode & DBI is supported.



```
unsigned int DISP_AutoTest()
{
    unsigned int ret = 0;
    if (down_interruptible(&sem_update_screen)) {
        DISP_DRV_WRAN("ERROR: Can't get sem_update_screen in DISP_Change_Update()\n");
        return DISP_STATUS_ERROR;
    }

    if(LCM_TYPE_DBI == lcm_params->type){//DBI
            ret = LCD_Check_LCM(color);
    }
    else if(LCM_TYPE_DPI == lcm_params->type){//DPI
    }
    else if(LCM_TYPE_DSI == lcm_params->type){ //dsi buffer
        if(lcm_params->dsi.mode == CMD_MODE)
            ret = DSI_Check_LCM(color);
        else//video mode
            ret = 1;
    }
    else
    {
        DISP_DRV_WRAN("DISP_AutoTest():unknown interface\n");
        ret = 0;
    }
    up(&sem_update_screen);
    return ret;
} ? end DISP_AutoTest ?
```

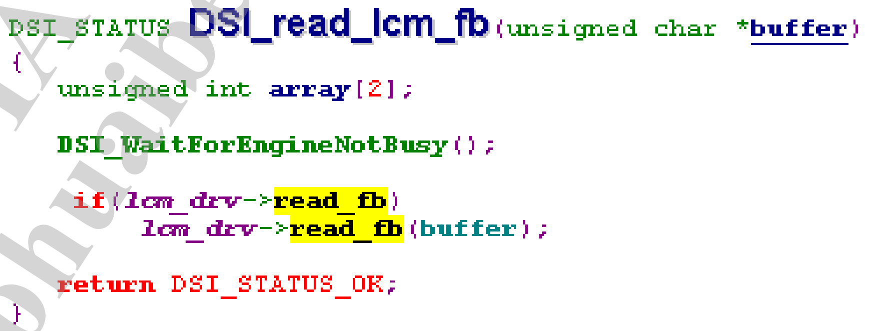
DSI command mode

- Dsi_drv.c



```
unsigned int DSI_Check_LCM(UINT32 color)
{
    unsigned int ret = 1;
    unsigned char buffer[60];
    unsigned int i=0;
    OUTREG32(&DSI_REG->DSI_MEM_CONTI, DSI_RMEM_CONTI);
    DSI_read_lcm_fb(buffer);
    for(i=0;i<60;i++)
        printk("%d\n",buffer[i]);
    OUTREG32(&DSI_REG->DSI_MEM_CONTI, DSI_WMEM_CONTI);

    for(i=0;i<60;i+=3){
        printk("read pixel = 0x%x,",(buffer[i]<<16)|(buffer[i+1]<<8)|(buffer[i+2]));
        if(((buffer[i]<<16)|(buffer[i+1]<<8)|(buffer[i+2])) != (color&0xFFFFFF)){
            ret = 0;
            break;
        }
    }
    return ret;
} ? end DSI_Check_LCM ?
```

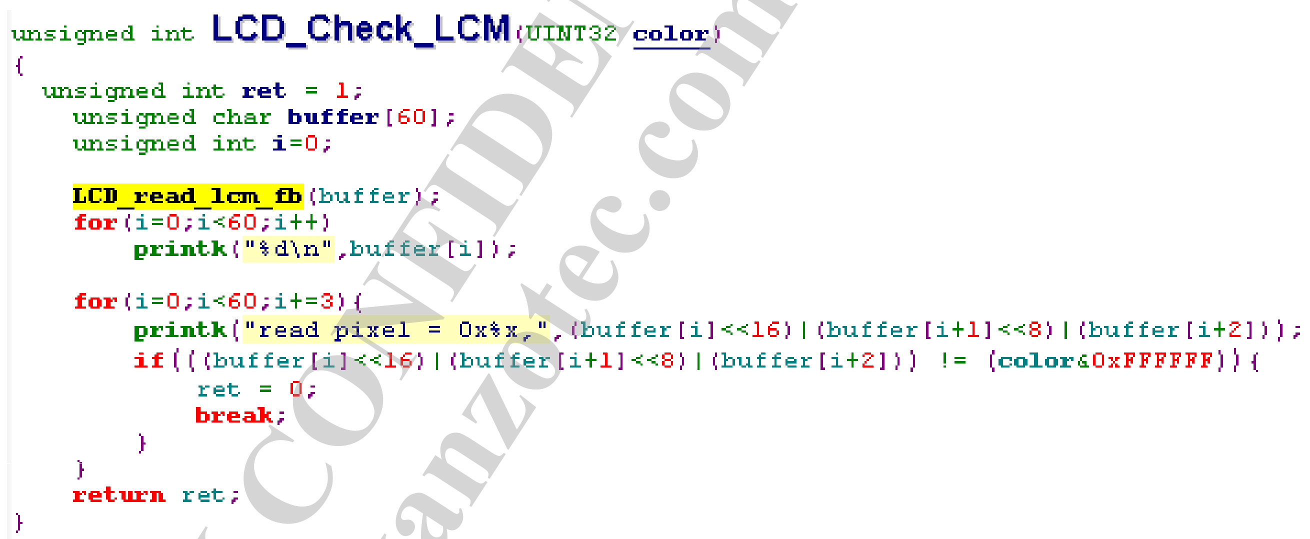
  - Dsi driver pass one buffer pointer to LCM driver, so customer must implement read_fb() function in LCM driver to support this feature



```
DSI_STATUS DSI_read_lcm_fb(unsigned char *buffer)
{
    unsigned int array[2];

    DSI_WaitForEngineNotBusy();

    if(lcm_drv->read_fb)
        lcm_drv->read_fb(buffer);

    return DSI_STATUS_OK;
}
```

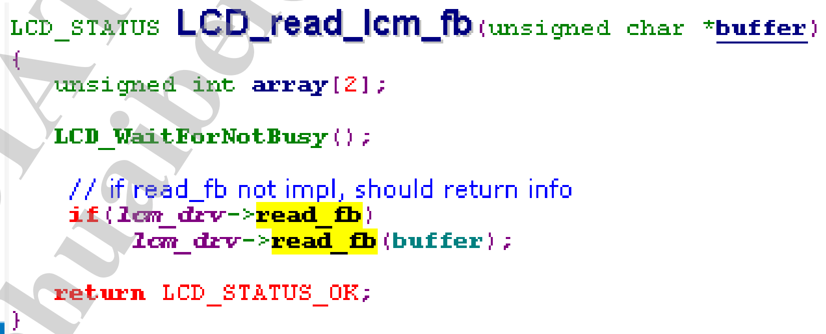
DBI

- Lcd_drv.c



```
unsigned int LCD_Check_LCM(UINT32 color)
{
  unsigned int ret = 1;
    unsigned char buffer[60];
    unsigned int i=0;

    LCD_read_lcm_fb(buffer);
    for(i=0;i<60;i++)
        printk("%d\n",buffer[i]);

    for(i=0;i<60;i+=3){
        printk("read pixel = 0x%x,",(buffer[i]<<16)|(buffer[i+1]<<8)|(buffer[i+2]));
        if(((buffer[i]<<16)|(buffer[i+1]<<8)|(buffer[i+2])) != (color&0xFFFFFF)){
            ret = 0;
            break;
        }
    }
    return ret;
}
```

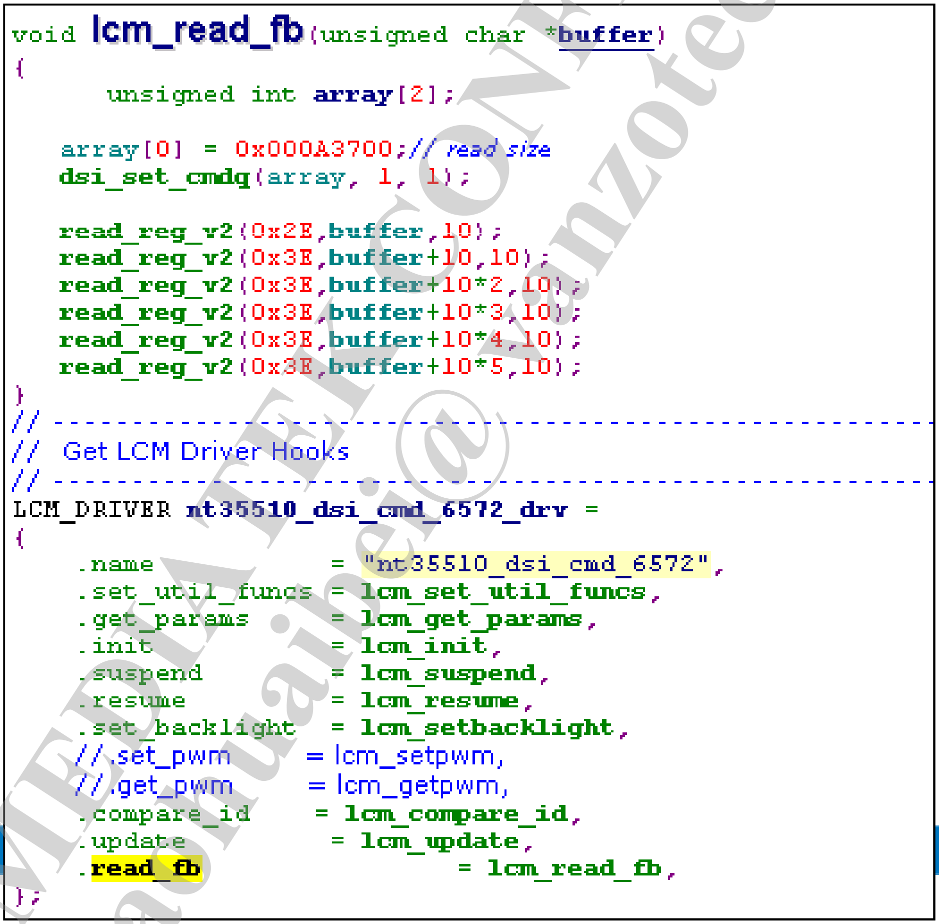
  - DBI driver pass one buffer pointer to LCM driver, so customer must implement read_fb() function in LCM driver to support this feature



```
LCD_STATUS LCD_read_lcm_fb(unsigned char *buffer)
{
   unsigned int array[2];

   LCD_WaitForNotBusy();

    // if read_fb not impl, should return info
    if(lcm_drv->read_fb)
        lcm_drv->read_fb(buffer);

   return LCD_STATUS_OK;
}
```

Customization

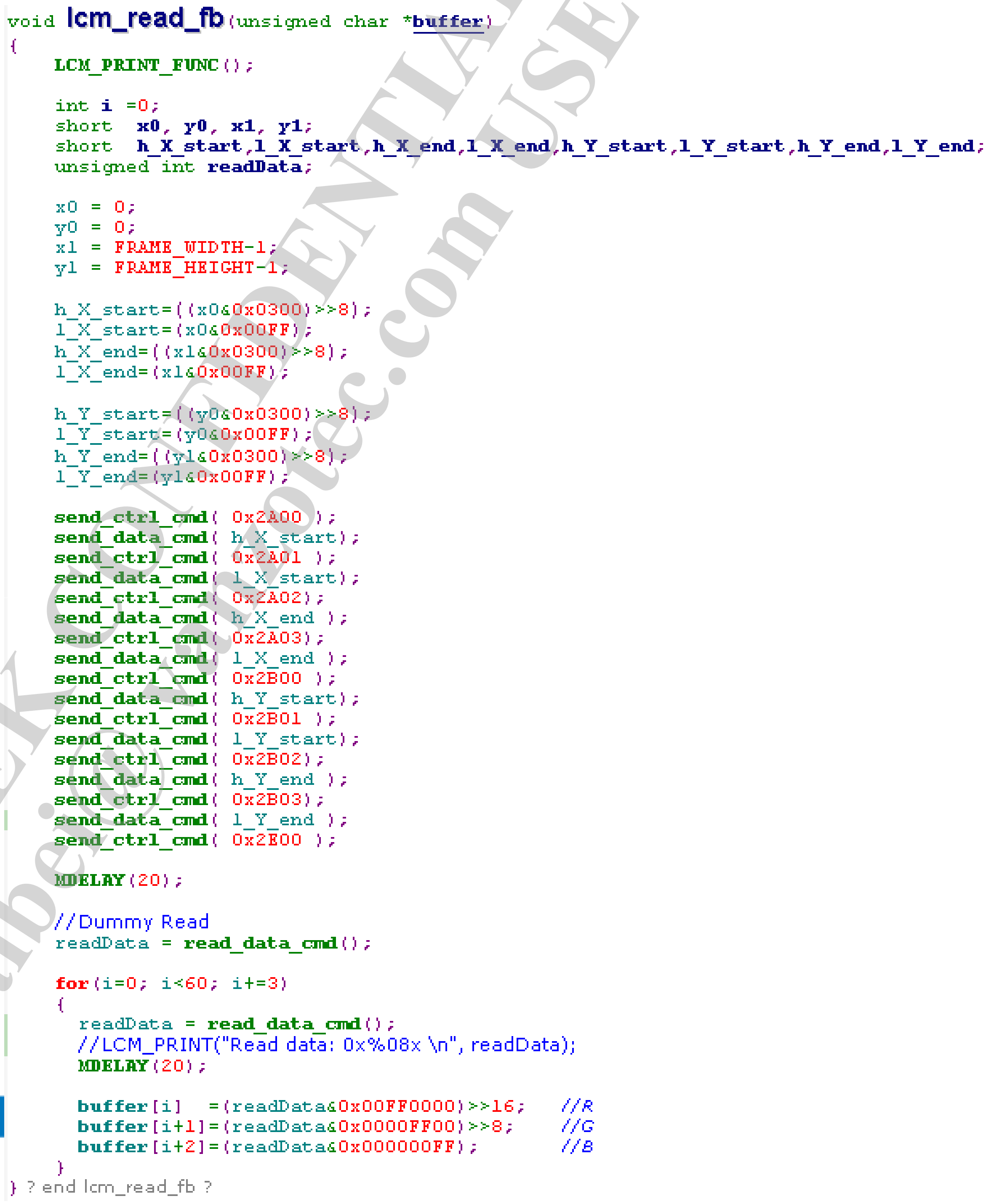
- DSI command mode
  - LCM driver: nt35510_dsi_cmd_6572



```
void lcm_read_fb(unsigned char *buffer)
{
      unsigned int array[2];

   array[0] = 0x000A3700;// read size
   dsi_set_cmdq(array, 1, 1);

   read_reg_v2(0x2E,buffer,10);
   read_reg_v2(0x3E,buffer+10,10);
   read_reg_v2(0x3E,buffer+10*2,10);
   read_reg_v2(0x3E,buffer+10*3,10);
   read_reg_v2(0x3E,buffer+10*4,10);
   read_reg_v2(0x3E,buffer+10*5,10);
}
// ---------------------------------------------------------------
//  Get LCM Driver Hooks
// ---------------------------------------------------------------
LCM_DRIVER nt35510_dsi_cmd_6572_drv =
{
    .name               = "nt35510_dsi_cmd_6572",
    .set_util_funcs     = lcm_set_util_funcs,
    .get_params         = lcm_get_params,
    .init               = lcm_init,
    .suspend            = lcm_suspend,
    .resume             = lcm_resume,
    .set_backlight      = lcm_setbacklight,
    //.set_pwm        = lcm_setpwm,
    //.get_pwm        = lcm_getpwm,
    .compare_id        = lcm_compare_id,
    .update             = lcm_update,
    .read_fb                     = lcm_read_fb,
};
```

Customization

- DBI
  - LCM driver: nt35510_dbi_18bit



```
void lcm_read_fb(unsigned char *buffer)
{
    LCM_PRINT_FUNC();

    int i =0;
    short  x0, y0, x1, y1;
    short  h_X_start,l_X_start,h_X_end,l_X_end,h_Y_start,l_Y_start,h_Y_end,l_Y_end;
    unsigned int readData;

    x0 = 0;
    y0 = 0;
    x1 = FRAME_WIDTH-1;
    y1 = FRAME_HEIGHT-1;

    h_X_start=((x0&0x0300)>>8);
    l_X_start=(x0&0x00FF);
    h_X_end=((x1&0x0300)>>8);
    l_X_end=(x1&0x00FF);

    h_Y_start=((y0&0x0300)>>8);
    l_Y_start=(y0&0x00FF);
    h_Y_end=((y1&0x0300)>>8);
    l_Y_end=(y1&0x00FF);

    send_ctrl_cmd( 0x2A00 );
    send_data_cmd( h_X_start);
    send_ctrl_cmd( 0x2A01 );
    send_data_cmd( l_X_start);
    send_ctrl_cmd( 0x2A02);
    send_data_cmd( h_X_end );
    send_ctrl_cmd( 0x2A03);
    send_data_cmd( l_X_end );
    send_ctrl_cmd( 0x2B00 );
    send_data_cmd( h_Y_start);
    send_ctrl_cmd( 0x2B01 );
    send_data_cmd( l_Y_start);
    send_ctrl_cmd( 0x2B02);
    send_data_cmd( h_Y_end );
    send_ctrl_cmd( 0x2B03);
    send_data_cmd( l_Y_end );
    send_ctrl_cmd( 0x2E00 );

    MDELAY(20);

    //Dummy Read
    readData = read_data_cmd();

    for(i=0; i<60; i+=3)
    {
      readData = read_data_cmd();
      //LCM_PRINT("Read data: 0x%08x \n", readData);
      MDELAY(20);

      buffer[i]  =(readData&0x00FF0000)>>16;   //R
      buffer[i+1]=(readData&0x0000FF00)>>8;    //G
      buffer[i+2]=(readData&0x000000FF);       //B
    }
} ? end lcm_read_fb ?
```