



Linux DSPO

开发指南

版本号: 0.1
发布日期: 2021.4.08

版本历史

| 版本号 | 日期 | 制/修订人 | 内容描述 |
|-----|-----------|---------|-----------------|
| 0.1 | 2021.4.08 | AWA1221 | Initial Version |



目 录

| | |
|--------------------------------------|-----------|
| 1 概述 | 1 |
| 1.1 编写目的 | 1 |
| 1.2 适用范围 | 1 |
| 1.3 相关人员 | 1 |
| 2 相关术语介绍 | 2 |
| 3 模块介绍 | 3 |
| 3.1 IP 规格 | 3 |
| 4 配置 | 4 |
| 4.1 menuconfig | 4 |
| 4.2 dts 配置实例 | 4 |
| 4.3 dts 属性详解 | 6 |
| 4.3.1 enable | 6 |
| 4.3.2 protocol | 6 |
| 4.3.3 bit_width | 7 |
| 4.3.4 separate_sync | 7 |
| 4.3.5 data_seq_sel | 7 |
| 4.3.6 output_mode | 7 |
| 4.3.7 dclk_dly_en | 8 |
| 4.3.8 dclk_dly_num | 8 |
| 4.3.9 dclk_invert | 8 |
| 4.3.10 pixel_clk | 9 |
| 4.3.11 x_res | 9 |
| 4.3.12 y_res | 9 |
| 4.3.13 ht | 9 |
| 4.3.14 hbp | 10 |
| 4.3.15 hfp | 10 |
| 4.3.16 hspw | 10 |
| 4.3.17 vt | 10 |
| 4.3.18 vbp | 11 |
| 4.3.19 vfp | 11 |
| 4.3.20 vspw | 11 |
| 4.3.21 hpol | 11 |
| 4.3.22 vpol | 12 |
| 5 用户接口 | 13 |
| 5.1 DSPO_CMD_DEVICE_SWITCH | 13 |
| 5.2 DSPO_CMD_COMMIT | 15 |
| 6 调试 | 17 |
| 6.1 打印信息 | 17 |

| | |
|-------------------------|-----------|
| 6.2 检查硬件 | 17 |
| 6.3 调整 timing | 18 |
| 6.4 调整管脚驱动能力 | 18 |
| 7 新平台移植 | 19 |



1 概述

1.1 编写目的

介绍 sunxi 平台 DSPO 模块移植，配置，使用和调试方法。

1.2 适用范围

表 1-1: 适用产品列表

| 产品名称 | 内核版本 | 驱动文件 |
|------|-----------|-------------------------|
| V833 | Linux-4.9 | drivers/char/sunxi_dspo |

1.3 相关人员

驱动开发人员，用户。

2 相关术语介绍

表 2-1: LCD 相关术语

| 术语 | 解释说明 |
|--------|--------------------------------|
| SUNXI | Allwinner 一系列 SOC 硬件平台 |
| DSPO | Display Stream Parallel Output |
| BT1120 | ITU-R BT.1120 |
| BT656 | ITU-R BT.656 |



3 模块介绍

DSPO 模块是一个兼容 ITU.R BT.656 和 BT.1120 协议的并行数字视频输出接口。它从 DDR 总线中取数据然后按照协议编码输出，常用于传输视频数据到另外一个芯片。

3.1 IP 规格

特性：

1. 支持 BT1120 (16 bits) , 最大支持 2688x2688@15 Hz / 3840 x 2160@15 Hz
2. 支持 BT656 (8 bits) , 最大支持 2688x2688@8 Hz / 3840 x 2160@8 Hz
3. 支持内嵌 H.V.F 同步信号，也支持外同步。
4. 支持 2 个专用 DMA 通道，数据格式支持 ARGB888/YUV444/YUV422/YUV420 等。
5. 3V 或者 1.8V IO 电

4 配置

4.1 menuconfig

如下图所示，路径是 **Device Drivers -> Character devices -> sunxi DSPO driver**

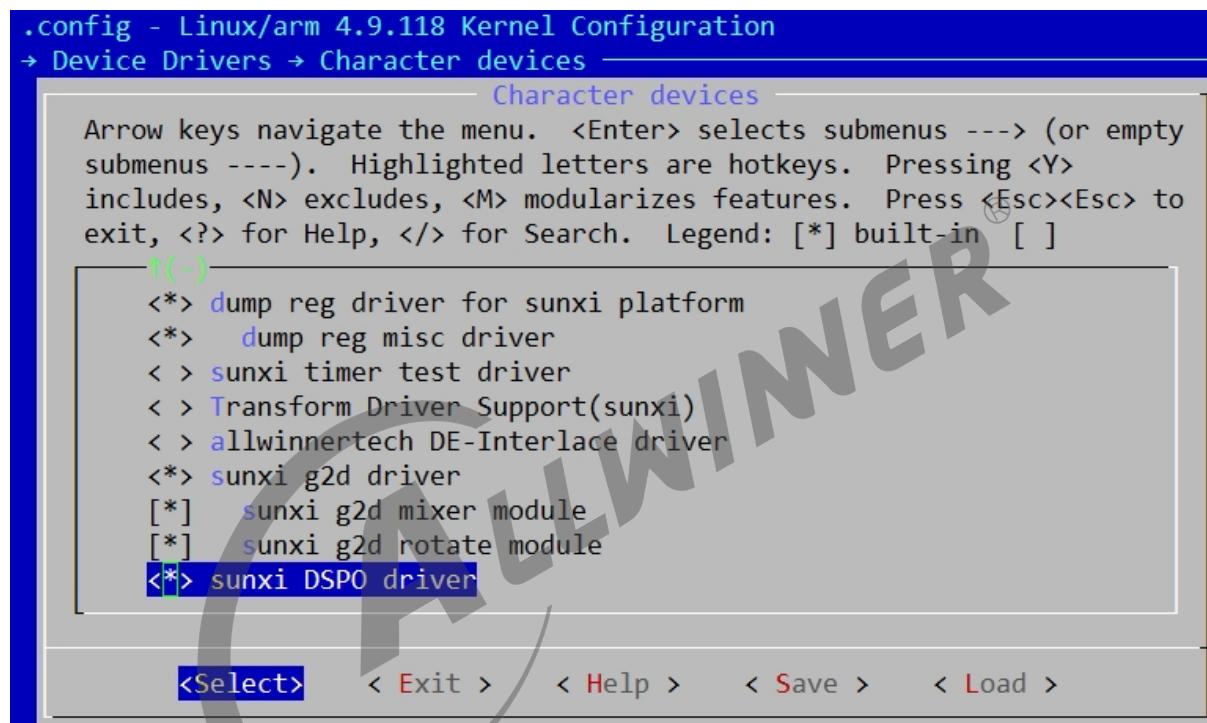


图 4-1: menuconfig

4.2 dts 配置实例

用户如果需要在加载驱动的时候就使能 DSPO 模块，那么仅需要在板级 **board.dts** 配置。用户也可以选择通过用户接口来使能 DSPO，这个时候就不需要配置 board.dts。

请将 dsopo 节点配置在 **soc** 节点之下，如下所示：

```
soc@03000000 {  
    ....  
  
    dsopo: dsopo@0 {  
        ....  
    };
```

```
};
```

下面列举三个例子：

BT1120 内嵌同步 1080p@60

```
/*
enable: 1, enable; 0, disable
protocol: 0, BT1120; 1, BT656
bit_width: 0, 8bit; 1, 16bit
separate_sync: 0, embedded sync; 1, separate_sync
data_seq_sel: 0, CB_Y_CR_Y; 1, CR_Y_CB_Y; 2, Y_CB_Y_CR; 3, Y_CR_Y_CB
dclk_dly_en: 0, disable; 1, enable clk delay
dclk_dly_num: greater then zero, the number of clk delay period
dclk_invert: invert clk
output_mode: output resolution. see enum dspo_output_mode in include/linux/dspo_drv.h
*/
dsopo: dsopo@0 {
    enable = <1>;
    protocol = <0>;
    bit_width = <1>;
    separate_sync = <0>;
    data_seq_sel = <0>;
    output_mode = <12>;
    dclk_dly_en = <0>;
    dclk_dly_num = <0>;
    dclk_invert = <0>;
};
```

BT656 内嵌同步 720p@60

```
/*
enable: 1, enable; 0, disable
protocol: 0, BT1120; 1, BT656
bit_width: 0, 8bit; 1, 16bit
separate_sync: 0, embedded sync; 1, separate_sync
data_seq_sel: 0, CB_Y_CR_Y; 1, CR_Y_CB_Y; 2, Y_CB_Y_CR; 3, Y_CR_Y_CB
dclk_dly_en: 0, disable; 1, enable clk delay
dclk_dly_num: greater then zero, the number of clk delay period
dclk_invert: invert clk
output_mode: output resolution. see enum dspo_output_mode in include/linux/dspo_drv.h
*/
dsopo: dsopo@0 {
    enable = <1>;
    protocol = <1>;
    bit_width = <0>;
    separate_sync = <0>;
    data_seq_sel = <0>;
    output_mode = <8>;
    dclk_dly_en = <0>;
    dclk_dly_num = <0>;
    dclk_invert = <0>;
};
```

自定义 timing:1080p@30

自定义 timing 的关键是将 output_mode 设置成 14，然后自定义时序 timing。

```
dspo: dspo@0 {  
    enable = <1>;  
    protocol = <0>;  
    bit_width = <1>;  
    separate_sync = <0>;  
    data_seq_sel = <0>;  
    output_mode = <14>;  
    dclk_dly_en = <1>;  
    dclk_dly_num = <0>;  
    dclk_invert = <0>;  
  
    pixel_clk = <80000000>;  
    x_res = <1920>;  
    y_res = <1080>;  
  
    ht = <2416>;  
    hbp = <248>;  
    hfp = <56>;  
    hspw = <192>;  
    hpol = <0>;  
  
    vt = <1102>;  
    vbp = <14>;  
    vfp = <3>;  
    vspw = <5>;  
    vpol = <1>;  
    interlace = <0>;  
};
```

4.3 dts 属性详解

4.3.1 enable

| 取值 | 描述 |
|----|-------------|
| 0 | 不使能 DSPO 设备 |
| 1 | 使能 |

4.3.2 protocol

协议选择

| 取值 | 描述 |
|----|---------------------|
| 0 | 使用 ITU.R BT.1120 协议 |
| 1 | 使用 ITU.R BT.656 协议 |

4.3.3 bit_width

数据线的位宽（数量），这里的取值要与硬件原理图连接相对应。

| 取值 | 描述 |
|----|--------|
| 0 | 8 位位宽 |
| 1 | 16 位位宽 |

4.3.4 separate_sync

同步方式选择。这里的取值要与硬件原理图的连接相对应。

| 取值 | 描述 |
|----|------------|
| 0 | 内嵌同步 H.V.F |
| 1 | 外同步 |

4.3.5 data_seq_sel

输出数据分量顺序调整。下面取值与内核头文件 `include/linux/dspo_drv.h` 的中的数据类型 `enum dsopo_data_seq_t` 一致。

| 取值 | 描述 |
|----|-----------|
| 0 | CB-Y-CR-Y |
| 1 | CR-Y-CB-Y |
| 2 | Y-CB-Y-CR |
| 3 | Y-CR-Y-CB |

4.3.6 output_mode

输出分辨率选择。下面取值与内核头文件 `include/linux/dspo_drv.h` 的中的数据类型 `enum dsopo_output_mode` 一致。

| 取值 | 描述 |
|----|-------------|
| 0 | NTSC |
| 1 | PAL |
| 2 | 720X480I_60 |
| 3 | 720X576I_60 |

| 取值 | 描述 |
|----|---------------|
| 4 | 720X480P_60 |
| 5 | 720X576P_50 |
| 6 | 1280X720P_25 |
| 7 | 1280X720P_50 |
| 8 | 1280X720P_60 |
| 9 | 1920X1080I_50 |
| 10 | 1920X1080I_60 |
| 11 | 1920X1080P_50 |
| 12 | 1920X1080P_60 |
| 13 | 3840X1080P_30 |
| 14 | 自定义模式 |

当 output_mode 取值为 14，也就是自定义模式的时候，用户可以定制分辨率的 timing

4.3.7 dclk_dly_en

时钟脚延迟。此功能常用于调整发送端和接收端之间时钟的差异导致的传输异常。

| 取值 | 描述 |
|----|-----------|
| 0 | 禁止 clk 延迟 |
| 1 | 使能 clk 延迟 |

4.3.8 dclk_dly_num

时钟延迟的周期数量。取值范围是 0 到 63。

4.3.9 dclk_invert

| 取值 | 描述 |
|----|-------------|
| 0 | 禁止 clk 极性颠倒 |
| 1 | 使能 clk 极性颠倒 |

4.3.10 pixel_clk

像素时钟，单位 Hz。像素时钟的意思是指 DSPO 发送一个像素所需要时间周期对应的频率。

此属性仅当 `output_mode` 为 14 时有效。

💡 说明

像素时钟的频率并不是时钟脚的频率值，两者不一定相等。

比如说 8 位的 **bt656** 协议，由于一个时钟周期只能发送一个字节 (8 位)，而 **yuv422** 格式至少需要两个时钟周期才能发送一个像素，所以像素时钟和时钟脚的频率关系是 **1:2** 而 16 位的 **bt1120** 协议，一个时钟能发两个字节，像素时钟和时钟脚频率是 **1:1**

💡 技巧

$$\text{pixel_clk} = \text{ht} * \text{vt} * \text{fps}$$

4.3.11 x_res

分辨率的宽。

此属性仅当 `output_mode` 为 14 时有效。

4.3.12 y_res

分辨率的高。

此属性仅当 `output_mode` 为 14 时有效。

4.3.13 ht

水平总像素数。

此属性仅当 `output_mode` 为 14 时有效。

💡 技巧

$$\text{ht} = \text{x_res} + \text{hbp} + \text{hfp} + \text{hspw}$$

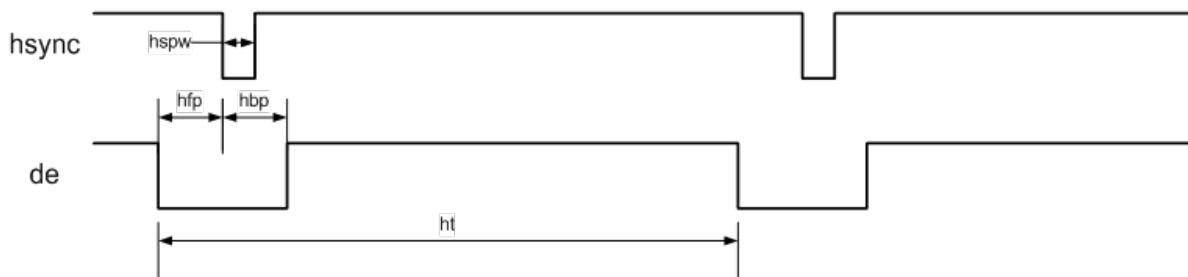


图 4-2: ht

4.3.14 hbp

Horizontal Back Porch

指有效行间，行同步信号（hsync）开始，到有效数据开始之间的 dclk 的 cycle 个数，包括同步信号区。见上图，注意的是包含了 hspw 段。

此属性仅当 `output_mode` 为 14 时有效。

4.3.15 hfp

Horizontal Front Porch

显示后沿。

此属性仅当 `output_mode` 为 14 时有效。

4.3.16 hspw

Horizontal Sync Pulse Width

指行同步信号的宽度。单位为 1 个 dclk 的时间（即是 1 个 data cycle 的时间）。

此属性仅当 `output_mode` 为 14 时有效。

4.3.17 vt

Vertical Total time

指一场的总行数。见下图：

此属性仅当 `output_mode` 为 14 时有效。

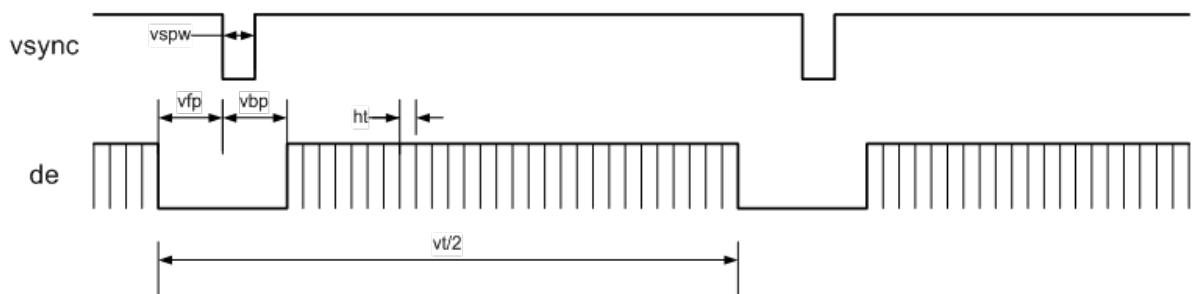


图 4-3: vt

技巧

$$vt = y_res + vbp + vfp + vspw$$

4.3.18 vbp

Vertical Back Porch

指场同步信号（vsync）开始，到有效数据行开始之间的行数，包括场同步信号区。

此属性仅当output_mode为 14 时有效。

4.3.19 vfp

Vertical front Porch

垂直显示后沿。

此属性仅当output_mode为 14 时有效。

4.3.20 vspw

Vertical Sync Pulse Width

指场同步信号的宽度。单位为行。

此属性仅当output_mode为 14 时有效。

4.3.21 hpol

水平同步信号极性。1 表示高电平，0 表示低电平。

此属性仅当output_mode为14时有效。

4.3.22 vpol

垂直同步信号极性。1表示高电平，0表示低电平。

此属性仅当output_mode为14时有效。



5 用户接口

dsopo 驱动加载之后，会生成/**dev/dspo** 设备节点，用户打开(open)此设备节点并通过指定 ioctl 命令，便可操作 DSPO 设备。

用户头文件位于内核目录中的 **include/linux/dspo_drv.h**

5.1 DSPO_CMD_DEVICE_SWITCH

ioctl 命令，用于使能或者禁止 DSPO 设备。

核心结构体，结构体成员含义与[dts 属性详解](#)中的解释一致。

```
struct dsopo_config_t {
    enum dsopo_protocol_t protocol;
    enum dsopo_datawidth_t bit_width;
    bool separate_sync;
    enum dsopo_data_seq_t data_seq_sel;
    struct dsopo_dclk_adjust_t dclk_set;
    enum dsopo_output_mode output_mode;
    struct dsopo_video_timings timing_info;
};
```

bt656 内嵌同步，720p@25 示例：

```
#include "dspo_drv.h"

int decd_fd = 0;
struct dsopo_config_t cfg;

decd_fd = open("/dev/dspo", O_RDWR);
if (!decd_fd) {
    perror("\n");
    goto OUT;
}

memset(&cfg, 0, sizeof(struct dsopo_config_t));
cfg.protocol = DSPO_BT656;
cfg.bit_width = DSPO_8_BIT;
cfg.separate_sync = false;
cfg.data_seq_sel = DSPO_CB_Y_CR_Y;
cfg.dclk_set.dclk_en = true;
cfg.dclk_set.dclk_invt = false;
cfg.dclk_set.dclk_dly_num = 0;
cfg.output_mode = DSPO_MOD_1280X720P_25;

arg[0] = myinfo.en;
```

```
arg[1] = (unsigned long)&cfg;

ret = ioctl(decd_fd, DSPO_CMD_DEVICE_SWITCH, arg);
if (ret) {
    printf("DSPO_CMD_DEVICE_SWITCH fail!: %d\n", ret);
    goto OUT;
}
```

关于分辨率及其 timing，一方面内置了若干个分辨率，但是考虑到接收端对分辨率 timing 要求的差异，为了给用户最大灵活度，当核心结构体的 **output_mode** 等于 **DSPO_CUSTOM_MODE**，用户填充另外一个成员 **timing_info** 即可实现自定义的模式。

自定义分辨率 (1080p@30) 示例：

```
#include "dsopo_drv.h"

int decd_fd = 0;
struct dsopo_config_t cfg;

decd_fd = open("/dev/dspo", O_RDWR);
if (!decd_fd) {
    perror("\n");
    goto OUT;
}

memset(&cfg, 0, sizeof(struct dsopo_config_t));
cfg.protocol = DSPO_BT656;
cfg.bit_width = DSPO_8_BIT;
cfg.separate_sync = false;
cfg.data_seq_sel = DSPO_CB_Y_CR_Y;
cfg.dclk_set.dclk_en = true;
cfg.dclk_set.dclk_invt = false;
cfg.dclk_set.dclk_dly_num = 0;
cfg.output_mode = DSPO_CUSTOM_MODE;
cfg.timing_info.pixel_clk = 80000000;
cfg.timing_info.pixel_repeat = 0;
cfg.timing_info.x_res = 1920;
cfg.timing_info.y_res = 1080;
cfg.timing_info.hor_total_time = 2416;
cfg.timing_info.hor_back_porch = 248;
cfg.timing_info.hor_front_porch = 56;
cfg.timing_info.hor_sync_time = 192;
cfg.timing_info.ver_total_time = 1102;
cfg.timing_info.ver_front_porch = 3;
cfg.timing_info.ver_back_porch = 14;
cfg.timing_info.ver_sync_time = 5;
cfg.timing_info.hor_sync_polarity = 0;
cfg.timing_info.ver_sync_polarity = 1;
cfg.timing_info.b_interlace = 0;
cfg.timing_info.vactive_space = 0;
cfg.timing_info.trd_mode = 0;

arg[0] = myinfo.en;
arg[1] = (unsigned long)&cfg;

ret = ioctl(decd_fd, DSPO_CMD_DEVICE_SWITCH, arg);
if (ret) {
    printf("DSPO_CMD_DEVICE_SWITCH fail!: %d\n", ret);
```

```
    goto OUT;  
}
```

5.2 DSPO_CMD_COMMIT

ioctl 命令，用于传输图像数据。

核心结构体：

```
struct dsopo_dma_info_t {  
    //图像像素格式  
    enum dsopo_infmt_t fmt;  
    //图像宽度，必须与接口分辨率的一致  
    __u32 height;  
    //图像高度，必须与接口分辨率的一致  
    __u32 width;  
    //该成员仅当use_phys_addr == false时有效。代表图像dma内存的文件描述符  
    int fd;  
    //该成员仅当use_phys_addr == true时有效。addr[0]表示图像y分量的起始地址，addr[1]表示图像uv分量的起始地址  
    unsigned long long addr[2];  
    //使用文件描述符来表示图像内存，还是使用物理地址来表示图像内存。  
    bool use_phys_addr;  
};
```

图像传输实例，如下，关于内存申请，这里使用的 ion 的方式，这种方式用文件描述符 (file descriptor) 来代表这块内存，传递 ioctl 的也是 fd，同时 DSPO_CMD_COMMIT 也是支持直接传递物理地址的。只要将 **use_phys_addr** 成员设置成 **true**，然后分别将图像的 y 分量和 uv 分量的起始地址填充到成员 **addr** 数组即可。

```
//打开指定路径图像文件  
myinfo.picFd = open(myinfo.picPath, O_RDONLY);  
if (myinfo.picFd < 0) {  
    perror("open fail:");  
    goto OUT;  
}  
//获取文件大小  
if (stat(myinfo.picPath, &statbuff) < 0) {  
    perror("stat fail:");  
    goto OUT;  
}  
myinfo.filesize = statbuff.st_size;  
  
//申请内存  
ret = ion_memory_request2(&myinfo.ionInfo, myinfo.filesize, ION_HEAP_TYPE_DMA_MASK |  
    ION_HEAP_SYSTEM_CONTIG_MASK);  
if (ret) {  
    printf("ion_memory_request fail:%d\n", ret);  
    goto OUT;  
}  
  
//将图像文件读到上一步申请的内存中。  
readSize = read(myinfo.picFd, (void *)myinfo.ionInfo.virt_addr, myinfo.filesize);
```

```
if (readSize < myinfo.filesize) {
    perror("read fail:");
    goto OUT;
}

//刷cache
ion_flush_cache(&myinfo.ionInfo);

myinfo.info.fmt = DSP0_FORMAT_YUV420_SP_UVUV;
myinfo.info.width = 1280;
myinfo.info.height = 720;
//将申请内存对应的ion fd赋值
myinfo.info.fd = myinfo.ionInfo.fd_data.fd;
//使用ion fd而不是直接使用物理地址。
myinfo.info.use_phy_addr = false;
//打开DSP0节点
decld_fd = open("/dev/dspo", O_RDWR);
if (!decld_fd) {
    perror("\n");
    goto OUT;
}

//传递结构体
arg[0] = (unsigned long)&myinfo.info;

//提交图像
ret = ioctl(decld_fd, DSP0_CMD_COMMIT, arg);
if (ret) {
    printf("DSP0_CMD_COMMIT fail!: %d\n", ret);
    goto OUT;
}

ion_memory_release(&myinfo.ionInfo);
```

说明

关于内存申请，现在越来越多的全志平台支持 **IOMMU** 硬件，但是 **DSPO** 模块现在并不支持 **IOMMU**，所以在用 **ion** 申请内存的时候传递的标志要换成申请连续的 **DMA** 内存 (**ION_HEAP_TYPE_DMA_MASK** | **ION_HEAP_SYSTEM_CONTIG_MASK**)。另外由于 **DSPO** 不支持 **IOMMU**，在与全志平台其它支持 **IOMMU** 的硬件形成通路时，**DSPO** 是无法直接使用经过 **IOMMU** 硬件的内存。

6 调试

6.1 打印信息

1. 确认/dev/dspo 节点是否存在
2. cat /sys/class/dspo/dspo/attr/info

通过这条命令我们可以类似下面的信息：

DSPO 没有使能时的打印：

```
DSPO INFOMATION:  
Enable status:0 irq cnt:0 err_cnt:0 fps:255  
== Hardware resource: ==  
reg base:0xe0868000 irq_no:310  
real clk rate:297000000 and real clk parent rate:297000000, rate to be set:0DSPO INFOMATION  
:
```

DSPO 使能时的打印：

```
DSPO INFOMATION:  
Enable status:1 irq cnt:0 err_cnt:0 fps:255  
== Output setting: ==  
Protocol:BT1120 separate_sync:0 data sequence:0 bit_width:1  
Dclk setting:1 0 0  
== Detail timing info: ==  
Output mode:6  
Resolution:[1280x720P@25Hz]hspw:80, ht:1980 hbp:520 hfp:100 hpol:0  
vspw:5, vt:750 vbp:20 vfp:100 vpol:1  
== Hardware resource: ==  
reg base:0xe0868000 irq_no:310  
real clk rate:74250000 and real clk parent rate:594000000, rate to be set:74250000
```

如果已经送了图片，那么 irq cnt 正常情况会不断增加，增加的频率就等于接口的频率。

6.2 检查硬件

1. 连线是否正确，对应的 IO 电压是否正确
2. 接收端是否正常工作
3. dspo 时钟脚频率是否满足接收端要求。

6.3 调整 timing

由于实现的差异，以及需求的不同，分辨率的 timing 的差异可能造成接收端接收不准确，这个时候应该耐心了解接收端对 timing 的要求，然后通过 **DSPO_CMD_DEVICE_SWITCH** 提到的自定义分辨率来进行实时调整，以及 **dclk_set** 成员来调整时钟信号。

timing 里面那些参数可以通过下面方式得到：

1. 由于发送端配合接收端，所以询问接收端芯片 timing 即可。
2. 可以根据 vesa 标准来设置，主要是 DMT 和 CVT 标准。

其中 DMT，指的是《VESA and Industry Standards and Guidelines for Computer Display Monitor Timing(DMT)》，下载该标准，里面有各种常用分辨率的 timing。

其中的 CVT，指的是《VESA Coordinated Video Timings(CVT) Standard》，该标准提供一种通用公式用于计算出指定分辨率，刷新率等参数的 timing。

可以下载这个 excel 表来计算 VESA Coordinated Video Timing Generator。

调整 timing 过程记住下面几条公式，在满足下面公式下，进行调整。

```
vt = y_res + vbp + vfp + vspw  
ht = x_res + hbp + hfp + hspw  
pixel_clk = ht * vt * fps
```

6.4 调整管脚驱动能力

修改 `linux-4.9/arch/arm/boot/dts/sun8iw19p1-pinctrl.dtsi`,

总共有几组管脚定义，其中，

2. `bt656_emb` 开头的表示 BT656,8 位数据线，内嵌同步
3. `bt656_sep` 开头的表示 BT656,8 位数据线，外同步
4. `bt1120_emb` 开头的表示 BT1120,16 位数据线，内嵌同步
5. `bt1120_sep` 开头的表示 BT1120,16 位数据线，外同步

要修改驱动能力，只需要修改上面几组管脚定义中的 `allwinner,drive` 成员即可，范围是 0 到 3，数字越大驱动能力越强。

7 新平台移植

将 DSPO 模块移植到新的平台上，如果内核版本保持一致，那么主要就是修改 dts 就行。

在linux-4.9/arch/arm/boot/dts/sun8iw19p1.dtsi新增 dsopo 节点。对于新平台要更换其中的基地址，irq 号和时钟名等。

对于管脚 (pinctrl) 部分名字，这里是固定的，不能改变，否则驱动可能无法正常工作。

```
dsopo: dsopo@0 {
    compatible = "allwinner,sunxi-dspo";
    reg = <0x0 0x06543000 0x0 0x3ff>;
    interrupts = <GIC_SPI 93 IRQ_TYPE_LEVEL_HIGH>;
    clocks = <&clk_dspo>;
    pinctrl-names = "bt656-seperate", "bt656-seperate-sleep",
    "bt656-embed", "bt656-embed-sleep",
    "bt1120-seperate", "bt1120-seperate-sleep",
    "bt1120-embed", "bt1120-embed-sleep";
    status = "okay";
    pinctrl-0 = <&bt656_sep_pins_a>;
    pinctrl-1 = <&bt656_sep_pins_b>;
    pinctrl-2 = <&bt656_emb_pins_a>;
    pinctrl-3 = <&bt656_emb_pins_b>;
    pinctrl-4 = <&bt1120_sep_pins_a>;
    pinctrl-5 = <&bt1120_sep_pins_b>;
    pinctrl-6 = <&bt1120_emb_pins_a>;
    pinctrl-7 = <&bt1120_emb_pins_b>;
};
```

修改linux-4.9/arch/arm/boot/dts/sun8iw19p1-pinctrl.dtsi，根据 IC 来调整下面几种情况的管脚配置。

```
bt656_sep_pins_a: bt656_sep@0 {
    allwinner,pins = "PD1", "PD2", "PD3", "PD4",
    "PD5", "PD6", "PD7", "PD8", "PD18", "PD19", "PD20", "PD21";
    allwinner,function = "bt1120";
    allwinner,muxsel = <4>;
    allwinner,drive = <1>;
    allwinner,pull = <0>;
};

bt656_sep_pins_b: bt656_sep@1 {
    allwinner,pins = "PD1", "PD2", "PD3", "PD4",
    "PD5", "PD6", "PD7", "PD8", "PD18", "PD19", "PD20", "PD21";
    allwinner,function = "bt1120";
    allwinner,muxsel = <7>;
    allwinner,drive = <1>;
    allwinner,pull = <0>;
};

bt656_emb_pins_a: bt656_emb@2 {
```

```
allwinner,pins = "PD1", "PD2", "PD3", "PD4",
"PD5", "PD6", "PD7", "PD8", "PD18";
allwinner,function = "bt1120";
allwinner,muxsel = <4>;
allwinner,drive = <1>;
allwinner,pull = <0>;
};

bt656_emb_pins_b: bt656_emb@3 {
    allwinner,pins = "PD1", "PD2", "PD3", "PD4",
"PD5", "PD6", "PD7", "PD8", "PD18";
    allwinner,function = "bt1120";
    allwinner,muxsel = <7>;
    allwinner,drive = <1>;
    allwinner,pull = <0>;
};

bt1120_sep_pins_a: bt1120_sep@4 {
    allwinner,pins = "PD1", "PD2", "PD3", "PD4",
"PD5", "PD6", "PD7", "PD8", "PD10", "PD11", "PD12",
"PD13", "PD14", "PD15", "PD16", "PD17",
"PD18", "PD19", "PD20", "PD21";
    allwinner,function = "bt1120";
    allwinner,muxsel = <4>;
    allwinner,drive = <1>;
    allwinner,pull = <0>;
};

bt1120_sep_pins_b: bt1120_sep@5 {
    allwinner,pins = "PD1", "PD2", "PD3", "PD4",
"PD5", "PD6", "PD7", "PD8", "PD10", "PD11", "PD12",
"PD13", "PD14", "PD15", "PD16", "PD17",
"PD18", "PD19", "PD20", "PD21";
    allwinner,function = "bt1120";
    allwinner,muxsel = <7>;
    allwinner,drive = <1>;
    allwinner,pull = <0>;
};

bt1120_emb_pins_a: bt1120_emb@6 {
    allwinner,pins = "PD1", "PD2", "PD3", "PD4",
"PD5", "PD6", "PD7", "PD8", "PD10", "PD11", "PD12",
"PD13", "PD14", "PD15", "PD16", "PD17", "PD18";
    allwinner,function = "bt1120";
    allwinner,muxsel = <4>;
    allwinner,drive = <1>;
    allwinner,pull = <0>;
};

bt1120_emb_pins_b: bt1120_emb@7 {
    allwinner,pins = "PD1", "PD2", "PD3", "PD4",
"PD5", "PD6", "PD7", "PD8", "PD10", "PD11", "PD12",
"PD13", "PD14", "PD15", "PD16", "PD17", "PD18";
    allwinner,function = "bt1120";
    allwinner,muxsel = <7>;
    allwinner,drive = <1>;
    allwinner,pull = <0>;
};
```

1. _a结尾的表示使能时用的，_b结尾的表示 disable 时使用的。
2. bt656_emb开头的表示 BT656,8 位数据线，内嵌同步

3. bt656_sep开头的表示 BT656,8 位数据线，外同步
4. bt1120_emb开头的表示 BT1120,16 位数据线，内嵌同步
5. bt1120_sep开头的表示 BT1120,16 位数据线，外同步



著作权声明

版权所有 © 2021 珠海全志科技股份有限公司。保留一切权利。

本文档及内容受著作权法保护，其著作权由珠海全志科技股份有限公司（“全志”）拥有并保留一切权利。

本文档是全志的原创作品和版权财产，未经全志书面许可，任何单位和个人不得擅自摘抄、复制、修改、发表或传播本文档内容的部分或全部，且不得以任何形式传播。

商标声明

  **全志科技**  (不完全列举) 均为珠海全志科技股份有限公司的商标或者注册商标。在本文档描述的产品中出现的其它商标，产品名称，和服务名称，均由其各自所有人拥有。

免责声明

您购买的产品、服务或特性应受您与珠海全志科技股份有限公司（“全志”）之间签署的商业合同和条款的约束。本文档中描述的全部或部分产品、服务或特性可能不在您所购买或使用的范围内。使用前请认真阅读合同条款和相关说明，并严格遵循本文档的使用说明。您将自行承担任何不当使用行为（包括但不限于如超压，超频，超温使用）造成的不利后果，全志概不负责。

本文档作为使用指导仅供参考。由于产品版本升级或其他原因，本文档内容有可能修改，如有变更，恕不另行通知。全志尽全力在本文档中提供准确的信息，但并不确保内容完全没有错误，因使用本文档而发生损害（包括但不限于间接的、偶然的、特殊的损失）或发生侵犯第三方权利事件，全志概不负责。本文档中的所有陈述、信息和建议并不构成任何明示或暗示的保证或承诺。

本文档未以明示或暗示或其他方式授予全志的任何专利或知识产权。在您实施方案或使用产品的过程中，可能需要获得第三方的权利许可。请您自行向第三方权利人获取相关的许可。全志不承担也不代为支付任何关于获取第三方许可的许可费或版税（专利税）。全志不对您所使用的第三方许可技术做出任何保证、赔偿或承担其他义务。