



AX MIPI SWITCH 开发参考

文档版本：V1.0

发布日期：2024/10/18

目录

前 言.....	4
修订历史.....	5
1 概述.....	6
1.1 功能示图.....	6
1.2 功能介绍.....	7
2 API 参考	8
ax_mipi_switch_init.....	8
ax_mipi_switch_start	9
ax_mipi_switch_stop	10
ax_mipi_switch_change.....	11
ax_mipi_switch_set_fps.....	12
3 数据结构.....	13
AX_MIPI_SWITCH_WORK_MODE_E.....	13
AX_VSYNC_SNS_REG_T	15
AX_VSYNC_SNS_I2C_T	16
AX_VSYNC_INFO_T	18
AX_SWITCH_INFO_T.....	20
4 配置流程说明.....	21
4.1 初始化 MIPI SWITCH。	21
4.2 使能 MIPI SWITCH。	22
4.3 停止 MIPI SWITCH.....	22

4.4 SENSOR 通路切换	22
4.5 SENSOR 帧率切换	22
4.6 DTS 配置	23
5 proc 信息说明	25
5.1 mipi switch attr	25
5.2 mipi switch status.....	25
6 开源代码接口说明.....	27
CB_VIN_INT_NOTIFY_RegisterCallback(dev_id, int_callback)	27
CB_VIN_Vsync_Resume_Callback(pipe_id, cb)	27
CB_VIN_Vsync_Suspend_Callback(pipe_id, cb).....	27
CB_VIN_Get_Boot_Mode(dev_id).....	27
CB_VIN_FTC_TRIGER_START (dev_id).....	27
CB_VIN_SetHsyncAttr(dev_id, hsync_attr)	27
CB_VIN_SetVsyncAttr(dev_id, vsync_attr)	27
CB_VIN_FTC_SetTimingAttr(dev_id, idx, timing_attr)	28
CB_VIN_EnableVsync(dev_id, sync_idx).....	28
CB_VIN_DisableVsync(dev_id, sync_idx).....	28
CB_VIN_EnableFtcTiming(dev_id, idx).....	28
CB_VIN_DisableFtcTiming(dev_id, idx)	28

权利声明

爱芯元智半导体股份有限公司或其许可人保留一切权利。

非经权利人书面许可，任何单位和个人不得擅自摘抄、复制本文档内容的部分或全部，并不得以任何形式传播。

注意

您购买的产品、服务或特性等应受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的购买或使用范围之内。除非商业合同另有约定，本公司对本文档内容不做任何明示或默示的声明或保证。

由于产品版本升级或其他原因，本文档内容会不定期进行更新。除非另有约定，本文档仅作为使用指导，本文档中的所有陈述、信息和建议不构成任何明示或暗示的担保。

前言

适用产品

AX630C/AX620Q

适读人群

- 技术支持工程师
- 软件开发工程师

符号与格式定义

符号/格式	说明
xxx	表示您可以执行的命令行。
说明/备注:	表示您在使用产品的过程中，我们向您说明的事项。
注意:	表示您在使用产品的过程中，需要您特别注意的事项。

修订历史

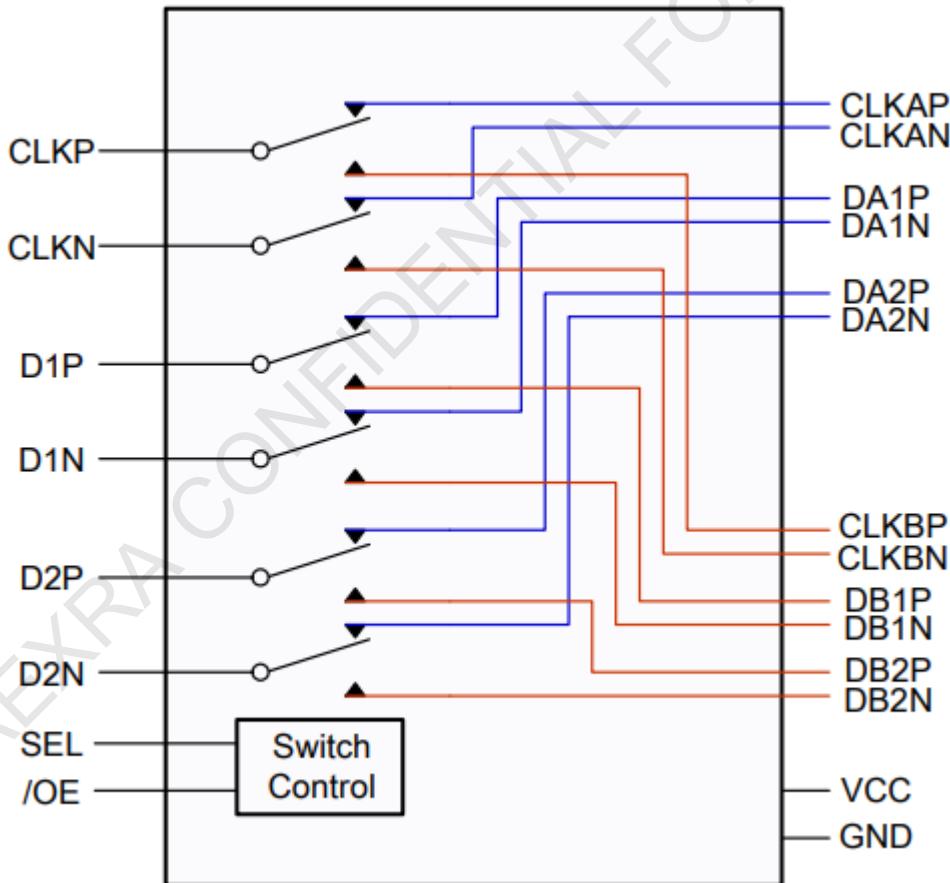
文档版本	发布时间	修订说明
V1.0	2024/10/18	文档初版
V1.1	2024/11/15	增加 eVsyncType 枚举和说明

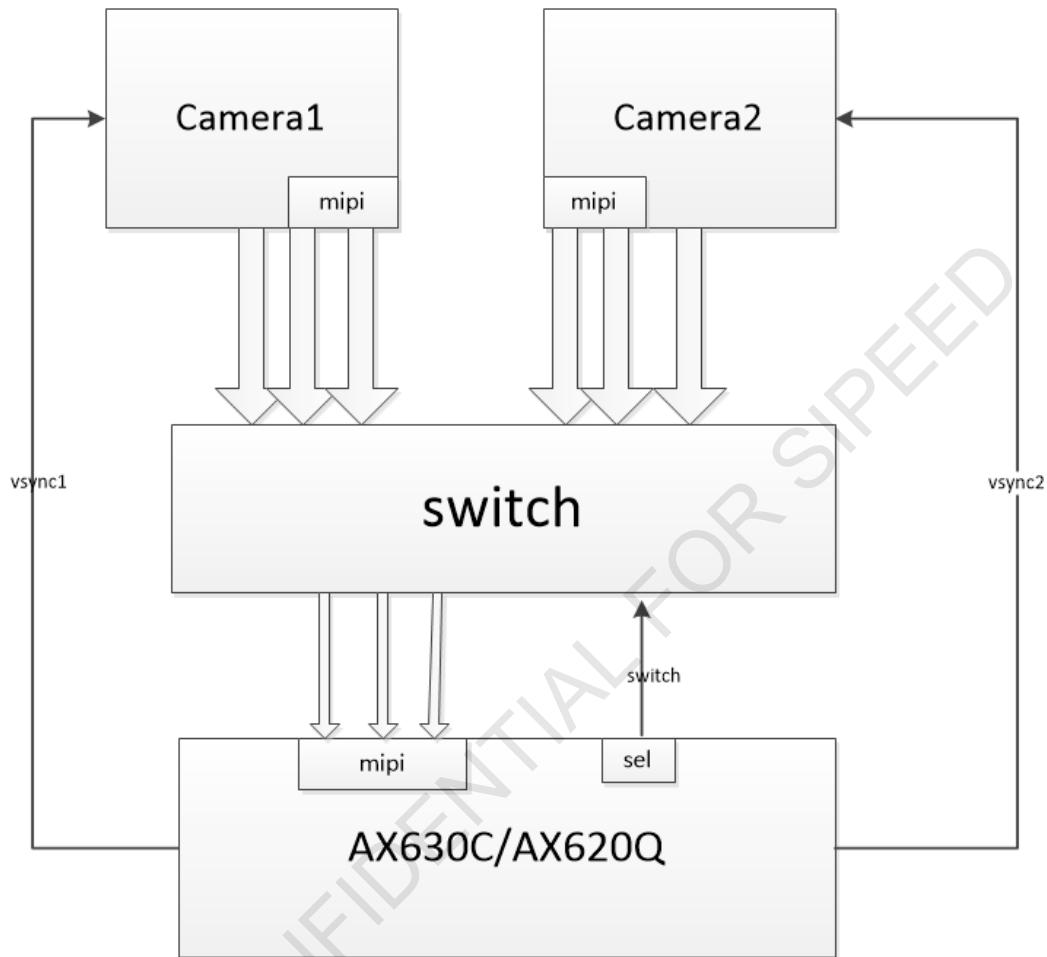
1 概述

MIPI SWITCH 旨在两个 MIPI 设备（如摄像头或 LCD 显示屏）和主板上的 MAP 之间进行切换。

通过控制 SWITCH 信号，来选择一路 MIPI 信号接收。

1.1 功能示图





1.2 功能介绍

- 支持输出 2 路 VSYNC 信号。
- 支持输出 1 路 SWITCH 信号。
- 支持 VSYNC 信号 SWITCH 信号动态切换。
- 支持 VSYNC 信号频率动态调节。

ax_mipi switch ko

实现 2 路 sensor vsync 和 switch 信号的控制。

2 API 参考

ax_mipi_switch_init

【描述】

初始化 MIPI SWITCH。

【语法】

```
int ax_mipi_switch_init(AX_SWITCH_INFO_T *tSwitchInfo);
```

【参数】

参数名称	描述	输入/输出
tSwitchInfo	AX_SWITCH_INFO_T 相关参数	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: mipi_switch.h

【注意】

无

【举例】

无

【相关主题】

无

ax_mipi_switch_start

【描述】

使能 MIPI SWITCH，输出 VSYNC。

【语法】

```
int ax_mipi_switch_start(void);
```

【参数】

参数名称	描述	输入/输出
无		

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: mipi_switch.h

【注意】

无

【举例】

无

【相关主题】

无

ax_mipi_switch_stop

【描述】

停止 SWITCH 和 VSYNC 信号输出。

【语法】

```
int ax_mipi_switch_stop(void);
```

【参数】

参数名称	描述	输入/输出
无		

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: mipi_switch.h

【注意】

无

【举例】

无

【相关主题】

无

ax_mipi_switch_change

【描述】

切换 sensor 通路选择。

【语法】

```
int ax_mipi_switch_change(int nSnsId);
```

【参数】

参数名称	描述	输入/输出
nSnsId	切换到对应 id 的 sensor	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: mipi_switch.h

【注意】

禁止在 AOV 过程中调用此接口

【举例】

无

【相关主题】

无

ax_mipi_switch_set_fps

【描述】

改变 vsync 输出频率来切换 sensor 帧率，适配 sensor 的 slow shutter 功能。

【语法】

```
int mipi_switch_set_fps(AX_VSYNC_INFO_T * pVsyncInfo);
```

【参数】

参数名称	描述	输入/输出
pVsyncInfo	AX_VSYNC_INFO_T 相关参数	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: mipi_switch.h

【注意】

无

【举例】

无

【相关主题】

无

3 数据结构

AX_MIPI_SWITCH_WORK_MODE_E

【说明】

SWITCH PIN 工作模式。

【定义】

```
typedef enum _AX_MIPI_SWITCH_WORK_MODE_E {  
    AX_MIPI_SWITCH_STAY_LOW,  
    AX_MIPI_SWITCH_STAY_HIGH,  
    AX_MIPI_SWITCH_SWITCH_PERIODIC,  
} AX_MIPI_SWITCH_WORK_MODE_E;
```

【成员】

成员名称	描述
AX_MIPI_SWITCH_STAY_LOW	SWITCH PIN 工作在低电平
AX_MIPI_SWITCH_STAY_HIGH	SWITCH PIN 工作在高电平
AX_MIPI_SWITCH_SWITCH_PERIODIC	SWITCH PIN 周期性的高低电平转换

【注意】

【相关数据类型及接口】

无

AX_MIPI_SWITCH_VSYNC_TYPE_E

【说明】

VSYNC PIN 脚选择。

【定义】

```
typedef enum _AX_MIPI_SWITCH_VSYNC_TYPE_E {  
    AX_MIPI_SWITCH_FSYNC_VSYNC,  
    AX_MIPI_SWITCH_FSYNC_FLASH,  
} AX_MIPI_SWITCH_VSYNC_TYPE_E;
```

【成员】

成员名称	描述
AX_MIPI_SWITCH_FSYNC_VSYNC	VSYNC PIN 作为 VSYNC
AX_MIPI_SWITCH_FSYNC_FLASH	FLASH PIN 作为 VSYNC

【注意】

当选择 VSYNC PIN 作为 vsync 时,我们只能选择 GPIO15(SEN_VSYNC_D1).

当选择 FLASH PIN 作为 vsync 时, 可以选择 SEN_FLASH_D0~D5 对应的 GPIO。

【相关数据类型及接口】

无

AX_VSYNC_SNS_REG_T

【说明】

MIPI SWITCH 同步要配置 SENSOR 的寄存器的地址和值。

【定义】

```
typedef struct _ AX_VSYNC_SNS_REG_T {  
    unsigned int nRegAddr;  
    unsigned int nData;  
} AX_VSYNC_SNS_REG_T;
```

【成员】

成员名称	描述
nRegAddr	Sensor 寄存器地址
nData	Sensor 寄存器值

【注意】

【相关数据类型及接口】

无

AX_VSYNC_SNS_I2C_T

【说明】

MIPI SWITCH 同步要配置 SENSOR 的 I2C 地址及寄存器个数内容等信息。

【定义】

```
typedef struct _ AX_VSYNC_SNS_I2C_T {
    unsigned char    nI2cNode;
    unsigned char    nI2cAddr;
    unsigned char    nDelayFrmNum;
    unsigned char    nIntPos;
    unsigned char    nRegNum;
    unsigned int     nAddrByteNum;
    unsigned int     nDataByteNum;
    AX_VSYNC_SNS_REG_T tSnsReg[AX_MIPI_SWITCH_SNS_REG_NUM];
} AX_VSYNC_SNS_I2C_T;
```

【成员】

成员名称	描述
nI2cNode	Sensor 所在的 i2c 总线
nI2cAddr	Sensor 的 slave addr
nDelayFrmNum	Sensor 配置在第几帧生效
nIntPos	未使用
nRegNum	配置的寄存器个数
nAddrByteNum	寄存器地址所占字节数
nDataByteNum	寄存器值所占字节数
tSnsReg	AX_VSYNC_SNS_REG_T 数据结构

【注意】

nDelayFrmNum: 当 sensor 工作在 slave mode 时, vts 是隔 1 帧生效, 也就是当前帧配置, 下一帧生效。而 Master 时是 N+2 帧生效, 所以这里 nDelayFrmNum 一般都是需要配置为 1, 并且不能配置成小于 1 的值。

【相关数据类型及接口】

无

AEXRA CONFIDENTIAL FOR SIPEED

AX_VSYNC_INFO_T

【说明】

MIPI SWITCH 同步要配置 SENSOR 的 i2c info 和帧率信息。

【定义】

```
typedef struct _AX_VSYNC_INFO_T {  
    AX_VSYNC_SNS_I2C_T tI2cInfo[AX_MIPI_SWITCH_PIPE_NUM];  
    int nFps;  
} AX_VSYNC_INFO_T;
```

【成员】

成员名称	描述
tI2cInfo	AX_VSYNC_SNS_I2C_T 相关参数
nFps	SENSOR 帧率

【注意】

【相关数据类型及接口】

无

AX_SWITCH_SNS_INFO_T

【说明】

MIPI SWITCH 同步要配置 SENSOR 的 snsid、pipeid、lenstype vsync type 和 workmode。

【定义】

```
typedef struct _AX_SWITCH_SNS_INFO_T {  
    int nSnsId;  
    int nPipeId;  
    AX_LENS_TYPE_E eLensType;  
    AX_MIPI_SWITCH_WORK_MODE_E eWorkMode;  
    AX_MIPI_SWITCH_VSYNC_TYPE_E eVsyncType;  
} AX_SWITCH_SNS_INFO_T;
```

【成员】

成员名称	描述
nSnsId	Sensor 的 device id
nPipeId	Sensor 的 pipe id
eLensType	Sensor 的 lens 类型
eWorkMode	Sensor 选通时 switch pin 的电平状态
eVsyncType	Sensor 接入的 vsync 脚的类型

【注意】

当两个 sensor 共用一个 vsync 时, 请将两个 sensor 的 eVsyncType 配置一致, 并且 dts 里两个 vysnc 也配置成相同的 pin.

【相关数据类型及接口】

无

AX_SWITCH_INFO_T

【说明】

Mipi switch 初始化要配置 sensor 帧率、pipe 数、switch pin 工作模式，sns_info 等信息。

【定义】

```
typedef struct _AX_SWITCH_INFO_T {  
    int nFps;  
    int nPipeNum;  
    AX_MIPI_SWITCH_WORK_MODE_E eWorkMode;  
    AX_MIPI_SWITCH_VSYNC_TYPE_E eVsyncType;  
    AX_SWITCH_SNS_INFO_T tSnsInfo[AX_MIPI_SWITCH_PIPE_NUM];  
} AX_SWITCH_INFO_T;
```

【成员】

成员名称	描述
nFps	Sensor 帧率
nPipeNum	Switch 的两个 sensor 占用的 Pipe 数。 共 pipe 时，pipe_num 为 1； 不共 pipe 时，为 2。
eWorkMode	Switch pin 工作模式
eVsyncType	Vsync pin 选择
tSnsInfo	SWITCH_SNS_INFO_T 相关参数

【注意】

【相关数据类型及接口】

无

4 配置流程说明

4.1 初始化 MIPI SWITCH。

```
AX_SWITCH_INFO_T switch_info = {0};  
switch_info.nFps = 30;  
switch_info.nPipeNum = 1;  
switch_info.tSnsInfo[0].nSnsId = 1;  
switch_info.tSnsInfo[0].nPipeId = 1;  
switch_info.tSnsInfo[0].eLensType = AX_LENS_TYPE_LONG_FOCAL;  
switch_info.tSnsInfo[0].eWorkMode = AX_MIPI_SWITCH_STAY_LOW;  
switch_info.tSnsInfo[0].eVsyncType = AX_MIPI_SWITCH_FSYNC_FLASH;  
switch_info.tSnsInfo[1].nSnsId = 2;  
switch_info.tSnsInfo[1].nPipeId = 1;  
switch_info.tSnsInfo[1].eLensType = AX_LENS_TYPE_WIDE_FIELD;  
switch_info.tSnsInfo[1].eWorkMode = AX_MIPI_SWITCH_STAY_HIGH;  
switch_info.tSnsInfo[1].eVsyncType = AX_MIPI_SWITCH_FSYNC_VSYNC;  
switch_info.eWorkMode = AX_MIPI_SWITCH_STAY_HIGH;  
ax_mipi_switch_init(&switch_info);  
ax_mipi_switch_start();
```

nFps: 这里需要配置 VSYNC 的频率

nPipeNum: 当 SWITCH PIN 一直保持高/低电平时, 共用一个 PIPE, 两个 SENSOR 只有一个在工作, pipe_num 配置为 1。

当 SWITCH PIN 周期性的切换时, 各用一个 PIPE, 两个 SENSOR 都在工作, pipe_num 配置为 2。

nSnsId: nSnsId 为 1 就是原来双摄时的 SENSOR1, 此时 SWITCH PIN 工作在低电平。nSnsId 为 2 是 SWITCH 做切换新增的 SENSOR, 当 SWITCH PIN 切到高电平时切换到这个 SENSOR。

eWorkMode: 选通 SENSOR1 或 SENSOR2。

eVsyncType: SENSOR 接入的 VSYNC PIN 类型。

4.2 使能 MIPI SWITCH。

```
ax_mipi_switch_start();
```

使能 VSYNC/SWITCH 信号输出。

当 SENSOR 工作在 SLAVE MODE, STREAM ON 是触发曝光, VSYNC 触发 READ OUT, 因此第一帧的曝光存在过曝或欠曝的情况。

所以此函数调用应该放到 SENSOR 开流函数之后比较合适, 并添加适当的延时, 以此来调整曝光的时间。也可以修改开源代码 ax_mipi_switch.ko 选择丢弃第一帧。

4.3 停止 MIPI SWITCH

```
int ax_mipi_switch_stop(void);
```

Mipi_switch_stop()会停止 vsync 输出。

对于 slave sensor 来说, 需要配置没有 vsync 就不出帧, 收到 vsync 才出帧.

4.4 SENSOR 通路切换

```
int ax_mipi_switch_change(int nSnsId);
```

用于切换 SENSOR, 参数为 tSnsInfo 信息中所填的 nSnsId。

4.5 SENSOR 帧率切换

```
int ax_mipi_switch_set_fps(AX_VSYNC_INFO_T *pVsyncInfo);
```

帧率切换目前只用在 sensor 驱动接口 pfn_sensor_set_slow_fps 中, 用于 slow shutter 切换帧率。因为 vsync 配置与 sensor vts 配置存在一个需要同步生效的关系, 所以他们在 mipi switch ko 中去配置, 保持同步。

示例如下:

```
msp/component/isp_proton/sensor/driver/smartsens_sc200ai/sc200ai_ae_ctrl.c
```

```

if (sns_obj->sns_mode_obj.eMasterSlaveSel == AX_SNS_SYNC_SLAVE) {
    for (i = 0; i < AX_VIN_MAX_PIPE_NUM; i++) {
        if(gSc200aiSlaveMode[i] == 0) {
            continue;
        }
        tVsyncInfo.tI2cInfo[j].nI2cNode = sc200ai_get_bus_num(nPipeId: i);
        tVsyncInfo.tI2cInfo[j].nI2cAddr = gSc200aiSlaveAddr[i];
        tVsyncInfo.tI2cInfo[j].nRegNum = 2;
        tVsyncInfo.tI2cInfo[j].tSnsReg[0].nRegAddr = SC200AI_VTS_LONG_H;
        tVsyncInfo.tI2cInfo[j].tSnsReg[0].nData = (vts >> 8) & 0x7f;
        tVsyncInfo.tI2cInfo[j].tSnsReg[1].nRegAddr = SC200AI_VTS_LONG_L;
        tVsyncInfo.tI2cInfo[j].tSnsReg[1].nData = vts & 0xff;
        tVsyncInfo.tI2cInfo[j].nAddrByteNum = SC200AI_ADDR_BYTE;
        tVsyncInfo.tI2cInfo[j].nDataByteNum = SC200AI_DATA_BYTE;
        tVsyncInfo.tI2cInfo[j].nDelayFrmNum = 1;
        tVsyncInfo.nFps = (AX_U32)fFps;
        j++;
    }
}

if (sns_obj->sns_mode_obj.eMasterSlaveSel == AX_SNS_SYNC_SLAVE) {
    ax_mipi_switch_set_fps(pVsyncInfo: &tVsyncInfo);
} else {
    sc200ai_set_vts(nPipeId, vts);
}

```

注意: nDelayFrmNum 表示 vts 配置会在第几帧生效, 1 表示隔 1 帧生效, 2 表示隔 2 帧生效, 请配置正确。

4.6 DTS 配置

kernel/linux/linux-4.19.125/arch/arm64/boot/dts/axera/AX620E.dtsi

```

mipi_switch: mipi_switch@2406200 {
    compatible = "axera,mipi_switch";
    sensor-vsync0 = <15>; /*GPIO0-A15 for sensor vsync*/
    sensor-vsync1 = <15>; /*GPIO1-A2 for sensor vsync*/
    switch0 = <32>; /*GPIO1-A0 for mipi switch*/
};

```

一路 VSYNC 用的是 SEN_VSYNC_D1, 对应的 GPIO 是 GPIO0-A15.

一路 VSYNC 用的是 SEN_FLASH_D4, 对应的 GPIO 是 GPIO1-A2.

SWITCH 用的是 SEN_FLASH_D3, 对应的 GPIO 是 GPIO1-A0.

AEXRA CONFIDENTIAL FOR SIPEED

5 proc 信息说明

5.1 mipi switch attr

Cat /proc/ax_proc/mipi_switch/attr

```
/ # cat /proc/ax_proc/mipi_switch/attr
----- MIPI SWITCH VERSION -----
[Axera version]: axdrv v2.17.0 Oct 18 2024 10:39:30
-----MIPI SWITCH ATTR-----
MipiDev  Vsync_gpio      switch_gpio      switch_Pin_State
1        gpio34          gpio35          stay low
2        gpio15          gpio35          stay high
```

参数名称	描述
MipiDev	Device number
Vsync_gpio	Device 对应的 vsync gpio number
Switch_gpio	Device 对应的 switch gpio number
Switch_pin_state	Device 选通时对应的 switch gpio 状态

5.2 mipi switch status

Cat /proc/ax_proc/mipi_switch/status

```
/ # cat /proc/ax_proc/mipi_switch/status
----- MIPI SWITCH VERSION -----
[Axera version]: axdrv V2.17.0 Oct 18 2024 10:39:30

-----MIPI SWITCH STATUS-----
Current_Dev  Switch_Pin_State      suspend_status    fps
2           stay high            resume           15

-----MIPI SWITCH SENSOR INFO-----
I2cNode    slaveaddr          reg          val
6          0x32                0x320e      0x12
6          0x32                0x320f      0xbc
0          0x32                0x320e      0x12
0          0x32                0x320f      0xbc
```

参数名称	描述
Current_Dev	当前选通的 device number
Switch_Pin_State	当前 switch pin 所处的状态
Suspend_Status	这个只对 aov 有用，表明是否走了休眠唤醒接口
fps	当前 vsync 的帧率
I2cNode	Switch sensor 对应的 i2c number
Salveaddr	Switch sensor 对应的 slave address
reg	Switch sensor 对应的 vts 寄存器
val	Switch sensor 对应的 vts 寄存器的值

6 开源代码接口说明

CB_VIN_INT_NOTIFY_RegisterCallback(dev_id, int_callback)

注册中断回调函数，用于 sensor sof/eof 中断处理。

CB_VIN_Vsync_Resume_Callback(pipe_id, cb)

注册唤醒回调函数，用于休眠唤醒。

CB_VIN_Vsync_Suspend_Callback(pipe_id, cb)

注册休眠回调函数，用于休眠唤醒。

CB_VIN_Get_Boot_Mode(dev_id)

获取当前启动模式接口，区分正常模式和快启模式。

CB_VIN_FTC_TRIGGER_START (dev_id)

调用 FTC TRIGER 接口，触发 VSYNC 输出。

CB_VIN_SetHsyncAttr(dev_id, hsync_attr)

配置 HSYNC 的相关属性，配置一行的时间作为基础时间，与 SENSOR 的 HTS 无关。

CB_VIN_SetVsyncAttr(dev_id, vsync_attr)

配置 VSYNC 的相关属性，根据 HSYNC 和 FPS，可以算出 VSYNC 应该配置为多少。

CB_VIN_FTC_SetTimingAttr(dev_id, idx, timing_attr)

配置 VSYNC 或 SWITCH 的相关属性。

CB_VIN_EnableVsync(dev_id, sync_idx)

使能 VSYNC 信号。

CB_VIN_DisableVsync(dev_id, sync_idx)

停止 VSYNC 信号。

CB_VIN_EnableFtcTiming(dev_id, idx)

使能 VSYNC 信号或 SWITCH 信号。

CB_VIN_DisableFtcTiming(dev_id, idx)

停止 VSYNC 信号或 SWITCH 信号。

☞ 说明

- 一路 VSYNC 输出用的由 VSYNC 控制，一路 VSYNC 输出由 FLASH 控制，SWITCH 输出由 FLASH 控制。
- 由 VSYNC 控制的这一路，对应我们硬件上的 SEN_VSYNC_D1，不要用 SEN_VSYNC_D0。
- 由 FLASH 控制的这两路，对应我们硬件上的 SEN_FLASH_D0~5.可以根据当前硬件情况选择没有被占用的两路即可。