



A40i 系列 & T3 系列

Linux-5.10 CSI 模块开发指南

版本号: 1.0
发布日期: 2022.11.8

版本历史

版本号	日期	制/修订人	内容描述
1.0	2022.11.8	AWA1689	初始版本

目 录

1 前言	1
1.1 文档简介	1
1.2 目标读者	1
1.3 适用范围	1
2 模块介绍	2
2.1 模块功能介绍	2
2.2 相关术语介绍	2
2.3 模块配置介绍	2
2.3.1 kernel menuconfig 配置	2
2.3.2 Device Tree 配置说明	3
2.4 源码模块结构	6
3 V4L2 接口描述	9
3.1 VIDIOC_QUERYCAP	9
3.1.1 Parameters	9
3.1.2 Returns	9
3.1.3 Description	9
3.2 VIDIOC_ENUM_INPUT	9
3.2.1 Parameters	9
3.2.2 Returns	10
3.2.3 Description	10
3.3 VIDIOC_S_INPUT	10
3.3.1 Parameters	10
3.3.2 Returns	10
3.3.3 Description	10
3.4 VIDIOC_G_INPUT	11
3.4.1 Parameters	11
3.4.2 Returns	11
3.4.3 Description	11
3.5 VIDIOC_S_PARM	11
3.5.1 Parameters	11
3.5.2 Returns	12
3.5.3 Description	12
3.6 VIDIOC_G_PARM	12
3.6.1 Parameters	12
3.6.2 Returns	12
3.6.3 Description	12
3.7 VIDIOC_ENUM_FMT	13

3.7.1	Parameters	13
3.7.2	Returns	13
3.7.3	Description	13
3.8	VIDIOC_TRY_FMT	13
3.8.1	Parameters	13
3.8.2	Returns	14
3.8.3	Description	14
3.9	VIDIOC_S_FMT	14
3.9.1	Parameters	14
3.9.2	Returns	14
3.9.3	Description	14
3.10	VIDIOC_G_FMT	15
3.10.1	Parameters	15
3.10.2	Returns	15
3.10.3	Description	15
3.11	VIDIOC_OVERLAY	15
3.11.1	Parameters	15
3.11.2	Returns	15
3.11.3	Description	16
3.12	VIDIOC_REQBUFS	16
3.12.1	Parameters	16
3.12.2	Returns	16
3.12.3	Description	16
3.13	VIDIOC_QUERYBUF	17
3.13.1	Parameters	17
3.13.2	Returns	17
3.13.3	Description	17
3.14	VIDIOC_DQBUF	17
3.14.1	Parameters	17
3.14.2	Returns	18
3.14.3	Description	18
3.15	VIDIOC_QBUF	18
3.15.1	Parameters	18
3.15.2	Returns	18
3.15.3	Description	18
3.16	VIDIOC_STREAMON	18
3.16.1	Parameters	18
3.16.2	Returns	18
3.16.3	Description	19
3.17	VIDIOC_STREAMOFF	19
3.17.1	Parameters	19
3.17.2	Returns	19
3.17.3	Description	19

3.18 VIDIOC_QUERYCTRL	19
3.18.1 Parameters	19
3.18.2 Returns	20
3.18.3 Description	20
3.19 VIDIOC_S_CTRL	20
3.19.1 Parameters	20
3.19.2 Returns	20
3.19.3 Description	20
3.20 VIDIOC_G_CTRL	20
3.20.1 Parameters	20
3.20.2 Returns	21
3.20.3 Description	21
3.21 VIDIOC_ENUM_FRAMEIZES	21
3.21.1 Parameters	21
3.21.2 Returns	22
3.21.3 Description	22
3.22 VIDIOC_ENUM_FRAMEINTERVALS	22
3.22.1 Parameters	22
3.22.2 Returns	22
3.22.3 Description	23
4 模块使用范例	24
4.1 测试 demo	24
4.2 调用流程	25
5 FAQ	26
5.1 调试方法	26
5.1.1 中断状态	26
5.2 常见问题	26
5.2.1 I2C 不通	26
5.2.2 sensor 不出图	26
5.2.3 已出图但画面是绿色或者粉红色	27
5.2.4 I2c 已通，但是读所有 sensor 寄存器值都为 0	27
5.2.5 没有 video 节点	27

1 前言

1.1 文档简介

介绍 VFE (video front end) 驱动配置，API 接口和上层使用方法。

1.2 目标读者

camera 驱动开发、维护人员和应用开发人员。

1.3 适用范围

表 1-1: 适用产品列表

平台	内核版本	驱动文件
T3/T3-C/T3-Pro	Linux-5.10	longan/bsp/drivers/vfe/*.c
A40i-H/A40i-C	Linux-5.10	longan/bsp/drivers/vfe/*.c

2 模块介绍

2.1 模块功能介绍

1. CSI 主要实现视频数据的捕捉；
2. CSI 也可对捕获的视频数据做水平以及垂直的剪裁；
3. CSI0 支持 BT656/BT1120，支持 8bit YUV422 sensor；
4. CSI1 支持 BT656，支持 8bit YUV422 sensor；

2.2 相关术语介绍

表 2-1: 软件术语

相关术语	解释说明
CCI	Camera Control Interface 摄像头控制接口
MCLK	Master clock (From AP to camera) 摄像头主时钟
PCLK	Pixel clock (From camera to AP, Sampling clock for data-bus) 像素时钟
YUV	Color Presentation (Y for luminance, U&V for Chrominance) 图像数据格式

2.3 模块配置介绍

2.3.1 kernel menuconfig 配置

进入 Allwinner BSP > Device Drivers > VFE (camera) Drivers，选择 v4l2 driver for SUNXI 即可，如下图所示。

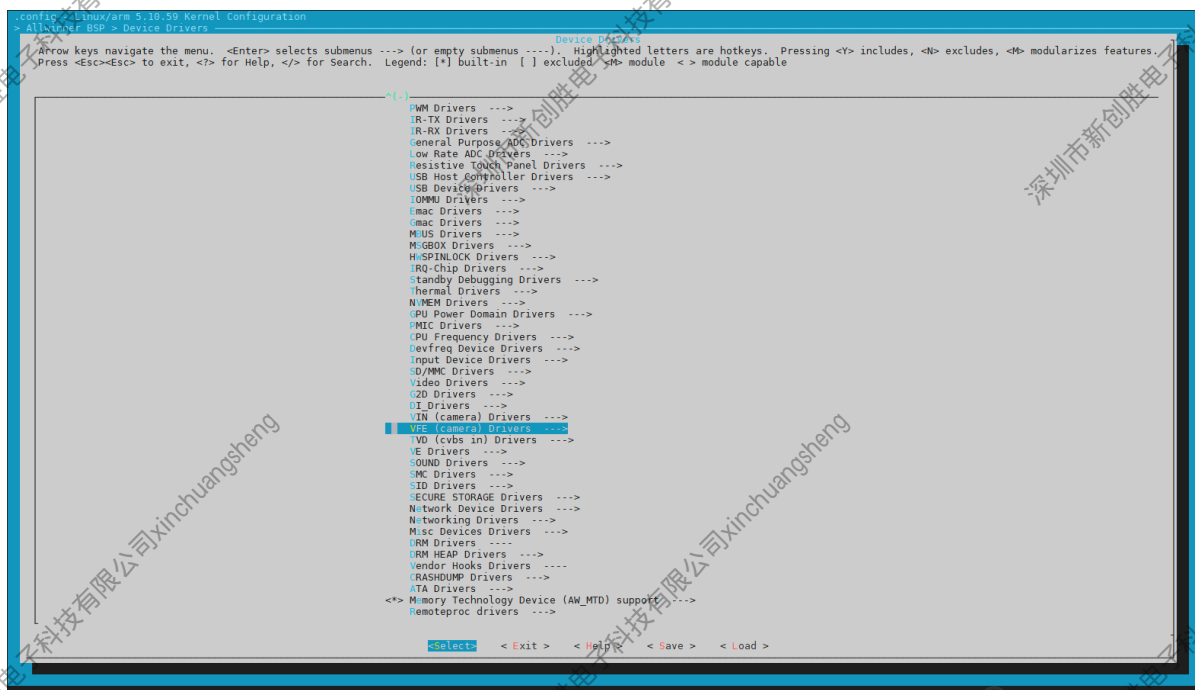


图 2-1: Device Drivers 选项配置

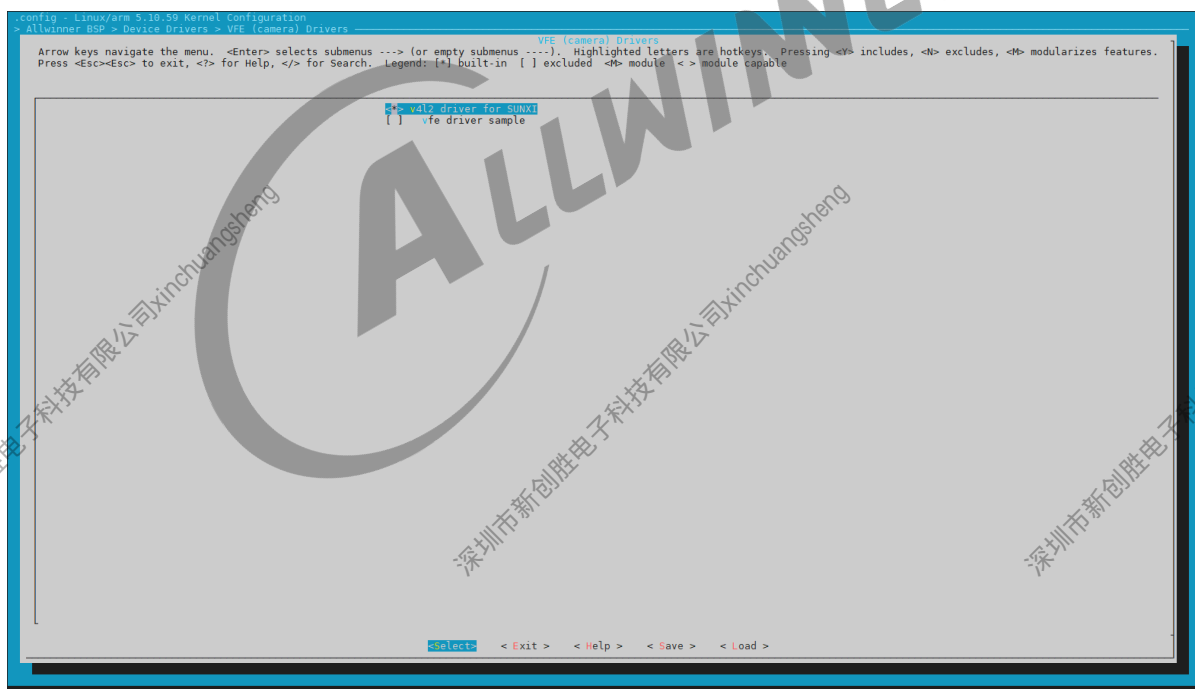


图 2-2: Device Drivers 选项配置

2.3.2 Device Tree 配置说明

- 设备树文件的配置是该 SoC 所有方案的通用配置，设备树的路径：
/bsp/config/{KERNEL_VERSION}/sun*.dtsi。

- 板级设备树 (board.dts) 路径：

/device/config/chips/{IC}/configs/{BOARD}/{KERNEL_VERSION}/board.dts。

在 dtsi 文件中，配置了该 SoC 的 CSI 控制器的通用配置信息，一般不建议修改，由 CSI 驱动维护者维护，如果需要修改配置请修改板级设备树 board.dts，板级设备树里面的内容会覆盖 dtsi 对应的信息。

```
&csi_cci0 {
    status = "disabled";
};
&csi_res0 {
    status = "okay";
};
&csi_res1 {
    status = "disabled";
};
&csi0 {
    device_type = "csi0";
    cci_sel      = <0>;
    csi_sel      = <0>;
    mipi_sel     = <0>;
    isp_sel      = <0>;
    csi0_sensor_list = <0>;
    csi0_mck     = <0>; /*PE1 .mul_sel = 1, .pull = 0, .drv_level = 1, .data = 0*/
    status = "okay";
    csi0_dev0:dev0{
        csi0_dev0_mname = "n5";
        csi0_dev0_twi_addr = <0x64>;
        csi0_dev0_twi_id = <3>;
        csi0_dev0_pos = "rear";
        csi0_dev0_isp_used = <0>;
        csi0_dev0_fmt = <0>;
        csi0_dev0_stby_mode = <0>;
        csi0_dev0_vflip = <0>;
        csi0_dev0_hflip = <0>;
        csi0_dev0_iovdd-supply = <&reg_eldo1>;
        csi0_dev0_iovdd_vol = <3300000>;
        csi0_dev0_avdd-supply = <0>;
        csi0_dev0_avdd_vol = <2800000>;
        csi0_dev0_dvdd-supply = <0>;
        csi0_dev0_dvdd_vol = <1500000>;
        csi0_dev0_afvdd-supply = <0>;
        csi0_dev0_afvdd_vol = <2800000>;
        csi0_dev0_power_en = <0>;
        csi0_dev0_reset = <0>; /*PC20 .mul_sel = 1, .pull = 0, .drv_level = 1, .data =
0*/
        csi0_dev0_pwn = <0>; /*PH16 .mul_sel = 1, .pull = 0, .drv_level = 1, .data =
0*/
        csi0_dev0_flash_en = <0>;
        csi0_dev0_flash_mode = <0>;
        csi0_dev0_af_pwn = <0>;
        csi0_dev0_act_used = <0>;
        csi0_dev0_act_name = "ad5820_act";
        csi0_dev0_act_slave = <0x18>;
        status = "okay";
    };
};
&csi1 {
```

```

device_type= "csil";
cci_sel      = <0>;
csi_sel      = <1>;
mipi_sel     = <0>;
isp_sel      = <0>;
csil_sensor_list = <0>;
csil_mck     = <>; /*PG1 .mul_sel = 1, .pull = 0, .drv_level = 1, .data = 0*/
status = "disabled";
csil_dev0:dev0{
    csil_dev0_mname      = "ov5640";
    csil_dev0_twi_addr   = <0x78>;
    csil_dev0_twi_id     = <1>;
    csil_dev0_pos        = "front";
    csil_dev0_isp_used   = <1>;
    csil_dev0_fmt        = <0>;
    csil_dev0_stby_mode  = <0>;
    csil_dev0_vflip      = <0>;
    csil_dev0_hflip      = <0>;
    csil_dev0_iovdd      = "iovdd-csi";
    csil_dev0_iovdd_vol  = <2800000>;
    csil_dev0_avdd       = "avdd-csi";
    csil_dev0_avdd_vol   = <2800000>;
    csil_dev0_dvdd       = "dvdd-csi-18";
    csil_dev0_dvdd_vol   = <1500000>;
    csil_dev0_afvdd      = "afvcc-csi";
    csil_dev0_afvdd_vol  = <2800000>;
    csil_dev0_power_en   = <>;
    csil_dev0_reset      = <>; /*PH14 .mul_sel = 1, .pull = 0, .drv_level = 1, .data =
0*/
    csil_dev0_pwn      = <>; /*PH17 .mul_sel = 1, .pull = 0, .drv_level = 1, .data =
0*/
    csil_dev0_flash_en  = <>;
    csil_dev0_flash_mode = <>;
    csil_dev0_af_pwn    = <>;
    csil_dev0_act_used   = <0>;
    csil_dev0_act_name   = "ad5820_act";
    csil_dev0_act_slave  = <0x18>;
    status = "disabled";
};

```

其中：

csi_cci0: cci 是 csi 内部专用 i2c 总线，若平台支持则可以打开使用，status = "okay" 代表打开，status = "disabled" 代表关闭。

csi_res0: 与 csi0 相关联，若使用 csi0，则需打开 csi_res0，包括一些时钟以及 gpio 的配置，用户无需配置其具体内容，只需负责开关即可。

csi0: csi0 的接口配置，包括 csi、mipi、isp 的选择，以及 sensor 配置。

cci_sel: cci 使用选择。

csi_sel: csi 使用选择。

mipi_sel: mipi 使用选择。

isp_sel: isp 使用选择。

csi0_sensor_list: 是否使用 sensor_list 功能，使用置 1，不使用置 0。

csi0_mck: mclk pin 脚配置。

csi0_dev0: 代表 sensor 设备配置。

csi0_dev0_mname: camera 的型号（与驱动对应）。

csi0_dev0_twi_addr: 对应 camera 的 I2C slave 地址（8bit 写地址）。

csi0_dev0_twi_id: 对应 camera 接到的 I2C 控制器 id。

csi0_dev0_pos: rear 代表后置，front 代表前置。

csi0_dev0_isp_used: 是否使用 isp。

csi0_dev0_fmt: camera 的数据格式（0 为 yuv; 1 为 bayer raw）。

csi0_dev0_stby_mode: 填 0。

csi0_dev0_vflip: sensor 输出的图像作上下颠倒。

csi0_dev0_hflip: sensor 输出的图像作左右镜像。

csi0_dev0_iovdd-supply: camera 的 IO 电源，参考原理图填写实际值。

csi0_dev0_iovdd_vol: 对应电源的电压，参考原理图填写实际值。

csi0_dev0_avdd-supply: camera 的模拟电源，参考原理图填写实际值。

csi0_dev0_avdd_vol: 对应电源的电压，参考原理图填写实际值。

csi0_dev0_dvdd-supply: camera 的数字电源，参考原理图填写实际值。

csi0_dev0_dvdd_vol: 对应电源的电压，参考原理图填写实际值。

csi0_dev0_afvdd-supply: camera 的对焦马达电源，参考原理图填写实际值。

csi0_dev0_afvdd_vol: 对应电源的电压，参考原理图填写实际值。

csi0_dev0_power_en: 外接 LDO 的控制 pin，根据实际填写。

csi0_dev0_reset: camera 模组的复位控制，根据实际填写。

csi0_dev0_pwdn: camera 模组的省电模式控制，根据实际填写。

csi0_dev0_flash_en: camera 模组的闪光灯 enable。

csi0_dev0_flash_mode: camera 模组的闪光灯 mode。

csi0_dev0_af_pwdn: camera 模组的闪光灯 mode。

csi0_dev0_act_used: 是否使用对焦马达。

csi0_dev0_act_name: 对焦马达名称，与对焦马达驱动对应。

csi0_dev0_act_slave: 对焦马达 slave 地址。

2.4 源码模块结构

驱动路径位于 longan/bsp/drivers/vfe 目录。

```
vfe:
├── actuator
│   ├── actuator.c
│   ├── actuator.h
│   ├── ad5820_act.c
│   ├── dw9714_act.c
│   ├── Makefile
│   └── ov8825_act.c
├── bsp_common.c
├── bsp_common.h
└── config.c
```

```
config.h
csi
| bsp_csi.c
| bsp_csi.h
| csi_reg.c
| csi_reg.h
| csi_reg_i.h
| csi_reg_v1.c
| csi_reg_v1.h
| sunxi_csi.c
| sunxi_csi.h
csi_cci
| bsp_cci.c
| bsp_cci.h
| cci_helper.c
| cci_helper.h
| cci_platform_drv.c
| cci_platform_drv.h
| csi_cci_reg.c
| csi_cci_reg.h
| csi_cci_reg_i.h
device
| camera_cfg.h
| camera.h
| gc0312.c
| gc0328c.c
| gc1034_mipi.c
| gc2145.c
| hm2131.c
| imx214.c
| imx219.c
| Makefile
| n5_dvp.c
| ov2710.c
| ov2710_mipi.c
| ov2775_mipi.c
| ov5640.c
| sensor_helper.c
| sensor_helper.h
flash_light
| flash.c
| flash.h
isp_cfg
| isp_cfg.c
| isp_cfg.h
| SENSOR_H
| gc1034_mipi_default_v3.h
| hm2131_default_f35.h
| ov2710_mipi_isp_cfg.h
| ov2775_mipi_default_v3.h
Kconfig
lib
| bsp_isp_algo.h
| bsp_isp_comm.h
| bsp_isp.h
| bsp_isp_null.c
| isp_module_cfg.h
| libisp_32
| libisp_64
| lib_mipicsi2_v1
```

```
└─ lib_mipicsi2_v2
└─ Makefile
└─ mipi_csi
    └─ bsp_mipi_csi.c
    └─ bsp_mipi_csi.h
    └─ bsp_mipi_csi_null.c
    └─ bsp_mipi_csi_v1.c
    └─ dphy
        └─ dphy.h
        └─ dphy_reg.c
        └─ dphy_reg.h
        └─ dphy_reg_i.h
    └─ protocol
        └─ protocol.h
        └─ protocol_reg.c
        └─ protocol_reg.h
        └─ protocol_reg_i.h
    └─ protocol.h
    └─ sunxi_mipi.c
    └─ sunxi_mipi.h
└─ platform
    └─ sun3iw1p1_vfe_cfg.h
    └─ sun50iw1p1_vfe_cfg.h
    └─ sun50iw2p1_vfe_cfg.h
    └─ sun8iw10p1_vfe_cfg.h
    └─ sun8iw11p1_vfe_cfg.h
    └─ sun8iw5p1_vfe_cfg.h
    └─ sun8iw6p1_vfe_cfg.h
    └─ sun8iw7p1_vfe_cfg.h
    └─ sun8iw8p1_vfe_cfg.h
└─ platform_cfg.h
└─ sample.c
└─ sunxi_isp.c
└─ sunxi_isp.h
└─ test
    └─ csi_test_0921
    └─ csi_test.c
    └─ Makefile
    └─ sunxi_camera.h
└─ utility
    └─ cfg_op.c
    └─ cfg_op.h
    └─ sensor_info.c
    └─ sensor_info.h
    └─ vfe_io.h
└─ vfe.c
└─ vfe.h
└─ vfe_os.c
└─ vfe_os.h
└─ vfe_subdev.c
└─ vfe_subdev.h
```

3 V4L2 接口描述

3.1 VIDIOC_QUERYCAP

3.1.1 Parameters

```
Capability of csi driver (struct v4l2_capability * capability)
struct v4l2_capability {
    __u8    driver[16]; /* i.e. "bttv" */
    __u8    card[32];  /* i.e. "Hauppauge WinTV" */
    __u8    bus_info[32]; /* "PCI:" + pci_name(pci_dev) */
    __u32    version;    /* should use KERNEL_VERSION() */
    __u32    capabilities; /* Device capabilities */
    __u32    reserved[4];
};
```

3.1.2 Returns

Success:0; Fail: Failure Number

3.1.3 Description

获取驱动的名称、版本、支持的 capabilities 等，如 V4L2_CAP_STREAMIN, V4L2_BUF_TYPE_VIDEO_CAPTURE 等。

3.2 VIDIOC_ENUM_INPUT

3.2.1 Parameters

```
input (struct v4l2_input *inp)
struct v4l2_input {
    __u32    index;    /* Which input */
    __u8     name[32]; /* Label */
    __u32    type;     /* Type of input */
    __u32    audioset; /* Associated audios (bitfield) */
    __u32    tuner;    /* Associated tuner */
    v4l2_std_id std;
};
```

```
_u32      status;  
_u32      capabilities;  
_u32      reserved[3];  
};
```

3.2.2 Returns

Success:0; Fail: Failure Number

3.2.3 Description

获取驱动支持的 input index。目前驱动只支持 input index = 0 或 index = 1。

Index = 0 表示 primary csi device

Index = 1 表示 secondary csi device

应用输入 index 参数，驱动返回 type。对于 VFE 设备来说，type 为 V4L2_INPUT_TYPE_CAMERA。

3.3 VIDIOC_S_INPUT

3.3.1 Parameters

```
input (struct v4l2_input *inp)  
The same as VIDIOC_ENUM_INPUT
```

3.3.2 Returns

Success:0; Fail: Failure Number

3.3.3 Description

通过 inp.index 设置当前要访问的 csi device 为 primary device 还是 secondary device。

Index = 0（双摄像头配置中，一般对应后置摄像头。若只有一个摄像头设备，则 index 固定为 0）

Index = 1（双摄像头配置中，一般对应前置摄像头）

调用该接口后，实际上会对 csi device 进行初始化工作。

在 A133 平台：Index 在 video0、1 时固定要设为 0；在 video2、3 要设为 1。

3.4 VIDIOC_G_INPUT

3.4.1 Parameters

```
input (struct v4l2_input *inp)  
The same as VIDIOC_ENUM_INPUT
```

3.4.2 Returns

Success:0; Fail: Failure Number

3.4.3 Description

获取 inp.index，判断当前设置的 csi device 为 primary device 还是 secondary device。

Index = 0（双摄像头配置中，一般对应后置摄像头。若只有一个摄像头设备，则 index 固定为 0）

Index = 1（双摄像头配置中，一般对应前置摄像头）

3.5 VIDIOC_S_PARM

3.5.1 Parameters

```
Parameter (struct v4l2_streamparm *parms)  
struct v4l2_streamparm {  
    enum v4l2_buf_type type;  
    union {  
        struct v4l2_captureparm capture;  
        struct v4l2_outputparm output;  
        __u8    raw_data[200]; /* user-defined */  
    } parm;  
};  
  
struct v4l2_captureparm {  
    __u32    capability; /* Supported modes */  
    __u32    capturemode; /* Current mode */  
    struct v4l2_fract timeperframe; /* Time per frame in .1us units */  
    __u32    extendedmode; /* Driver-specific extensions */  
    __u32    readbuffers; /* # of buffers for read */  
    __u32    reserved[4];  
};
```


3.5.2 Returns

Success:0; Fail: Failure Number

3.5.3 Description

CSI 作为输入设备，只关注 `parms.type` 和 `parms.capture`。

应用使用时，`parms.type = V4L2_BUF_TYPE_VIDEO_CAPTURE`；

其中通过设定 `parms->capture.capturemode` (`V4L2_MODE_VIDEO` 或 `V4L2_MODE_IMAGE`)，实现视频或图片的采集。通过设定 `parms->capture.timeperframe`，可以设置帧率。

3.6 VIDIOC_G_PARM

3.6.1 Parameters

Parameter (struct `v4l2_streamparm *parms`)
The same as `VIDIOC_S_PARM`

3.6.2 Returns

Success:0; Fail: Failure Number

3.6.3 Description

应用使用时，`parms.type = V4L2_BUF_TYPE_VIDEO_CAPTURE`；

通过 `parms->capture.capturemode` 返回当前是 `V4L2_MODE_VIDEO` 或 `V4L2_MODE_IMAGE`；

通过 `parms->capture.timeperframe`，返回当前设置的帧率。

3.7 VIDIOC_ENUM_FMT

3.7.1 Parameters

```
V4L2 format (struct v4l2_fmtdesc * fmtdesc)
struct v4l2_fmtdesc {
    __u32          index;           /* Format number      */
    enum v4l2_buf_type type;       /* buffer type       */
    __u32          flags;
    __u8           description[32]; /* Description string */
    __u32          pixelformat;    /* Format fourcc      */
    __u32          reserved[4];
};
```

3.7.2 Returns

Success:0; Fail: Failure Number

3.7.3 Description

获取驱动支持的 V4L2 格式。

应用输入 type, index 参数, 驱动返回 pixelformat。对于 VFE 设备来说, type 为 V4L2_BUF_TYPE_VIDEO_CAPTURE。

3.8 VIDIOC_TRY_FMT

3.8.1 Parameters

```
Video type, format and size (struct v4l2_format * fmt)
struct v4l2_format {
    enum v4l2_buf_type type;
    union {
        struct v4l2_pix_format    pix;
        struct v4l2_pix_format_mplane pix_mp;
        struct v4l2_window        win;
        struct v4l2_vbi_format     vbi;
        struct v4l2_sliced_vbi_format sliced;
        __u8 raw_data[200];
    } fmt;
};

struct v4l2_pix_format {
    __u32 width;
    __u32 height;
```

```
_u32          pixelformat;
enum v4l2_field field;
_u32          bytesperline; /* for padding, zero if unused */
_u32          sizeimage;
enum v4l2_colorspace colorspace;
_u32          priv;        /* private data, depends on pixelformat */
};
```

3.8.2 Returns

Success:0; Fail: Failure Number

3.8.3 Description

根据捕捉视频的类型、格式和大小，判断模式、格式等是否被驱动支持。不会改变任何硬件设置。

对于 VFE 设备，type 为 V4L2_BUF_TYPE_VIDEO_CAPTURE。使用 struct v4l2_pix_format 进行参数传递。

应用程序输入 struct v4l2_pix_format 结构体里面的 width、height、pixelformat、field 等参数，驱动返回最接近的 width、height；若 pixelformat、field 不支持，则默认选择驱动支持的第一种格式。

3.9 VIDIOC_S_FMT

3.9.1 Parameters

Video type, format and size (struct v4l2_format * fmt)
The same as VIDIOC_TRY_FMT

3.9.2 Returns

Success:0; Fail: Failure Number

3.9.3 Description

设置捕捉视频的类型、格式和大小，设置之前会调用 VIDIOC_TRY_FMT。

对于 VFE 设备，type 为 V4L2_BUF_TYPE_VIDEO_CAPTURE。使用 struct v4l2_pix_format 进行参数传递。

应用程序输入 width、height、pixelformat、field 等，驱动返回最接近的 width、height；若 pixelformat、field 不支持，则默认选择驱动支持的第一种格式。

应用程序应该以驱动返回的 width、height、pixelformat、field 等作为后续使用传递的参数。

对于 OSD 设备，type 为 V4L2_BUF_TYPE_VIDEO_OVERLAY。使用 struct v4l2_window 进行参数传递。

应用程序输入水印的个数、窗口位置和大小、bitmap 地址、bitmap 格式以及 global_alpha 等。驱动保存这些参数，并在 VIDIOC_OVERLAY 命令传递使能命令时生效。

3.10 VIDIOC_G_FMT

3.10.1 Parameters

Video type, format and size (struct v4l2_format * fmt)
The same as VIDIOC_TRY_FMT

3.10.2 Returns

Success:0; Fail: Failure Number

3.10.3 Description

获取捕捉视频的 width、height、pixelformat、field、bytesperline、sizeimage 等参数。

3.11 VIDIOC_OVERLAY

3.11.1 Parameters

Overlay on/off (unsigned int i)

3.11.2 Returns

Success:0; Fail: Failure Number

3.11.3 Description

传递 1 表示使能，0 表示关闭。设置使能时会更新 osd 参数，使之生效。

3.12 VIDIOC_REQBUFS

3.12.1 Parameters

```
Buffer type, count and memory map type (struct v4l2_requestbuffers * req)
struct v4l2_requestbuffers {
    __u32          count;
    enum v4l2_buf_type  type;
    enum v4l2_memory   memory;
    __u32          reserved[2];
};
```

3.12.2 Returns

Success:0; Fail: Failure Number

3.12.3 Description

v4l2_requestbuffers 结构中定义了缓存的数量，驱动会据此申请对应数量的视频缓存。多个缓存可以用于建立 FIFO，来提高视频采集的效率。这些 buffer 通过内核申请，申请后需要通过 mmap 方法，映射到 User 空间。

Count: 定义需要申请的 video buffer 数量；

Type: 对于 VFE 设备，为 V4L2_BUF_TYPE_VIDEO_CAPTURE；

Memory: 目前支持 V4L2_MEMORY_MMAP、V4L2_MEMORY_USERPTR、V4L2_MEMORY_DMABUF 方式。

应用程序传递上述三个参数，驱动会根据 VIDIOC_S_FMT 设置的格式计算供需要 buffer 的大小，并返回 count 数量。

3.13 VIDIOC_QUERYBUF

3.13.1 Parameters

```
Buffer type ,index and memory map type (struct v4l2_buffer *buf)
struct v4l2_buffer {
    __u32          index;
    enum v4l2_buf_type    type;
    __u32          bytesused;
    __u32          flags;
    enum v4l2_field    field;
    struct timeval    timestamp;
    struct v4l2_timecode    timecode;
    __u32          sequence;

    /* memory location */
    enum v4l2_memory    memory;
    union {
        __u32          offset;
        unsigned long    userptr;
        struct v4l2_plane *planes;
    } m;
    __u32          length;
    __u32          input;
    __u32          reserved;
};
```

3.13.2 Returns

Success:0; Fail: Failure Number

3.13.3 Description

通过 struct v4l2_buffer 结构体的 index，访问对应序号的 buffer，获取到对应 buffer 的缓存信息。主要利用 length 信息及 m.offset 信息来完成 mmap 操作。

3.14 VIDIOC_DQBUF

3.14.1 Parameters

```
Buffer type ,index and memory map type (struct v4l2_buffer *buf)
struct v4l2_buffer is the same as VIDIOC_QUERYBUF
```

3.14.2 Returns

Success:0; Fail: Failure Number

3.14.3 Description

将 driver 已经填充好数据的 buffer 出列，供应用使用。

应用程序根据 index 来识别 buffer，此时 m.offset 表示 buffer 对应的物理地址。

3.15 VIDIOC_QBUF

3.15.1 Parameters

Buffer type ,index and memory map type (struct v4l2_buffer *buf)

3.15.2 Returns

Success:0; Fail: Failure Number

3.15.3 Description

将 User 空间已经处理过的 buffer，重新入队，移交给 driver，等待填充数据。

应用程序根据 index 来识别 buffer。

3.16 VIDIOC_STREAMON

3.16.1 Parameters

Buffer type (enum v4l2_buf_type *type)

3.16.2 Returns

Success:0; Fail: Failure Number

3.16.3 Description

此处的 buffer type 为 V4L2_BUF_TYPE_VIDEO_CAPTURE。运行此 IOCTL，将 buffer 队列中所有 buffer 入队，并开启 CSIC DMA 硬件中断，每次中断便表示完成一帧 buffer 数据的填入。

3.17 VIDIOC_STREAMOFF

3.17.1 Parameters

Buffer type (enum v4l2_buf_type *type)

3.17.2 Returns

Success:0; Fail: Failure Number

3.17.3 Description

此处的 buffer type 为 V4L2_BUF_TYPE_VIDEO_CAPTURE。运行此 IOCTL，停止捕捉视频，将 frame buffer 队列清空，以及 video buffer 释放。

3.18 VIDIOC_QUERYCTRL

3.18.1 Parameters

```
Control id and value (struct v4l2_queryctrl *qc)
struct v4l2_queryctrl {
    __u32          id;
    enum v4l2_ctrl_type type;
    __u8           name[32]; /* Whatever */
    __s32          minimum; /* Note signedness */
    __s32          maximum;
    __s32          step;
    __s32          default_value;
    __u32          flags;
    __u32          reserved[2];
};
```


3.18.2 Returns

Success:0; Fail: Failure Number

3.18.3 Description

应用程序通过 id 参数，驱动返回需要调节参数的 name, minmum, maximum, default_value 以及步进 step。（由 v4l2 controls framework 完成）

目前可能支持的 id 请参考 VIDIOC_S_CTRL。

3.19 VIDIOC_S_CTRL

3.19.1 Parameters

Control id and value (struct v4l2_queryctrl *qc)
The same as VIDIOC_QUERYCTRL

3.19.2 Returns

Success:0; Fail: Failure Number

3.19.3 Description

应用程序通过 id, value 等参数，对 camera 驱动对应的参数进行设置。

驱动内部会先调用 vidioc_queryctrl，判断 id 是否支持，value 是否在 minimum 和 maximum 之间。（由 v4l2 controls framework 完成）

目前可能支持的 id 和 value 参考附件。

3.20 VIDIOC_G_CTRL

3.20.1 Parameters

Control id and value (struct v4l2_queryctrl *qc)
The same as VIDIOC_QUERYCTRL

3.20.2 Returns

Success:0; Fail: Failure Number

3.20.3 Description

应用程序通过 id，驱动返回对应 id 当前设置的 value。

3.21 VIDIOC_ENUM_FRAMESIZES

3.21.1 Parameters

```
index,type,format (struct v4l2_frmsizeenum)

enum v4l2_frmsizetypes {
    V4L2_FRMSIZE_TYPE_DISCRETE = 1,
    V4L2_FRMSIZE_TYPE_CONTINUOUS = 2,
    V4L2_FRMSIZE_TYPE_STEPWISE = 3,
};

struct v4l2_frmsize_discrete {
    __u32 width; /* Frame width [pixel] */
    __u32 height; /* Frame height [pixel] */
};

struct v4l2_frmsize_stepwise {
    __u32 min_width; /* Minimum frame width [pixel] */
    __u32 max_width; /* Maximum frame width [pixel] */
    __u32 step_width; /* Frame width step size [pixel] */
    __u32 min_height; /* Minimum frame height [pixel] */
    __u32 max_height; /* Maximum frame height [pixel] */
    __u32 step_height; /* Frame height step size [pixel] */
};

struct v4l2_frmsizeenum {
    __u32 index; /* Frame size number */
    __u32 pixel_format; /* Pixel format */
    __u32 type; /* Frame size type the device supports. */

    union { /* Frame size */
        struct v4l2_frmsize_discrete discrete;
        struct v4l2_frmsize_stepwise stepwise;
    };

    __u32 reserved[2]; /* Reserved space for future use */
};
```

3.21.2 Returns

Success:0; Fail: Failure Number

3.21.3 Description

根据应用传进来的 index, pixel_format, 驱动返回 type, 并根据 type 填写 discrete 或 stepwise 的值。Discrete 表示分辨率固定的值; stepwise 表示分辨率有最小值和最大值, 并根据 step 递增。上层根据返回的 type, 做对应不同的操作。

3.22 VIDIOC_ENUM_FRAMEINTERVALS

3.22.1 Parameters

```
Index, format, size, type (struct v4l2_fmvalenum)

enum v4l2_fmvaltypes {
    V4L2_FRMIVAL_TYPE_DISCRETE = 1,
    V4L2_FRMIVAL_TYPE_CONTINUOUS = 2,
    V4L2_FRMIVAL_TYPE_STEPWISE = 3,
};

struct v4l2_fmval_stepwise {
    struct v4l2_fract min; /* Minimum frame interval [s] */
    struct v4l2_fract max; /* Maximum frame interval [s] */
    struct v4l2_fract step; /* Frame interval step size [s] */
};

struct v4l2_fmvalenum {
    __u32 index; /* Frame format index */
    __u32 pixel_format; /* Pixel format */
    __u32 width; /* Frame width */
    __u32 height; /* Frame height */
    __u32 type; /* Frame interval type the device supports. */

    union { /* Frame interval */
        struct v4l2_fract discrete;
        struct v4l2_fmval_stepwise stepwise;
    };

    __u32 reserved[2]; /* Reserved space for future use */
};
```

3.22.2 Returns

Success:0; Fail: Failure Number

3.22.3 Description

应用程序通过 pixel_format、width、height、驱动返回 type，并根据 type 填写 V4L2_FRMIVAL_TYPE_DISCRETE、V4L2_FRMIVAL_TYPE_CONTINUOUS 或 V4L2_FRMIVAL_TYPE_STEPWISE。Discrete 表示支持单一的帧率；stepwise 表示支持步进的帧率。

4 模块使用范例

4.1 测试 demo

模块使用的 demo 的代码位于 `longan/bsp/drivers/vfe/test`，此目录下可以直接 `make` 生成 demo，把 demo 推到机器里面执行便可以获取指定 video 节点的图像。

推荐在 pc 上创建 bat 批处理文件，使用 adb 命令完成一系列抓图的动作，bat 内容参考如下，不同机器请注意修改 push 进去的路径：

```
adb root
adb remount
adb shell mount /dev/mmcblk0p1 /mnt
adb shell "mkdir /mnt/csi_test/"
adb shell rm /mnt/csi_test/*
adb push 路径\csi_test /mnt/csi_test/
adb shell chmod 777 /mnt/csi_test/csi_test
adb shell "cd /mnt/csi_test/ && ./csi_test 0 0 1920 1080 ./ 0 5"
adb shell sync
adb shell ls /mnt/csi_test/
adb pull /mnt/csi_test/
pause
```

最后会在 bat 指令的文件夹生成 csi_test 文件夹里面保存二进制的图像数据 *.bin 文件；可用 RawViewer 等软件查看图像数据。demo 参数说明：0 0 1920 1080 ./ 0 5 60 0，分别表示 video0，set_input index0，目标分辨率宽，目标分辨率高，bin 文件保存路径、图像格式（如 NV12，具体含义可以看 demo 代码的 s_fmt 参数）、采集帧数（帧数大于 10000 即为常开节点）。

4.2 调用流程

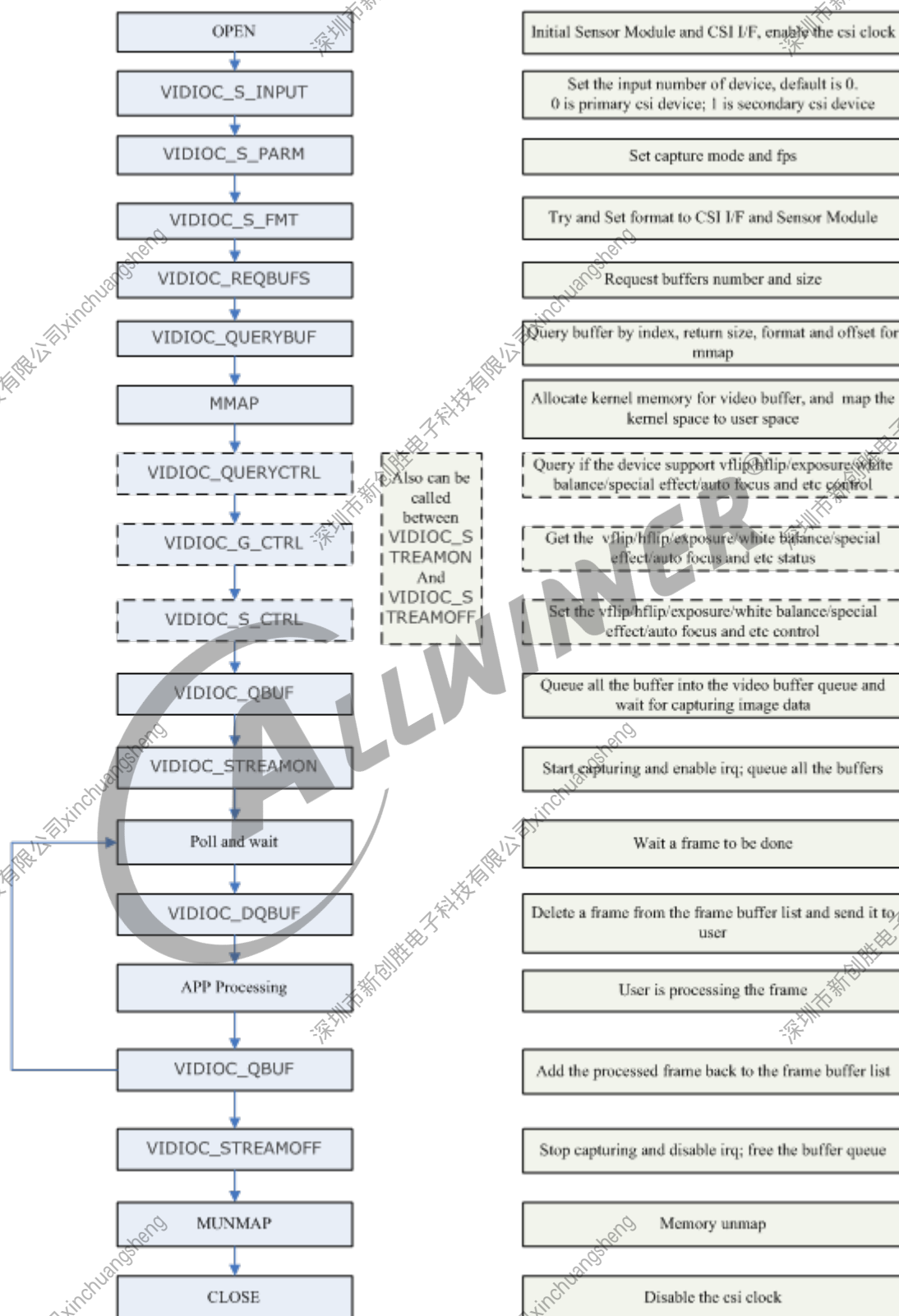


图 4-1: CSI 调用流程

5 FAQ

5.1 调试方法

5.1.1 中断状态

介绍如何观测 SOC 主控的接收数据的中断状态。

对于并口接口的 sensor，可以查看 user manual CSI 部分的 interrupt status 寄存器，观察 CSI 接收数据的中断状态，有无 vsync、frame_done 等中断。以此判断 CSI 是否有识别到 sensor 端发送的图像数据。

5.2 常见问题

5.2.1 I2C 不通

【分析步骤一】：确认供电、MCLK、i2c 上拉等外围电路信号是否正常。

使用万用表测量板上 AVDD、DVDD、IOVDD 供电电压、MCLK 频率、幅度、RESET、PWDN 的电平是否符合要求。

【分析步骤二】：确认 i2c 地址，TWI 通道是否和原理图一致。

【分析步骤三】：以上都正常就用示波器或者逻辑分析仪测量分析主控发出 i2c 波形是否正确、有无回应。

最后可以考虑 sensor 损坏或者接口错位等问题。

5.2.2 sensor 不出图

【分析步骤一】：确认 chip id 和 datasheet 上一致。

在对应 sensor 驱动的 sensor_detect 函数中读 chip id 寄存器，这一步也能检验 i2c 的读写是否正确。

【分析步骤二】：确认配置已经配置到 sensor 里。

可以把写进去的寄存器读出来和写入值对比是否一致。

【分析步骤三】：确认配置正确并且 sensor 已经输出图像。
和原厂确认寄存器配置、用示波器测量 sensor 端的数据和时钟波形，分析是否正在发送数据。

5.2.3 已出图但画面是绿色或者粉红色

一般是 YUYV 顺序反了，可以修改 sensor 驱动中 sensor_formats 结构体的 mbus_code 参数，修改 YUV 顺序即可。

5.2.4 I2c 已通，但是读所有 sensor 寄存器值都为 0

【分析步骤一】检查 i2c 通讯 addr 和 data 的位宽。
检查 sensor 驱动中 cci_drv 结构体中定义的值是否符合 datasheet 要求。

【分析步骤二】检查 i2c 通讯数据大小端是否不一致。
可以在读 sensor id 时把地址高低位相反来快速验证一下。

5.2.5 没有 video 节点

【问题解析】没有加载 ko 或者 ko 加载失败。

【分析步骤一】检查模块加载顺序是否正确。
lsmod 看一下模块是否加载正确，如果报的错误是 [VFE_ERR]registering gc2355_mipi, No such device, 则表明 sensor 模块 gc2355_mipi 没有加载。

【分析步骤二】检查 board.dts 文件配置是否配置了 csi 相关节点，且 status 为 okay。

【分析步骤三】如果是加载失败检查加载失败的原始是 i2c 不通还是没有 ko。




著作权声明

版权所有 © 2022 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。