AC54 拍照APP

AC54 拍照APP
- 拍照模式
  - 1）注册app
  - 2）状态机(state_machine)
    - APP_STATE_CREATE
    - APP_STATE_START
      - ACTION_PHOTO_TAKE_MAIN
      - ACTION_PHOTO_TAKE_SET_CONFIG
      - ACTION_PHOTO_TAKE_GET_CONFIG
      - ACTION_PHOTO_TAKE_CHANGE_STATUS
    - APP_STA_PAUSE
    - APP_STA_RESUME
    - APP_STA_STOP
    - APP_STA_DESTROY
  - 3）事件处理器（event_handler）
    - SYS_KEY_EVENT
    - SYS_DEVICE_EVENT

拍照模式

拍照模式是由 `video_photo.c` 和 `video_photo_cfg.c` 来实现app的处理，主要数据结构在 `video_photo.h` 中描述：

```
struct photo_menu_sta {
    u8 delay_mode;
    u8 quality;
    u8 repeat;
    u8 resolution;
    s8 ev;
    u8 iso;
    u8 hand_shake;
    u8 wb;
    u8 shpn_level;
    u8 date_label;
    u8 color;
    u8 quick_scan;
};
```

`struct photo_menu_sta` 描述了菜单设置的主要参数

```
struct photo_camera {
    u8 id;
    u8 state;
    u16 prev_width;
    u16 prev_height;
    u16 width;
    u16 height;
    struct server *server;
    struct photo_menu_sta menu_status;
};
```

`struct photo_camera` 描述了每个摄像头使用过程中的包含配置

```
struct video_photo_handle {
    u8 state;
    u8 camera_id;//
    int timeout;
    int delay_ms;
    u8 *cap_buf;
    u8 *zoom_buf;
    struct photo_camera camera[PHOTO_CAMERA_NUM];
    struct server *ui;
    struct server *display;
    struct server *video_dec;
    struct imc_osd_info label;
    char file_str[8];
};
```

`struct video_photo_handle` 描述app实际运行过程中的句柄参数，包括共有的**服务、buffer、标签**等。

1）注册app

整个photo app由以下代码注册到app列表

```
static const struct application_operation video_photo_ops = {
.state_machine  = state_machine,
.event_handler  = event_handler,
};

REGISTER_APPLICATION(app_video_photo) = {
.name   = "video_photo",
.action = ACTION_PHOTO_TAKE_MAIN,
.ops    = &video_photo_ops, //操作函数
.state  = APP_STA_DESTROY,
};
```

经过注册，将主要操作以 `state_machine` (状态机)和 `event_handler` (事件处理器)加入到app中。

2）状态机(state_machine)

`state_machine` 处理app调用时的**创建、运行、暂停、恢复、停止、注销**几个状态。

APP_STATE_CREATE

负责初始化拍照模式，app起始的服务加载、打开UI服务和app配置加载

APP_STATE_START

拍照模式下在运行app的动作处理都在该状态下，主要分为几个action动作：

```
#define ACTION_PHOTO_TAKE_MAIN          0x00008001
#define ACTION_PHOTO_TAKE_SET_CONFIG    0x00008002
#define ACTION_PHOTO_TAKE_GET_CONFIG    0x00008003
#define ACTION_PHOTO_TAKE_CHANGE_STATUS 0x00008004
```

ACTION_PHOTO_TAKE_MAIN

调用 video_photo_start 运行app的主功能：
初始化前后路相机（默认前路摄像头在线）
设置相机初始状态（默认后拉VGA分辨率，可调整）
打开相机的显示 photo_camera_display
打开UI show_main_ui

ACTION_PHOTO_TAKE_SET_CONFIG

一般由菜单发起，调用该动作设置app上的配置参数，
由 video_photo_set_config() 设置参数至app，再使用 sys_config_store 保存参数。

ACTION_PHOTO_TAKE_GET_CONFIG

获取app的配置参数，通过调用 video_photo_get_config() 实现

ACTION_PHOTO_TAKE_CHANGE_STATUS

该动作表示有菜单或其他状态的改变，例如：
摄像头切换
拍照后剩余数量重获取
菜单打开或关闭
调用 video_photo_change_status 来实现

APP_STA_PAUSE

暂时未处理该状态

APP_STA_RESUME

暂时未处理该状态

APP_STA_STOP

停止app运行，通过 video_photo_stop() 实现，
关闭相机（前后路）– camera_close()
关闭快速预览（如果有打开快速预览）– 关闭解码服务
关闭摄像头显示 – photo_camera_stop_display()
关闭延时拍照
关闭UI – hide_main_ui()

APP_STA_DESTROY

app销毁，主要关闭app的ui服务。

3）事件处理器（event_handler）

event_handler用来处理系统的设备事件和按键消息，以实现在拍照模式下的拍照、设备热插拔处理。

SYS_KEY_EVENT

按键消息在***KEY_EVENT_CLICK***事件类型下主要有：
KEY_PHOTO – 拍照
KEY_UP – 前后路切换

KEY_PHOTO :
拍照处理，正常拍照调用 `video_take_photo()` ，延时拍照调用 `video_delay_take_photo()`
KEY_UP :
摄像头切换，调用 `photo_switch_camera()`

SYS_DEVICE_EVENT

拍照模式下只需处理后视的热插拔，根据设备参数名称 `video1` 和插拔消息类型 `DEVICE_EVENT_IN` 、 `DEVICE_EVENT_OUT` 进行前后视的处理。