



NT96680 Sensor Driver Application Note

Table of Content

NT96680 Sensor Driver Application Note	1
Table of Content.....	2
1 概述.....	5
1.1 功能描述	5
1.2 內部流程	6
1.3 檔案架構	7
2 回調函數	8
2.1 註冊回調函數.....	8
2.2 電源控制函數.....	10
2.3 插入函數	13
2.4 驅動函數	14
3 I2C.....	16
3.1 初始化	16
3.2 移除.....	17
3.3 寫入.....	18
3.4 讀取.....	20
3.5 寫入同步 VD	22
4 設定值讀寫.....	25
4.1 寫入.....	25
4.2 讀取.....	27
4.3 使用者自定.....	31
5 設定文檔	32
6 AP 端的設定	34
6.1 線性模式	34
6.2 SHDR 模式	36
7 附錄.....	38
7.1 附錄：PQ 相關 KO 檔的載入.....	38
7.2 附錄：連結共用檔案.....	39
7.3 附錄：設定曝光及增益的彈性.....	40
7.4 設定參數	41
7.4.1 基本參數.....	41
7.4.2 信號參數.....	42

7.4.3	指令介面參數	43
7.4.4	幀率參數	44
7.4.5	時脈參數	45
7.4.6	模式選擇參數	46
7.4.7	模式基本參數	48
7.4.8	LVDS 參數	51
7.4.9	MIPI 參數	52
7.4.10	TGE 參數	54
7.5	附錄：除錯	56
7.5.1	列印設定	56
7.5.2	列印驅動訊息	57
7.5.3	列印 VD	57

修訂紀錄

版本	日期	作者	描述
1.00	2018/12/29	Photon Lin	初版
1.10	2019/1/8	Photon Lin	1. 第 6 章加入 SHDR 說明。 2. 附錄新增 PQ 相關 KO 檔的載入。 3. 版面優化及修正錯誤字。
1.20	2019/1/9	Photon Lin	1. 新增 7.4 章節,補充 4.2 章節的細節說明。 2. 版面優化。
1.30	2019/6/6	Photon Lin	1. 第 5 章節,加入 sensor 翻轉說明。 2. 第 6.2 章節,修改 SHDR 說明。

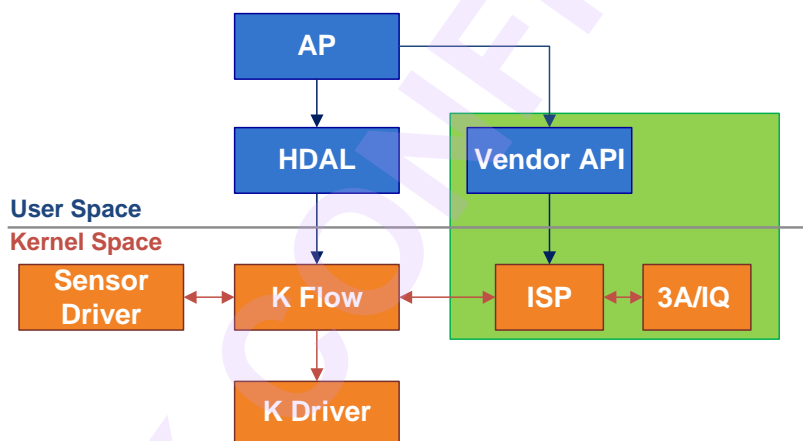
1 概述

1.1 功能描述

Sensor Driver，為在核心層內所有 Sensor 的驅動代碼。各個 Sensor 使用獨立的 ko 檔案，其代碼為不同的分辨率、幀率、傳輸介面設定值、流程、及相關應用的總成集合。

如圖表 1 所示，Sensor 驅動在初始化時會對 K Flow 註冊回調函數，供 HDAL 及 ISP 模塊使用。詳細說明參閱第 2 章節。

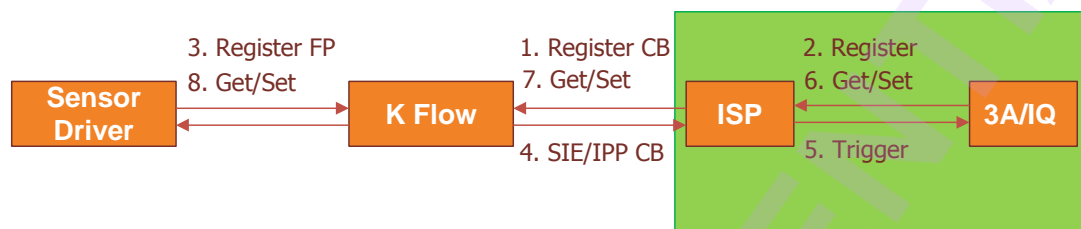
硬件相關及 pinmux 的部份，則為 AP 經由 HDAL，設定到 K flow，詳細說明參閱第 6 章節。



圖表 1：ISP 功能圖

1.2 內部流程

圖表 2 描述 ISP/K Flow/Sensor 驅動模塊間的初始化及運作流程。



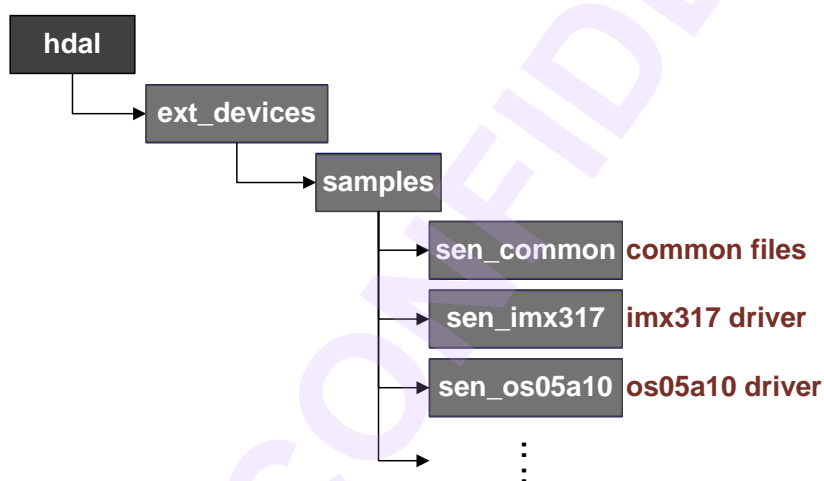
圖表 2：Sensor/K flow/ISP 流程

- (1) ISP 模塊載入時，對 CTL_SIE/CTL_IPP/Video Enc 註冊回調時間點及回調函數。
- (2) NVT 算法模塊載入時，對 ISP 註冊回調函數。
- (3) Sensor 模塊載入時，對 K flow 註冊回調函數。
- (4) Sensor 啟動後，K flow 中的 CTL_SIE 及 CTL_IPP 模塊會在對 ISP 回調。
- (5) 承上，ISP 模塊收到回調後，會對有註冊的算法模塊調用觸發函數。
- (6) 算法及 IQ 模塊之間的參數交換，透過 ISP 模塊處理。
- (7) 上層接口、算法模塊、及 IQ 模塊，對 K flow 調用讀寫函數皆透過 ISP 模塊處理。
- (8) K Flow 對 Sensor 驅動調用讀取/寫入函數。此部份包含對 Sensor 暫存器的讀寫，及訊息的讀取。

1.3 檔案架構

如圖表 3 所示：

- **sen_common**：設定文檔的解析、I2C 驅動控制，及一些共用的代碼、宣告。[此目錄的檔案需使用連結的方式引用。參閱 7.2。](#)
- **sen_imx317**：imx317 的設定及驅動代碼。
- **sen_os05a10**：os05a10 的設定及驅動代碼。



圖表 3：檔案架構

2 回調函數

2.1 註冊回調函數

在載入 Sensor 驅動的 ko 檔時，會對 K flow 註冊回調函數。最多可掛載 8 個不同名稱的回調函數，供上層應用。若無註冊，則系統無法啟用 Sensor。

[定義]

ctl_sen.h

```
typedef struct {
    CTL_SEN_PWR_CTRL pwr_ctrl;
    CTL_SEN_PLUG_IN det_plug_in;
    CTL_SEN_DRV_TAB *drv_tab;
} CTL_SEN_REG_OBJ, *PCTL_SEN_REG_OBJ;

ER ctl_sen_reg_sendrv(CHAR *name, CTL_SEN_REG_OBJ *reg_obj);
ER ctl_sen_unreg_sendrv(CHAR *name);
```

[成員]

CTL_SEN_REG_OBJ 結構

成員名稱	描述
pwr_ctrl	電源控制函數指針。參閱 2.2。
det_plug_in	插入函數指針。參閱 2.3。
drv_tab	驅動函數指針表。參閱 2.4。

[舉例]

sen_os05a10.c

```
#include "kflow_videocapture/ctl_sen.h"

static int __init sen_init_os05a10(void)
{
    CTL_SEN_REG_OBJ reg_obj;
```



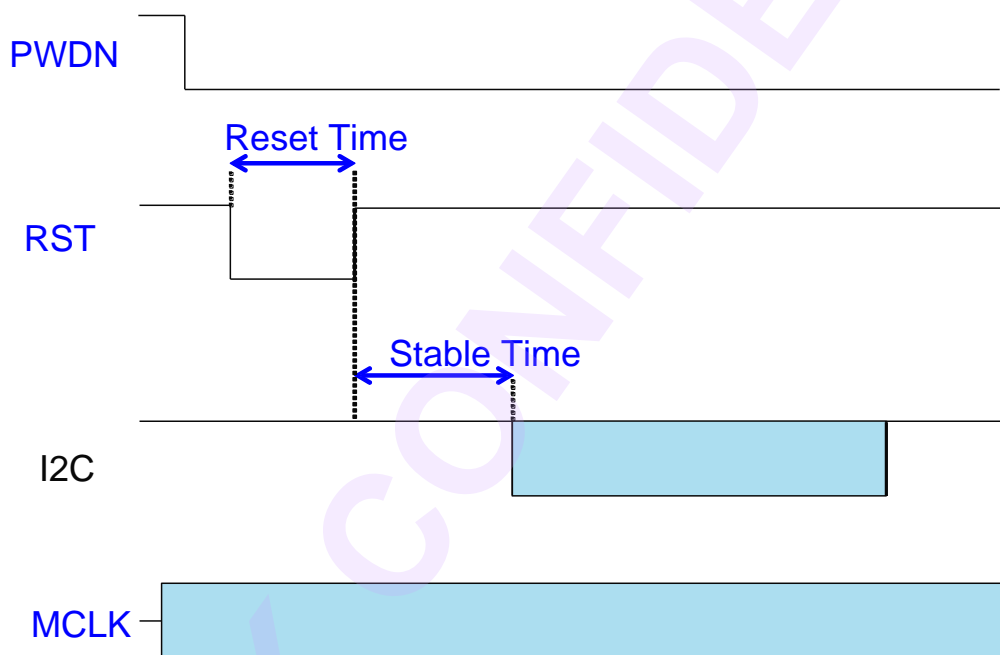
```
ER rt = E_OK;

memset((void *)&reg_obj, 0, sizeof(CTL_SEN_REG_OBJ));
reg_obj.pwr_ctrl = sen_pwr_ctrl_os05a10;
reg_obj.det_plug_in = NULL;
reg_obj.drv_tab = sen_get_drv_tab_os05a10();
rt = ctl_sen_reg_sendrv("nvt_sen_os05a10", &reg_obj);
if (rt != E_OK) {
    DBG_ERR("register sensor driver fail \r\n");
}
}
```

2.2 電源控制函數

電源控制函數在 Sensor 上電及斷電時由 K flow 調用。如下圖所示，通常包含以下幾個部份：

- MCLK 時脈控制
- Reset 流程控制
- I2C 可讀寫點控制



圖表 4：Sensor 電源啟動流程

[注意]

SEN_POWER 結構裡的成員，載入 ko 檔案時指定設定文檔，並修改其設定值。參閱第 5 章節。

[定義]

ctl_sen.h

```
typedef void (*CTL_SEN_PWR_CTRL)(CTL_SEN_ID id, CTL_SEN_PWR_CTRL_FLAG flag, CTL_SEN_CLK_CB
clk_cb);
```

sen_common.h

```
typedef struct _SEN_POWER {
    UINT32 mclk;
    UINT32 pwn_pin;
    UINT32 rst_pin;
    UINT32 rst_time;
    UINT32 stable_time;
} SEN_POWER;
```

[成員]

CTL_SEN_PWR_CTRL 函數指針

成員名稱	描述
id	Sensor 通道號。
flag	動作指示。 CTL_SEN_PWR_CTRL_TURN_ON 為電源開啟流程控制。 CTL_SEN_PWR_CTRL_TURN_OFF 為電源關閉流程控制。
clk_cb	MCLK 時脈回調函數指針。

SEN_POWER 結構

成員名稱	描述
mclk	MCLK 時脈來源選擇。可根據硬件設計選擇 CTL_SEN_CLK_SEL_SIEMCLK 或 CTL_SEN_CLK_SEL_SIEMCLK2。
pwn_pin	PWDN PIN 的 GPIO 接腳， 若無使用則填入 CTL_SEN_IGNORE。
rst_pin	RESET PIN 的 GPIO 接腳， 若無使用則填入 CTL_SEN_IGNORE。
rst_time	RESET 時間。單位為毫秒(ms)。
stable_time	RESET 結束到 I2C 可讀寫的時間。單位為毫秒(ms)。

[舉例]

sen_os05a10.c

```
#include "sen_common.h"

static SEN_POWER sen_power[CTL_SEN_ID_MAX] = {
    {CTL_SEN_CLK_SEL_SIEMCLK, CTL_SEN_IGNORE, 4, 1, 1},
    {CTL_SEN_CLK_SEL_SIEMCLK, CTL_SEN_IGNORE, 4, 1, 1},
```

```

{CTL_SEN_CLK_SEL_SIEMCLK, CTL_SEN_IGNORE, 4, 1, 1},
{CTL_SEN_CLK_SEL_SIEMCLK, CTL_SEN_IGNORE, 4, 1, 1},
{CTL_SEN_CLK_SEL_SIEMCLK, CTL_SEN_IGNORE, 4, 1, 1},
{CTL_SEN_CLK_SEL_SIEMCLK, CTL_SEN_IGNORE, 4, 1, 1},
{CTL_SEN_CLK_SEL_SIEMCLK, CTL_SEN_IGNORE, 4, 1, 1},
{CTL_SEN_CLK_SEL_SIEMCLK, CTL_SEN_IGNORE, 4, 1, 1}
};

static void sen_pwr_ctrl_os05a10(CTL_SEN_ID id, CTL_SEN_PWR_CTRL_FLAG flag, CTL_SEN_CLK_CB clk_cb)
{
    DBG_IND("enter flag %d \r\n", flag);

    if (flag == CTL_SEN_PWR_CTRL_TURN_ON) {

        if (clk_cb != NULL) {
            clk_cb(sen_power[id].mclk, TRUE);
        }

        if (sen_power[id].rst_pin != CTL_SEN_IGNORE) {
            gpio_direction_output(S_GPIO(sen_power[id].rst_pin), 0);
            gpio_set_value(S_GPIO(sen_power[id].rst_pin), 0);
            sen_common_delay_ms(sen_power[id].rst_time);
            gpio_set_value(S_GPIO(sen_power[id].rst_pin), 1);
            sen_common_delay_ms(sen_power[id].stable_time);
        }
    }

    if (flag == CTL_SEN_PWR_CTRL_TURN_OFF) {
        if (clk_cb != NULL) {
            clk_cb(sen_power[id].mclk, FALSE);
        }
    }
}

```

2.3 插入函數

在使用 VX1、AHD、及 TVI 傳輸介面時，需指定插入函數以偵測裝置的插入狀態。反之則設定為 NULL 即可。

[定義]

ctl_sen.h

```
typedef BOOL (*CTL_SEN_PLUG_IN)(CTL_SEN_ID id);
```

[成員]

CTL_SEN_PLUG_IN 函數指針

成員名稱	描述
id	Sensor 通道號。

2.4 驅動函數

K flow 經由調用驅動函數指針，完成所有的流程控制、讀寫設定、及讀寫暫存。點亮時，調用的順序為 open→chgmode→close。

【注意】

驅動函數指針表裡的成員，不同的 Sensor 可能會有些微差異，可直接參考各 Sensor 驅動代碼。

【定義】

ctl_sen.h

```
typedef struct {
    ER(*open)(CTL_SEN_ID id);
    ER(*close)(CTL_SEN_ID id);
    ER(*sleep)(CTL_SEN_ID id);
    ER(*wakeup)(CTL_SEN_ID id);
    ER(*write)(CTL_SEN_ID id, CTL_SEN_CMD *cmd);
    ER(*read)(CTL_SEN_ID id, CTL_SEN_CMD *cmd);
    ER(*chgmode)(CTL_SEN_ID id, CTL_SENDRV_CHGMODE_OBJ chgmode_obj);
    ER(*chgfps)(CTL_SEN_ID id, UINT32 frame_rate);
    ER(*set_cfg)(CTL_SEN_ID id, CTL_SENDRV_CFGID drv_cfg_id, void *data);
    ER(*get_cfg)(CTL_SEN_ID id, CTL_SENDRV_CFGID drv_cfg_id, void *data);
} CTL_SEN_DRV_TAB;
```

【成員】

CTL_SEN_DRV_TAB 結構

成員名稱	描述
open	開啟時調用。包含 I2C 的初始化，參閱 3.1。
close	關閉時調用。包含 I2C 的移除，參閱 3.2。
sleep	睡眠時調用。
wakeup	喚起時調用。
write	寫入 Sensor 暫存器時調用。參閱 3.3。
read	讀取 Sensor 暫存器時調用。參閱 3.4。

chgmode	切換 Sensor 操作模式時調用，包含 Sensor 的初始化設定
chgfps	切換幀率時調用
set_cfg	寫入設定時調用，參閱 4.1
get_cfg	讀取設定時調用，參閱 4.2

[舉例]

sen_os05a10.c

```
#include "kflow_videocapture/ctl_sen.h"

static CTL_SEN_DRV_TAB os05a10_sen_drv_tab = {
    sen_open_os05a10,
    sen_close_os05a10,
    sen_sleep_os05a10,
    sen_wakeup_os05a10,
    sen_write_reg_os05a10,
    sen_read_reg_os05a10,
    sen_chg_mode_os05a10,
    sen_chg_fps_os05a10,
    sen_set_info_os05a10,
    sen_get_info_os05a10,
};

static CTL_SEN_DRV_TAB *sen_get_drv_tab_os05a10(void)
{
    return &os05a10_sen_drv_tab;
}
```

3 I2C

3.1 初始化

[語法]

sen_i2c.c

```
ER sen_i2c_init_driver(UINT32 id, SEN_I2C_ID i2c_id)
```

[成員]

成員名稱	描述
id	Sensor 通道號。
i2c_id	I2C 編號。

[舉例]

sen_os05a10.c

```
#include "sen_i2c.c"

static ER sen_open_os05a10(CTL_SEN_ID id)
{
    ER rt = E_OK;

    rt = sen_i2c_init_driver(id, sen_i2c[id].id);
    if (rt != E_OK) {
        DBG_ERR("init. i2c driver fail (%d) \r\n", id);
    }
    return rt;
}
```


3.2 移除

[語法]

sen_i2c.c

```
void sen_i2c_remove_driver(UINT32 id)
```

[成員]

成員名稱	描述
id	Sensor 通道號。

[舉例]

sen_os05a10.c

```
#include "sen_i2c.c"

static ER sen_close_os05a10(CTL_SEN_ID id)
{
    sen_i2c_remove_driver(id);
    return E_OK;
}
```

3.3 寫入

I2C 的讀寫函數，需傳入 Sensor 通道號、msgs 結構、及 num。

【語法】

sen_i2c.c

```
INT32 sen_i2c_transfer(UINT32 id, struct i2c_msg *msgs, INT32 num)
```

【成員】

成員名稱	描述
id	Sensor 通道號。
msgs	Linux I2C framework 所需結構。
num	Linux I2C framework 所需參數。

【舉例】

sen_os05a10.c

```
#include "sen_i2c.c"

static ER sen_write_reg_os05a10(CTL_SEN_ID id, CTL_SEN_CMD *cmd)
{
    struct i2c_msg msgs;
    unsigned char buf[3];

    buf[0]    = (cmd->addr >> 8) & 0xFF;
    buf[1]    = cmd->addr & 0xFF;
    buf[2]    = cmd->data[0] & 0xFF;
    msgs.addr = sen_i2c[id].addr;
    msgs.flags = 0;
    msgs.len  = 3;
    msgs.buf  = buf;

    sen_i2c_transfer(id, &msgs, 1)

    return E_OK;
```

}

【說明】

舉例使用 2B1B 格式，即位址使用 2Byte、資料使用 1Byte。

3.4 讀取

[語法]

sen_i2c.c

```
INT32 sen_i2c_transfer(UINT32 id, struct i2c_msg *msgs, INT32 num)
```

[成員]

成員名稱	描述
id	Sensor 通道號。
msgs	Linux I2C framework 所需結構。
num	Linux I2C framework 所需參數。

[舉例]

sen_os05a10.c

```
static ER sen_read_reg_os05a10(CTL_SEN_ID id, CTL_SEN_CMD *cmd)
{
    struct i2c_msg msgs[2];
    unsigned char tmp[2], tmp2[2];

    tmp[0] = (cmd->addr >> 8) & 0xFF;
    tmp[1] = cmd->addr & 0xFF;
    msgs[0].addr = sen_i2c[id].addr;
    msgs[0].flags = 0;
    msgs[0].len = 2;
    msgs[0].buf = tmp;

    tmp2[0] = 0;
    msgs[1].addr = sen_i2c[id].addr;
    msgs[1].flags = 1;
    msgs[1].len = 1;
    msgs[1].buf = tmp2;

    sen_i2c_transfer(id, msgs, 2);
}
```

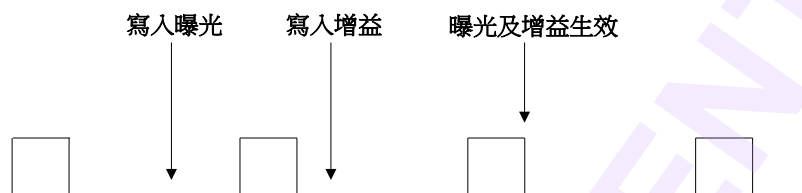
```
cmd->data[0] = tmp2[0];  
  
return E_OK;  
}
```

【說明】

舉例使用 2B1B 格式，即位址使用 2Byte、資料使用 1Byte。

3.5 寫入同步 VD

如圖表 5 所示，當 Sensor 的曝光及增益寫入點相差一幀時，可使用同步 VD 功能。

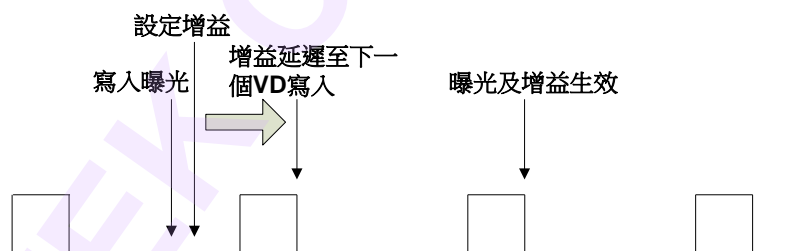


圖表 5：曝光及增益寫入時間點差一幀

I2C 驅動會偵測 SIE 的 VD 訊號，並在下一個 VD 起始點將設定寫入，如圖表 6 所示。此功能注意點如下：

- 以群為單位寫入設定
- 每次寫入前，需要先設定 SIE 通道，以獲取 VD 訊息
- 只能延遲至一個 VD。
- 設定後到下個 VD 前，該 I2C 編號的通道會被驅動鎖住。

故需先用一般 I2C 寫入功能寫入曝光設定，然後再使用同步 VD 功能寫入增益設定。



圖表 6：增益延遲一幀寫入

【語法】

sen_i2c.c

```
INT32 sen_i2c_transfer(UINT32 id, struct i2c_msg *msgs, INT32 num)
```

【成員】

成員名稱	描述
id	Sensor 通道號。

msgs	Linux I2C framework 所需結構。
num	Linux I2C framework 所需參數。

[舉例]

sen_imx317.c

```
#include <linux/i2c.h>
#include "sen_i2c.c"

#define EXPT_REG_NUM          3
#define NVT_I2C_VD_SEND      0x1000
#define NVT_I2C_VD_SRC       0x2000

static ER sen_write_reg_vd_imx317(CTL_SEN_ID id, CTL_SEN_CMD *cmd)
{
    struct i2c_msg msgs[EXPT_REG_NUM];
    unsigned char buf[3*EXPT_REG_NUM];
    UINT32 i;
    ER rt = E_OK;

    // Set sync with which SIE source
    buf[0]    = id+1; // SIE source
    msgs[0].addr = sen_i2c[id].addr;
    msgs[0].flags = NVT_I2C_VD_SRC;
    msgs[0].len  = 1;
    msgs[0].buf  = buf;
    rt = sen_i2c_transfer(id, msgs, 1);

    // Send a set of write command to wait VD sync
    for (i = 0; i < EXPT_REG_NUM; i++) {
        buf[i*3+0] = (cmd[i].addr >> 8) & 0xFF;
        buf[i*3+1] = cmd[i].addr & 0xFF;
        buf[i*3+2] = cmd[i].data[0] & 0xFF;

        msgs[i].addr = sen_i2c[id].addr;
        msgs[i].flags = NVT_I2C_VD_SEND;
        msgs[i].len  = 3;
    }
}
```

```
    msgs[i].buf  = &buf[i*3];  
}  
rt |= sen_i2c_transfer(id, msgs, EXPT_REG_NUM);  
  
return rt;  
}
```

[說明]

- 舉例使用 2B1B 格式，即位址使用 2Byte、資料使用 1Byte。
- `#define EXPT_REG_NUM 3`，代表一次寫入 3 筆設定。
- `// Set sync with which SIE source`，此註解代表設定 SIE 通道，

4 設定值讀寫

4.1 寫入

設定的寫入包含跟 AE 算法相關的曝光/增益，及應用相關的畫面翻轉。

[注意]

不同的 Sensor 可能會有些微差異，可直接參考各 Sensor 驅動代碼。

[定義]

ctl_sen.h

```
typedef enum {
    /* set only*/
    CTL_SENDRV_CFGID_SET_BASE = 0x00000000,
    CTL_SENDRV_CFGID_SET_EXPT,
    CTL_SENDRV_CFGID_SET_GAIN,
    CTL_SENDRV_CFGID_SET_FPS,
    :
    :
    /* set & get */
    CTL_SENDRV_CFGID_BASE = 0x02000000,
    CTL_SENDRV_CFGID_FLIP_TYPE,
    :
    :
} CTL_SENDRV_CFGID;
```

[成員]

CTL_SENDRV_CFGID 列舉

成員名稱	描述
CTL_SENDRV_CFGID_SET_EXPT	寫入曝光。單位為微秒(us)。
CTL_SENDRV_CFGID_SET_GAIN	寫入增益。1000 為 1 倍增益。
CTL_SENDRV_CFGID_SET_FPS	寫入幀率。精度為 100。

CTL_SENDRV_CFGID_FLIP_TYPE

寫入影像水平及垂直翻轉。

[舉例]**sen_os05a10.c**

```
#include "kflow_videocapture/ctl_sen.h"

static ER sen_set_info_os05a10(CTL_SEN_ID id, CTL_SENDRV_CFGID drv_cfg_id, void *data)
{
    switch (drv_cfg_id) {
        case CTL_SENDRV_CFGID_SET_EXPT:
            sen_set_expt_os05a10(id, data);
            break;
        case CTL_SENDRV_CFGID_SET_GAIN:
            sen_set_gain_os05a10(id, data);
            break;
        case CTL_SENDRV_CFGID_FLIP_TYPE:
            sen_set_flip_os05a10(id, (CTL_SEN_FLIP *) (data));
        default:
            return E_NOSPT;
    }
    return E_OK;
}
```

4.2 讀取

設定的讀取主要提供 K flow 要點亮 Sensor 時的必要訊息、AE 算法的訊息參考、及上層應用的訊息參考。

[注意]

不同的 Sensor 可能會有些微差異，可直接參考各 Sensor 驅動代碼。

[定義]

ctl_sen.h

```
typedef enum {  
    :  
    :  
    /* get only */  
    CTL_SENDRV_CFGID_GET_BASE = 0x01000000,  
    CTL_SENDRV_CFGID_GET_EXPT,  
    CTL_SENDRV_CFGID_GET_GAIN,  
    CTL_SENDRV_CFGID_GET_ATTR_BASIC,  
    CTL_SENDRV_CFGID_GET_ATTR_SIGNAL,  
    CTL_SENDRV_CFGID_GET_ATTR_CMDIF,  
    CTL_SENDRV_CFGID_GET_ATTR_IF,  
    CTL_SENDRV_CFGID_GET_TEMP,  
    CTL_SENDRV_CFGID_GET_FPS,  
    CTL_SENDRV_CFGID_GET_SPEED,  
    CTL_SENDRV_CFGID_GET_MODESEL,  
    CTL_SENDRV_CFGID_GET_MODE_BASIC,  
    CTL_SENDRV_CFGID_GET_MODE_LVDS,  
    CTL_SENDRV_CFGID_GET_MODE_MIPI,  
    CTL_SENDRV_CFGID_GET_MODE_PARA,  
    CTL_SENDRV_CFGID_GET_MODE_SLVSEC,  
    CTL_SENDRV_CFGID_GET_MODE_DVI,  
    CTL_SENDRV_CFGID_GET_MODE_TGE,  
    CTL_SENDRV_CFGID_GET_PLUG_INFO,  
    CTL_SENDRV_CFGID_GET_PROBE_SEN,  
}
```

```

CTL_SENDRV_CFGID_GET_SENDRV_VER,

/* set & get */
CTL_SENDRV_CFGID_BASE = 0x02000000,
CTL_SENDRV_CFGID_FLIP_TYPE,
:
:
} CTL_SENDRV_CFGID;

```

[成員]

CTL_SENDRV_CFGID 列舉

成員名稱	描述
CTL_SENDRV_CFGID_GET_EXPT	讀取曝光。單位為微秒(us)。
CTL_SENDRV_CFGID_GET_GAIN	讀取增益。1000 為 1 倍增益。
CTL_SENDRV_CFGID_GET_ATTR_BASIC	讀取基本訊息。參閱 7.4.1。
CTL_SENDRV_CFGID_GET_ATTR_SIGNAL	讀取信號訊息。參閱 7.4.2。
CTL_SENDRV_CFGID_GET_ATTR_CMDIF	讀取控制訊息。參閱 7.4.3。
CTL_SENDRV_CFGID_GET_ATTR_IF	讀取介面訊息。此訊息在使用 LVDS 格式，且廠商名稱為 CTL_SEN_VENDOR_OTHERS(參閱 7.4.1) 時需設定。
CTL_SENDRV_CFGID_GET_TEMP	讀取溫度訊息。支援溫度讀取功能時使用。
CTL_SENDRV_CFGID_GET_FPS	讀取幀率訊息。參閱 7.4.4。
CTL_SENDRV_CFGID_GET_SPEED	讀取時脈訊息。參閱 7.4.5。
CTL_SENDRV_CFGID_GET_MODESEL	讀取操作模式編號。參閱 7.4.6。
CTL_SENDRV_CFGID_GET_MODE_BASIC	讀取操作模式訊息。參閱 7.4.7。
CTL_SENDRV_CFGID_GET_MODE_LVDS	讀取操作模式的 LVDS 訊息。參閱 7.4.8。
CTL_SENDRV_CFGID_GET_MODE_MIPI	讀取操作模式的 MIPI 訊息。參閱 7.4.9。
CTL_SENDRV_CFGID_GET_MODE_PARA	讀取操作模式參數。此訊息只在並口格式時使用。
CTL_SENDRV_CFGID_GET_MODE_SLVSEC	讀取操作模式的 SLVSEC 訊息。在 SLVS-EC 格式時使用。
CTL_SENDRV_CFGID_GET_MODE_DVI	讀取操作模式的 DVI 訊息。在並口格式時使用。
CTL_SENDRV_CFGID_GET_MODE_TGE	讀取操作模式 TGE 訊息。此訊息只在 Sensor 為 Slave 模式時使用，參閱 7.4.10。
CTL_SENDRV_CFGID_GET_PLUG_INFO	提供系統外部裝置訊息。系統會根據此訊息做相對應的流程處理。

CTL_SENDRV_CFGID_GET_PROBE_SEN	讀取 Probe 訊息。
CTL_SENDRV_CFGID_GET_SENDRV_VER	讀取版本號。
CTL_SENDRV_CFGID_FLIP_TYPE	讀取影像水平及垂直翻轉。

[舉例]

sen_os05a10.c

```
#include "kflow_videocapture/ctl_sen.h"

static ER sen_get_info_os05a10(CTL_SEN_ID id, CTL_SENDRV_CFGID drv_cfg_id, void *data)
{
    ER rt = E_OK;

    switch (drv_cfg_id) {
        case CTL_SENDRV_CFGID_GET_EXPT:
            sen_get_expt_os05a10(id, data);
            break;
        case CTL_SENDRV_CFGID_GET_GAIN:
            rt = E_NOSPT;
            break;
        case CTL_SENDRV_CFGID_GET_ATTR_BASIC:
            sen_get_attr_basic_os05a10((CTL_SENDRV_GET_ATTR_BASIC_PARAM *)(data));
            break;
        case CTL_SENDRV_CFGID_GET_ATTR_SIGNAL:
            sen_get_attr_signal_os05a10((CTL_SENDRV_GET_ATTR_SIGNAL_PARAM *)(data));
            break;
        case CTL_SENDRV_CFGID_GET_ATTR_CMDIF:
            rt = sen_get_attr_cmdif_os05a10((CTL_SENDRV_GET_ATTR_CMDIF_PARAM *)(data));
            break;
        case CTL_SENDRV_CFGID_GET_ATTR_IF:
            rt = sen_get_attr_if_os05a10((CTL_SENDRV_GET_ATTR_IF_PARAM *)(data));
            break;
        case CTL_SENDRV_CFGID_GET_FPS:
            sen_get_fps_os05a10(id, (CTL_SENDRV_GET_FPS_PARAM *)(data));
            break;
        case CTL_SENDRV_CFGID_GET_SPEED:
            sen_get_speed_os05a10((CTL_SENDRV_GET_SPEED_PARAM *)(data));
```

```
        break;

    case CTL_SENDRV_CFGID_GET_MODE_BASIC:
        sen_get_mode_basic_os05a10((CTL_SENDRV_GET_MODE_BASIC_PARAM *)(data));
        break;

    case CTL_SENDRV_CFGID_GET_MODE_MIPI:
        sen_get_mode_mipi_os05a10((CTL_SENDRV_GET_MODE_MIPI_PARAM *)(data));
        break;

    case CTL_SENDRV_CFGID_GET_MODESEL:
        sen_get_modesel_os05a10((CTL_SENDRV_GET_MODESEL_PARAM *)(data));
        break;

    default:
        rt = E_NOSPT;
    }

    return rt;
}
```

4.3 使用者自定

[定義]

ctl_sen.h

```
typedef enum {  
    /* user define */  
    CTL_SENDRV_CFGID_USER_BASE = 0x03000000,  
    CTL_SENDRV_CFGID_USER_DEFINE1,  
    CTL_SENDRV_CFGID_USER_DEFINE2,  
    CTL_SENDRV_CFGID_USER_DEFINE3,  
    CTL_SENDRV_CFGID_USER_DEFINE4,  
    CTL_SENDRV_CFGID_USER_DEFINE5,  
    ENUM_DUMMY4WORD(CTL_SENDRV_CFGID)  
} CTL_SENDRV_CFGID;
```

[成員]

CTL_SENDRV_CFGID 列舉

成員名稱	描述
CTL_SENDRV_CFGID_USER_DEFINE1	使用者自定設定 1。
CTL_SENDRV_CFGID_USER_DEFINE2	使用者自定設定 2。
CTL_SENDRV_CFGID_USER_DEFINE3	使用者自定設定 3。
CTL_SENDRV_CFGID_USER_DEFINE4	使用者自定設定 4。
CTL_SENDRV_CFGID_USER_DEFINE5	使用者自定設定 5。

[舉例]

無

5 設定文檔

可根據專案應用及硬件設計，修改設定文檔。在載入 **Sensor ko** 檔時，加入指定路徑及檔名，則在初始化時，可讀取並解析文檔，並取代默認設定。

[定義]

```
[MAP]
path_1 = 1           #Path 1 Enable
path_2 = 0           #Path 2 Disable
path_3 = 0           #Path 3 Disable
path_4 = 0           #Path 4 Disable
path_5 = 0           #Path 5 Disable
path_6 = 0           #Path 6 Disable
path_7 = 0           #Path 7 Disable
path_8 = 0           #Path 8 Disable

[PRESET]
id_0_expt_time = 33333      #33333us
id_0_gain_ratio = 1000     #1x gain
id_1_expt_time = 33333      #33333us
id_1_gain_ratio = 1000     #1x gain

[DIRECTION]
id_0_mirror = 0           #no mirror
id_0_flip = 0             #no flip
id_1_mirror = 0           #no mirror
id_1_flip = 0             #no flip

[POWER]
id_0_mclk = 0             #CTL_SEN_CLK_SEL_SIEMCLK
id_0_pwn pin = 0xFFFFFFFF  #no pwn pin
id_0_rst_pin = 4           #S_GPIO_4
id_0_rst_time = 1         #1ms
id_0_stable_time = 1      #1ms
```



```
id_1_mclk = 0                #CTL_SEN_CLK_SEL_SIEMCLK
id_1_pwdn_pin = 0xFFFFFFFF   #no pwn pin
id_1_rst_pin = 5             #S_GPIO_4
id_1_rst_time = 2            #2ms
id_1_stable_time = 2         #2ms
```

[I2C]

```
id_0_i2c_id = 1              #SEN_I2C_ID_2
id_0_i2c_addr = 26           #0x34 >> 1
id_1_i2c_id = 0              #SEN_I2C_ID_1
id_1_i2c_addr = 26           #0x34 >> 1
```

【成員】

成員名稱	描述
MAP	指定所使用的通道。
PRESET	指定點亮時的曝光及增益。
DIRECTION	指定點亮時的水平及垂直翻轉。
POWER	指定點亮流程相關的參數設定。
I2C	指定 I2C 編號及位址。

6 AP 端的設定

6.1 線性模式

硬件接腳及 pinmux 的設定，目前是由 AP 設定，經由 HDAL 傳入 K flow。

[語法]

hd_videocapture.h

```
HD_RESULT hd_videocap_set(HD_PATH_ID path_id, HD_VIDEOCAP_PARAM_ID id, VOID *p_param);
```

[成員]

成員名稱	描述
path_id	路徑通道號。調用 hd_videocap_open() 後得到。
id	參數編號。指定配置參數類別。 Sensor 相關是使用 HD_VIDEOCAP_PARAM_DRV_CONFIG 列舉。
p_param	配置參數。使用 void * 指針類別。 Sensor 相關是使用 HD_VIDEOCAP_DRV_CONFIG 結構中的 HD_VIDEOCAP_SENSOR_DEVICE 結構，參閱下表。

HD_VIDEOCAP_SENSOR_DEVICE 結構

成員名稱	描述
driver_name	Sensor 名稱。需與 ko 檔的名字一致。
if_type	介面格式。例如：MIPI、LVDS、Parallel。
if_cfg	不需設定。
pin_cfg	pinmux.sensor_pinmux：介面格式，時脈相關的 pinmux 設定。 pinmux.serial_if_pinmux：介面接腳相關的 pinmux 設定。 pinmux.cmd_if_pinmux：I2C 相關的 pinmux 設定。 clk_lane_sel：串口時脈接腳設定。 sen_2_serial_pin_map：串口資料接腳設定。
option	不需設定。

[舉例]**video_liveview.c**

```
HD_RESULT set_cap_cfg(HD_PATH_ID *p_video_cap_ctrl)
{
    HD_RESULT ret = HD_OK;
    HD_VIDEOCAP_DRV_CONFIG cap_cfg = {0};
    HD_PATH_ID video_cap_ctrl = 0;

    snprintf(cap_cfg.sen_cfg.sen_dev.driver_name, HD_VIDEOCAP_SEN_NAME_LEN-1, "nvt_sen_imx317");
    cap_cfg.sen_cfg.sen_dev.if_type = HD_COMMON_VIDEO_IN_MIPI_CSI;
    cap_cfg.sen_cfg.sen_dev.pin_cfg.pinmux.sensor_pinmux = 0x220; //PIN_SENSOR_CFG_MIPI |
PIN_SENSOR_CFG_MCLK
    cap_cfg.sen_cfg.sen_dev.pin_cfg.pinmux.serial_if_pinmux = 0xf04; //PIN_MIPI_LVDS_CFG_CLK2 |
PIN_MIPI_LVDS_CFG_DAT0|PIN_MIPI_LVDS_CFG_DAT1 | PIN_MIPI_LVDS_CFG_DAT2 | PIN_MIPI_LVDS_CFG_DAT3
    cap_cfg.sen_cfg.sen_dev.pin_cfg.pinmux.cmd_if_pinmux = 0x10; //PIN_I2C_CFG_CH2
    cap_cfg.sen_cfg.sen_dev.pin_cfg.clk_lane_sel = HD_VIDEOCAP_SEN_CLANE_SEL_CSI0_USE_C2;
    cap_cfg.sen_cfg.sen_dev.pin_cfg.sen_2_serial_pin_map[0] = 0;
    cap_cfg.sen_cfg.sen_dev.pin_cfg.sen_2_serial_pin_map[1] = 1;
    cap_cfg.sen_cfg.sen_dev.pin_cfg.sen_2_serial_pin_map[2] = HD_VIDEOCAP_SEN_IGNORE;
    cap_cfg.sen_cfg.sen_dev.pin_cfg.sen_2_serial_pin_map[3] = HD_VIDEOCAP_SEN_IGNORE;
    cap_cfg.sen_cfg.sen_dev.pin_cfg.sen_2_serial_pin_map[4] = HD_VIDEOCAP_SEN_IGNORE;
    cap_cfg.sen_cfg.sen_dev.pin_cfg.sen_2_serial_pin_map[5] = HD_VIDEOCAP_SEN_IGNORE;
    cap_cfg.sen_cfg.sen_dev.pin_cfg.sen_2_serial_pin_map[6] = HD_VIDEOCAP_SEN_IGNORE;
    cap_cfg.sen_cfg.sen_dev.pin_cfg.sen_2_serial_pin_map[7] = HD_VIDEOCAP_SEN_IGNORE;
    ret = hd_videocap_open(0, HD_VIDEOCAP_0_CTRL, &video_cap_ctrl); //open this for device control
    if (ret != HD_OK) {
        return ret;
    }
    ret |= hd_videocap_set(video_cap_ctrl, HD_VIDEOCAP_PARAM_DRV_CONFIG, &cap_cfg);

    *p_video_cap_ctrl = video_cap_ctrl;
    return ret;
}
```

6.2 SHDR 模式

SHDR 模式需額外設定 SHDR Map，供 SIE 配置資源使用。

[語法]

hd_videocapture.h

```
HD_RESULT hd_videocap_set(HD_PATH_ID path_id, HD_VIDEOCAP_PARAM_ID id, VOID *p_param);
```

[成員]

成員名稱	描述
path_id	路徑通道號。調用 hd_videocap_open() 後得到。
id	參數編號。指定配置參數類別。 Sensor 相關是使用 HD_VIDEOCAP_PARAM_DRV_CONFIG 列舉。
p_param	配置參數。使用 void * 指針類別。 Sensor 相關是使用 HD_VIDEOCAP_DRV_CONFIG 結構中的 HD_VIDEOCAP_SEN_HDR_MAP 結構，參閱下表。

HD_VIDEOCAP_SEN_HDR_MAP 結構

成員名稱	描述
shdr_map	使用 HD_VIDEOCAP_SHDR_MAP(hdr_sensor, vdocap_id) 其中， hdr_sensor 為 sensor 編號。 vdocap_id 為使用的 VCAP 編號。

[舉例]

video_liveview_with_shdr.c

```
HD_RESULT set_cap_cfg(HD_PATH_ID *p_video_cap_ctrl, HD_OUT_ID _out_id, UINT32 *p_cap_pin_map,
HD_VIDEOCAP_SEN_HDR_MAP cap_shdr_map)
{
    HD_RESULT ret = HD_OK;

    HD_VIDEOCAP_DRV_CONFIG cap_cfg = {0};

    HD_PATH_ID video_cap_ctrl = 0;

    cap_cfg.sen_cfg.shdr_map = HD_VIDEOCAP_SHDR_MAP(HD_VIDEOCAP_HDR_SENSOR1,
```

```
(HD_VIDEOCAP_0|HD_VIDEOCAP_2));  
  
    ret |= hd_videocap_set(video_cap_ctrl, HD_VIDEOCAP_PARAM_DRV_CONFIG, &cap_cfg);  
  
    return ret;  
}
```

7 附錄

7.1 附錄：PQ 相關 KO 檔的載入

載入 Sensor 的 ko 檔前，需先載入 nvt_ctl_sen.ko。

[注意]

最多可掛載 8 個 ko 檔。

[範例]

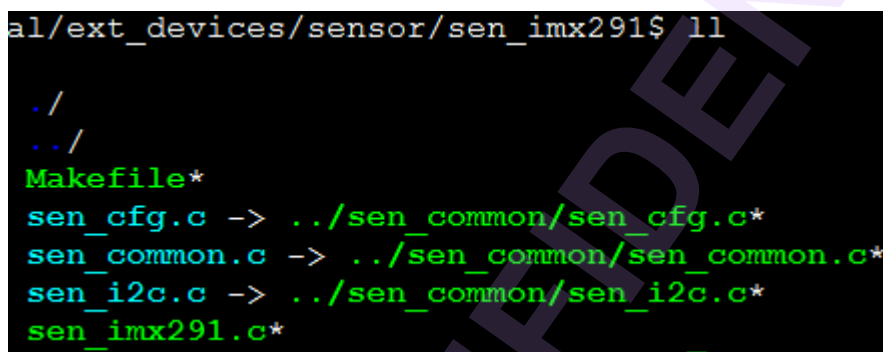
S10_SysInit2

```
# Necessary ko for sensor driver
insmod /lib/modules/4.1.0/hdal/kflow_videocapture/ctl_sen/nvt_ctl_sen.ko

#sensor
insmod /lib/modules/4.1.0/hdal/sen_os02k10/nvt_sen_os02k10.ko
insmod /lib/modules/4.1.0/hdal/sen_os05a10/nvt_sen_os05a10.ko
insmod /lib/modules/4.1.0/hdal/sen_imx317/nvt_sen_imx317.ko
```

7.2 附錄：連結共用檔案

因為使用 multi-build 功能，故需使用連結共用檔案的方式，確保編譯時不會發生錯誤。由圖表 7 可發現”sen_cfg.c”、”sen_common.c”、及”sen_i2c.c”是使用連結到 sen_common 這目錄的共用檔案。



```
al/ext_devices/sensor/sen_imx291$ ll
./
../
Makefile*
sen_cfg.c -> ../sen_common/sen_cfg.c*
sen_common.c -> ../sen_common/sen_common.c*
sen_i2c.c -> ../sen_common/sen_i2c.c*
sen_imx291.c*
```

圖表 7：ls-l 下的檔案顯示

在新增 Sensor 驅動代碼時，需下以下命令來連結共用檔案。

- ln -s ../sen_common/sen_cfg.c sen_cfg.c
- ln -s ../sen_common/sen_common.c sen_common.c
- ln -s ../sen_common/sen_i2c.c sen_i2c.c

7.3 附錄：設定曝光及增益的彈性

在 Sensor 驅動中的讀寫曝光及增益，其參數為 void *param，如表格 1 所示。

```
static void sen_set_gain_os05a10(CTL_SEN_ID id, void *param);  
static void sen_set_expt_os05a10(CTL_SEN_ID id, void *param);  
static void sen_get_expt_os05a10(CTL_SEN_ID id, void *param);
```

表格 1：讀寫曝光及增益函數的宣告

代表 K flow 在讀寫曝光及增益時，不經手這 2 個設定值的型態。此設定的型態是宣告在 isp_api.h，使用者可根據 AE 算法的需求，自行定義型態，如表格 2 所示。

isp_api.h

```
typedef struct {  
    UINT32 exp_time[ISP_SEN_MFRAME_MAX_NUM];  
    UINT32 gain_ratio[ISP_SEN_MFRAME_MAX_NUM];  
} ISP_SENSOR_CTRL;
```

表格 2：ISP_SENSOR_CTRL 結構

7.4 設定參數

此章節描述 4.2 中相關的結構體。不同 Sensor 其結構內容都不一樣，可直接參考各 Sensor 驅動代碼。

7.4.1 基本參數

對應 CTL_SENDRV_CFGID_GET_ATTR_BASIC。描述 Sensor 基本參數，此部份的參數與設定值無關聯。

[定義]

ctl_sen.h

```
typedef struct {
    CHAR name[CTL_SEN_NAME_LEN];
    CTL_SEN_VENDOR vendor;
    UINT32 max_senmode;
    CTL_SEN_SUPPORT_PROPERTY property;
    UINT32 sync_timing;
} CTL_SENDRV_GET_ATTR_BASIC_PARAM;
```

[成員]

成員名稱	描述
name	名稱。
vendor	廠商名稱，介面為 LVDS 時需填此參數。
max_senmode	可提供的模式數目。
property	是否支援水平翻轉、垂直翻轉、及動態幀率。
sync_timing	曝光與增益的生效時間差。以 VD 為單位，通常為 0 或 1。

[舉例]

sen_os05a10.c

```
#include "kflow_videocapture/ctl_sen.h"
```

```
static CTL_SENDRV_GET_ATTR_BASIC_PARAM basic_param = {
    SEN_OS05A10_MODULE_NAME,
    CTL_SEN_VENDOR_OMNIVISION,
    SEN_MAX_MODE,
    CTL_SEN_SUPPORT_PROPERTY_MIRROR|CTL_SEN_SUPPORT_PROPERTY_FLIP|CTL_SEN_SUPPORT_PROPERTY_CHGFP
S,
    0
};
```

7.4.2 信號參數

對應 CTL_SENDRV_CFGID_GET_ATTR_SIGNAL。描述 Sensor 信號的主從模式，及訊號特性。

【定義】

ctl_sen.h

```
typedef struct {
    CTL_SEN_SIGNAL_TYPE type;
    CTL_SEN_SIGNAL_INFO info;
} CTL_SENDRV_GET_ATTR_SIGNAL_PARAM;
```

【成員】

成員名稱	描述
type	設定為 CTL_SEN_SIGNAL_SLAVE 時，由主控提供 HD/VD 訊號。需額外設定 TGE 參數，參閱 7.4.10。
info	設定為 CTL_SEN_SIGNAL_MASTER 時，描述 Sensor 並口訊號，分別為 HD/VD 的有效極性，及 PCLK 的觸發相位。 設定為 CTL_SEN_SIGNAL_SLAVE 時，描述 TGE 訊號的，分別為 HD/VD 的有效極性，及來源時脈的觸發相位。

【舉例】

sen_os05a10.c

```
#include "kflow_videocapture/ctl_sen.h"

static CTL_SENDRV_GET_ATTR_SIGNAL_PARAM signal_param = {
```

```

CTL_SEN_SIGNAL_MASTER,
{CTL_SEN_ACTIVE_HIGH, CTL_SEN_ACTIVE_HIGH, CTL_SEN_PHASE_RISING, CTL_SEN_PHASE_RISING,
CTL_SEN_PHASE_RISING}
};

```

7.4.3 指令介面參數

對應 CTL_SENDRV_CFGID_GET_ATTR_CMDIF。描述讀寫暫存器的指令介面一般使用的指令介面為 I2C。

[定義]

ctl_sen.h

```

typedef struct {
    CTL_SEN_CMDIF_TYPE type;
    union {
        CTL_SENDRV_I2C i2c;
        CTL_SENDRV_SIF sif;
    } info;
    CTL_SENDRV_VX1 vx1;
} CTL_SENDRV_GET_ATTR_CMDIF_PARAM;

```

[成員]

成員名稱	描述
type	指令介面的型態，支援 VX1、SIF、I2C、及 IO。
i2c	I2C 介面參數。
sif	SIF 介面參數。
vx1	VX1 介面參數。

[舉例]

sen_os05a10.c

```

#include "kflow_videocapture/ctl_sen.h"

static CTL_SENDRV_I2C i2c = {
    {
        {CTL_SEN_I2C_W_ADDR_DFT,    0x6C},
    }
}

```

```

        {CTL_SEN_I2C_W_ADDR_OPTION1, 0xFF},
        {CTL_SEN_I2C_W_ADDR_OPTION2, 0xFF},
        {CTL_SEN_I2C_W_ADDR_OPTION3, 0xFF},
        {CTL_SEN_I2C_W_ADDR_OPTION4, 0xFF},
        {CTL_SEN_I2C_W_ADDR_OPTION5, 0xFF}
    }
};

static ER sen_get_attr_cmdif_os05a10(CTL_SENDRV_GET_ATTR_CMDIF_PARAM *data)
{
    data->type = CTL_SEN_CMDIF_TYPE_I2C;
    memcpy(&data->info, &i2c, sizeof(CTL_SENDRV_I2C));
    return E_OK;
}

```

7.4.4 幀率參數

對應 CTL_SENDRV_CFGID_GET_FPS。

[定義]

ctl_sen.h

```

typedef struct {
    UINT32 chg_fps;
    UINT32 cur_fps;
} CTL_SENDRV_GET_FPS_PARAM;

```

[成員]

成員名稱	描述
chg_fps	模式切換幀率。在切換模式時，由上層帶入的幀率，在 Sensor 不支援調整幀率時，會設定為該模式下的默認幀率。
cur_fps	目前幀率。由模式切換幀率，及由目前設定下算出來的幀率取最小值。

[舉例]

sen_os05a10.c

```
#include "kflow_videocapture/ctl_sen.h"

static void sen_get_fps_os05a10(CTL_SEN_ID id, CTL_SENDRV_GET_FPS_PARAM *data)
{
    data->cur_fps = sen_get_cur_fps_os05a10(id);
    data->chg_fps = sen_get_chgmode_fps_os05a10(id);
}
```

7.4.5 時脈參數

對應 CTL_CTL_SENDRV_CFGID_GET_SPEED。描述不同的 Sensor 模式下跟時脈相關的參數。

[注意]

過小的 data_rate 會導致平台無法正常接收傳輸資料。

[定義]

ctl_sen.h

```
typedef struct {
    CTL_SEN_MODE mode;
    CTL_SEN_SIEMCLK_SRC mclk_src;
    UINT32 mclk;
    UINT32 pclk;
    UINT32 data_rate;
} CTL_SENDRV_GET_SPEED_PARAM;
```

[成員]

成員名稱	描述
mode	Sensor 模式。此為輸入參數。
mclk_src	工作時脈來源。
mclk	工作時脈。
pclk	像素時脈。

data_rate	<p>CSI 資料傳輸率。根據以下公式填值。</p> $\text{data_rate} = \text{MIPI_data_rate} * \text{lane_number} / \text{bit_number} / 2$ <p>其中，</p> <p>MIPI_data_rate：總資料傳輸率</p> <p>lane_number：通道數目</p> <p>bit_number：每像素的位元數</p>
-----------	-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

[舉例]

sen_os05a10.c

```
#include "kflow_videocapture/ctl_sen.h"

static CTL_SENDRV_GET_SPEED_PARAM speed_param[SEN_MAX_MODE] = {
    {
        CTL_SEN_MODE_1,
        CTL_SEN_SIEMCLK_SRC_MCLK,
        24000000,
        54000000,
        218000000
    },
    {
        CTL_SEN_MODE_2,
        CTL_SEN_SIEMCLK_SRC_MCLK,
        24000000,
        54000000,
        300000000
    },
};
```

7.4.6 模式選擇參數

對應 CTL_SENDRV_CFGID_GET_MODESEL。此結構大部份參數由上層傳入，Sensor 經由計算，回傳最合適的 Sensor 模式供上層參考。

[定義]

ctl_sen.h

```
typedef struct {
    UINT32 frame_rate;
    USIZE size;
    CTL_SEN_IF_TYPE if_type;
    CTL_SEN_DATA_FMT data_fmt;
    UINT32 frame_num;
    CTL_SEN_PIXDEPTH pixdepth;
    CTL_SEN_MODESEL_CCIR ccir;
    BOOL mux_signal_en;
    CTL_SEN_PARA_MUX_INFO mux_signal_info;
    CTL_SEN_MODE mode;
} CTL_SENDRV_GET_MODESEL_PARAM;
```

【成員】

成員名稱	描述
frame_rate	幀率。精度為 100。
size	分辨率。
if_type	介面型態。使用 CTL_SEN_IF_TYPE 列舉表示。 一般為 CTL_SEN_IF_TYPE_LVDS 或 CTL_SEN_IF_TYPE_MIPI。
data_fmt	像素格式。使用 CTL_SEN_DATA_FMT 列舉表示。 一般為 CTL_SEN_DATA_FMT_RGB。
frame_num	幀數目。在 SHDR 模式時使用。
pixdepth	像數位元數。使用 CTL_SEN_PIXDEPTH 列舉表示。 一般為 CTL_SEN_PIXDEPTH_10BIT 或 CTL_SEN_PIXDEPTH_12BIT。
ccir	CCIR 格式。使用 CTL_SEN_MODESEL_CCIR 結構表示。 YUV Sensor 使用。
mux_signal_en	是否為混合信號。介面為並口時使用。
mux_signal_info	混合信號訊息。當 mux_signal_en 為 1 時用，描述混合訊號的數目。
mode	Sensor 模式。此為輸出參數。

【舉例】

sen_os05a10.c

```
#include "kflow_videocapture/ctl_sen.h"
```

```
static void sen_get_modesel_os05a10(CTL_SENDRV_GET_MODESEL_PARAM *data)
{
    if (data->if_type != CTL_SEN_IF_TYPE_MIPI) {
        DBG_ERR("if_type %d N.S. \r\n", data->if_type);
        return;
    }

    if (data->data_fmt != CTL_SEN_DATA_FMT_RGB) {
        DBG_ERR("data_fmt %d N.S. \r\n", data->data_fmt);
        return;
    }

    if (data->frame_num == 1) {
        if ((data->size.w <= 2592) && (data->size.h <= 1944)) {
            if (data->frame_rate <= 3000) {
                data->mode = CTL_SEN_MODE_1;
                return;
            }
        }
    }
    else if (data->frame_num == 2) {
        if ((data->size.w <= 2592) && (data->size.h <= 1944)) {
            if (data->frame_rate <= 3000) {
                data->mode = CTL_SEN_MODE_2;
                return;
            }
        }
    }

    data->mode = CTL_SEN_MODE_1;
}
```

7.4.7 模式基本參數

對應 CTL_SENDRV_CFGID_GET_MODE_BASIC。為不同的 Sensor 模式下對應的訊

息。

[定義]

ctl_sen.h

```
typedef struct {
    CTL_SEN_MODE mode;
    CTL_SEN_IF_TYPE if_type;
    CTL_SEN_DATA_FMT data_fmt;
    CTL_SEN_MODE_TYPE mode_type;
    UINT32 dft_fps;
    UINT32 frame_num;
    CTL_SEN_STPIX stpix;
    CTL_SEN_PIXDEPTH pixel_depth;
    CTL_SEN_FMT fmt;
    USIZE valid_size;
    URECT act_size[CTL_SEN_MFRAME_MAX_NUM];
    USIZE crp_size;
    CTL_SEN_MODE_SIGNAL signal_info;
    UINT32 ratio_h_v;)
    CTL_SEN_MODE_GAIN gain;
    UINT32 binning_ratio;
} CTL_SENDRV_GET_MODE_BASIC_PARAM;
```

[成員]

成員名稱	描述
mode	Sensor 模式。此為輸入參數。
if_type	介面型態。使用 CTL_SEN_IF_TYPE 列舉表示。 一般為 CTL_SEN_IF_TYPE_LVDS 或 CTL_SEN_IF_TYPE_MIPI。
data_fmt	像素格式。使用 CTL_SEN_DATA_FMT 列舉表示。 一般為 CTL_SEN_DATA_FMT_RGB。
mode_type	模式型態。使用 CTL_SEN_MODE_TYPE 列舉。 一般為 CTL_SEN_MODE_LINEAR 或 CTL_SEN_MODE_STAGGER_HDR。
dft_fps	幀率。
frame_num	幀數目。在線性模式為 1。在 SHDR 模式時大於 1。
stpik	初始像素點。

pixel_depth	像數位元數。使用 CTL_SEN_PIXDEPTH 列舉表示。 一般為 CTL_SEN_PIXDEPTH_10BIT 或 CTL_SEN_PIXDEPTH_12BIT。
fmt	使用 CTL_SEN_FMT 列舉。固定使用 CTL_SEN_FMT_POGRESSIVE。
valid_size	輸出分辨率。
act_size	SIE 接收分辨率。需小於或等於 valid_size。
crp_size	SIE 裁剪分辨率。需小於或等於 act_size。
signal_info	信號訊息。使用 CTL_SEN_MODE_SIGNAL 結構。 hd_period 為列長度(Line length)，為 AE 算法計算曝光時間使用。 vd_period 為幀長度，為調整幀率使用。 hd_sync 及 vd_sync 設定為 0。
ratio_h_v	長寬比。 4 比 3 時使用 CTL_SEN_RATIO(4, 3)。 16 比 9 時使用 CTL_SEN_RATIO(16, 9)
gain	增益。精度為 1000。使用 CTL_SEN_MODE_GAIN 結構。 min 為最小增益，一般設定為 1000。 max 為最大增益。
binning_ratio	Binning 比率。當使用 Binning 模式時需給定此訊息。 一般為 100。

[舉例]

sen_os05a10.c

```
#include "kflow_videocapture/ctl_sen.h"

static CTL_SENDRV_GET_MODE_BASIC_PARAM mode_basic_param[SEN_MAX_MODE] = {
    {
        CTL_SEN_MODE_1,
        CTL_SEN_IF_TYPE_MIPI,
        CTL_SEN_DATA_FMT_RGB,
        CTL_SEN_MODE_LINEAR,
        3000,
        1,
        CTL_SEN_STPIX_R,
        CTL_SEN_PIXDEPTH_10BIT,
        CTL_SEN_FMT_POGRESSIVE,
    }
}
```

```

        {2688, 1946},
        {{47, 2, 2592, 1944}, {0, 0, 0, 0}, {0, 0, 0, 0}, {0, 0, 0, 0}},
        {2592, 1944},
        {0, 720, 0, 2496},
        CTL_SEN_RATIO(4, 3),
        {1000, 248000},
        100
    },
    {
        CTL_SEN_MODE_2,
        CTL_SEN_IF_TYPE_MIPI,
        CTL_SEN_DATA_FMT_RGB,
        CTL_SEN_MODE_STAGGER_HDR,
        3000,
        2,
        CTL_SEN_STPIX_R,
        CTL_SEN_PIXDEPTH_10BIT,
        CTL_SEN_FMT_POGRESSIVE,
        {2688, 1946},
        {{47, 2, 2592, 1944}, {47, 2, 2592, 1944}, {0, 0, 0, 0}, {0, 0, 0, 0}},
        {2592, 1944},
        {0, 720, 0, 2496},
        CTL_SEN_RATIO(4, 3),
        {1000, 248000},
        100
    },
};

```

7.4.8 LVDS 參數

對應 CTL_SENDRV_CFGID_GET_MODE_LVDS。當介面為 LVDS 時的相關參數。

[定義]

ctl_sen.h

```
typedef struct {
    CTL_SEN_MODE mode;
    CTL_SEN_CLKLANE clk_lane;
    CTL_SEN_DATA_LANE data_lane;
    UINT32 output_pixel_order[CTL_SEN_LVDS_MAX_DATA_LANE]
    UINT32 sel_frm_id[CTL_SEN_MFRAME_MAX_NUM];
    UINT16 sel_bit_ofs;
    UINT32 fset_bit;
    CTL_SEN_DATA_IN_BIT_ORDER data_in_order;
    UINT32 rev;
} CTL_SENDRV_GET_MODE_LVDS_PARAM;
```

【成員】

成員名稱	描述
mode	Sensor 模式。此為輸入參數。
clk_lane	時脈通道數。使用 CTL_SEN_CLKLANE 列舉。 一般為 CTL_SEN_CLKLANE_1。
data_lane	資料通道數。使用 CTL_SEN_DATA_LANE 列舉。 一般為 CTL_SEN_DATA_LANE_2 或 CTL_SEN_DATA_LANE_4。
output_pixel_order	輸出像素排列。此部份與硬件設計相關。 一般為{0, 1}或{0, 1, 2, 3}。
sel_frm_id	SHDR 模式時，資料於 data_lane 的順序。 一般為{0, 1, 0, 0}或{0, 1, 2, 3}。
sel_bit_ofs	不需設定。
fset_bit	不需設定。
data_in_order	不需設定。
rev	不需設定。

7.4.9 MIPI 參數

對應 CTL_SENDRV_CFGID_GET_MODE_MIPI。當介面為 MIPI 時的相關參數。

【定義】

ctl_sen.h

```
typedef struct {
    CTL_SEN_MODE mode;
    CTL_SEN_CLKLANE clk_lane;
    CTL_SEN_DATA_LANE data_lane;
    CTL_SEN_MIPI_MANUAL_INFO manual_info[CTL_SEN_MIPI_MAX_MANUAL];
    BOOL save_pwr;
    UINT32 sel_frm_id[CTL_SEN_MFRAME_MAX_NUM];
    UINT16 sel_bit_ofs;
} CTL_SENDRV_GET_MODE_MIPI_PARAM;
```

【成員】

成員名稱	描述
mode	Sensor 模式。此為輸入參數。
clk_lane	時脈通道數。使用 CTL_SEN_CLKLANE 列舉。 一般為 CTL_SEN_CLKLANE_1。
data_lane	資料通道數。使用 CTL_SEN_DATA_LANE 列舉。 一般為 CTL_SEN_DATA_LANE_2 或 CTL_SEN_DATA_LANE_4。
manual_info	指定接收特定資料時使用。最多 3 組。
save_pwr	省電模式，資料傳輸率需小於 1Gbps。
sel_frm_id	SHDR 模式時，資料於 data_lane 的順序。 一般為{0, 1, 0, 0}或{0, 1, 2, 3}。
sel_bit_ofs	設定為 SEN_BIT_OFS_NONE。

【舉例】

sen_os05a10.c

```
#include "kflow_videocapture/ctl_sen.h"

static CTL_SENDRV_GET_MODE_MIPI_PARAM mipi_param[SEN_MAX_MODE] = {
    {
        CTL_SEN_MODE_1,
        CTL_SEN_CLKLANE_2,
        CTL_SEN_DATA_LANE_4,
        { {CTL_SEN_MIPI_MANUAL_NONE, 0}, {CTL_SEN_MIPI_MANUAL_NONE, 0}, {CTL_SEN_MIPI_MANUAL_NONE, 0} },
        0,
        {0, 0, 0, 0},
    }
};
```

```

        SEN_BIT_OFS_NONE
    },
    {
        CTL_SEN_MODE_2,
        CTL_SEN_CLKLANE_2,
        CTL_SEN_DATA_LANE_4,
        { {CTL_SEN_MIPI_MANUAL_NONE, 0}, {CTL_SEN_MIPI_MANUAL_NONE, 0}, {CTL_SEN_MIPI_MANUAL_NONE,
0} },
        0,
        {0, 1, 0, 0},
        SEN_BIT_OFS_0 | SEN_BIT_OFS_1
    },
};

```

7.4.10 TGE 參數

對應 CTL_SENDRV_CFGID_GET_MODE_TGE。當 Sensor 需要主控提供 HD/VD 訊號時的相關參數。

[定義]

ctl_sen.h

```

typedef struct {
    CTL_SEN_MODE mode;
    CTL_SEN_MODE_SIGNAL signal;
} CTL_SENDRV_GET_MODE_TGE_PARAM;

```

[成員]

成員名稱	描述
mode	Sensor 模式。此為輸入參數。
signal	信號訊息。 hd_sync 為水平信號起始點。 hd_period 為水平信號長度。 vd_sync 為垂直信號起始點。 vd_period 為垂直信號長度。

7.5 附錄：除錯

以下提供常用的除錯指令。更詳細的說明請參閱” NT96680_PureLinux ctl_sen FAQ”這份文件。

7.5.1 列印設定

使用此指令可列印 CTL_SEN_OBJ / get_cfg / CTL_SEN_CFGID 的設定值。

[語法]

echo dbg 0 0x1 > /proc/ctl_sen/cmd

```
=====sensor info=====
-----ATTR-----
      name      vendor  maxmode cmdif_type  if_type  sen_map_if  property  sync  sgl_type  vd_act  hd_act  vd pha  hd pha  dt pha
      sen_imx291  SONY    1      I2C      MIPI      DFT  0x00000007  0     MAST      H      H      R      R      R
-----CMDIF-----
      type
      I2C
-----IF-----
      mode      type
      1      MIPI
-----FPS-----
      dft      chg      cur
      3000     3000     3000
-----SPEED-----
      mode      mclk      pclk      dt_rate
      1      37125000  148500000  148500000
-----CLK_INFO-----
      mclk_freq  mclk_sel  mclk_src  data_rate
      37125000  MCLK      PLL5      240000000
-----MODE_BASIC-----
      mode  data_fmt mode_type  dft_fps  frame_num  stpidx  depth  fmt
      1      RGB  LINEAR  3000     1      R      12     POGRESSIVE
      valid.w  valid.h  act.x  act.y  act.w  act.h  crp.w  crp.h  hd_sync  hd_period  vd_sync  vd_period
      1920     1080     0      0      1920   1080   1920   1080   0      4400     0      1125
      ratio_h_v  gain.min  gain.max  binning  rt
      0x00100009  1000     4096000  100
-----IF_MAP-----
      map_if
      DFT
-----TIMEOUT-----
      timeout(ms)
      1000
=====sensor status=====
SEL      dest
AUTO     0x00000000
AUTO:    fps      sz.w      sz.h      num      fmt      bit      ccir_int  ccir_fmt  mux_en  mux_num
        3000    1920     1080     1      RGB      12      0      CCIR601  0      0
cur_senmode
        1
```


7.5.2 列印驅動訊息

使用此指令可列印 CTL_SEN_REG_OBJ / CTL_SEN_DRV_TAB / get_cfg / CTL_SENDRV_CFGID 的設定值。

【語法】

echo dbg 0 0x4 > /proc/ctl_sen/cmd

```
=====sensor driver info=====
----ATTR_BASIC-----
      name      vendor      maxmode      sp_prop sync
      sen_imx291  SONY      1 0x00000007  0
----ATTR_SIGNAL-----
      type vd_act hd_act vd pha hd pha dt pha
      MAST  H      H      R      R      R
----ATTR_CMDIF-----
      type
      I2C
ch      w_addr_sel  w_addr  clk
I2C1 DFT      0x00  0
w_addr_sel:w_addr  w_addr_sel:w_addr  w_addr_sel:w_addr  w_addr_sel:w_addr  w_addr_sel:w_addr  w_addr_sel:w_addr
DFT      :0x34  OP1      :0xff  OP2      :0xff  OP3      :0xff  OP4      :0xff  OP5      :0xff
----ATTR_IF-----
      type
      MIPI
----FPS-----
      chg      cur
      3000  3000
----SPEED-----
mode  mclk_src  mclk      pclk      dt_rate
1      MCLK  37125000  148500000  148500000
----MODE_BASIC-----
mode  if_type  data_fmt  mode_type  dft_fps  frame_num  stpixmap_depth  fmt
1      MIPI  RGB  LINEAR  3000  1  R  12  PROGRESSIVE
valid.w  valid.h  act.x  act.y  act.w  act.h  crp.w  crp.h  hd_sync  hd_period  vd_sync  vd_period
1920  1080  0  0  1920  1080  1920  1080  0  4400  0  1125
ratio_h_v  gain.min  gain.max  bining_rt
0x00100009  1000  4096000  100
----MODE_LVDS-----
----MODE_MIPI-----
mode  cl  dl  mn_bit0  mn_id0  mn_bit1  mn_id1  mn_bit2  mn_id2  svpwr  sl_id0  sl_id1  sl_ofs
1  1  2  0  0  0  0  0  0  0  0  0  0
----MODE_DVI-----
----MODE_TGE-----
```

7.5.3 列印 VD

使用此指令可列印 SIE 收到 VD 的時間週期。

【語法】

echo dbg 0 0x8 > /proc/ctl_sen/cmd

```
=====sensor wait inte=====
inte(VD    ) Time 33287 us
inte(VD    ) Time 33335 us
inte(VD    ) Time 33333 us
inte(VD    ) Time 33330 us
inte(VD    ) Time 33328 us
inte(VD    ) Time 33328 us
inte(VD    ) Time 33348 us
inte(VD    ) Time 33343 us
inte(VD    ) Time 33331 us
inte(VD    ) Time 33328 us
```