SigmaStar
High Performance IPCAM/PCCAM System-on-Chip

AE/AWB/AF Interface Version 1.2

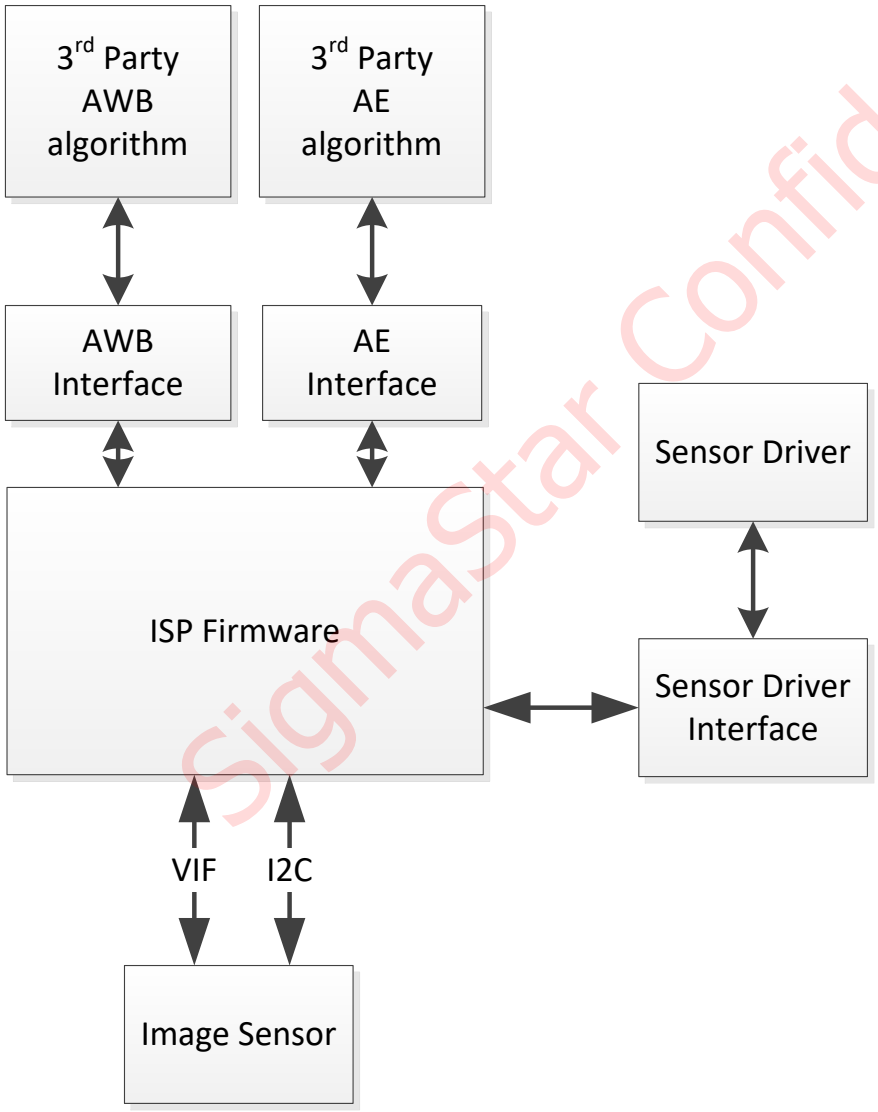
REVISION HISTORY

Revision No.	Description	Date
1.0	• Formal release	02/19/2019
1.1	• Added 3A RegInterface	03/06/2019
1.2	• Updated AF Windows Description	03/14/2019

TABLE OF CONTENTS

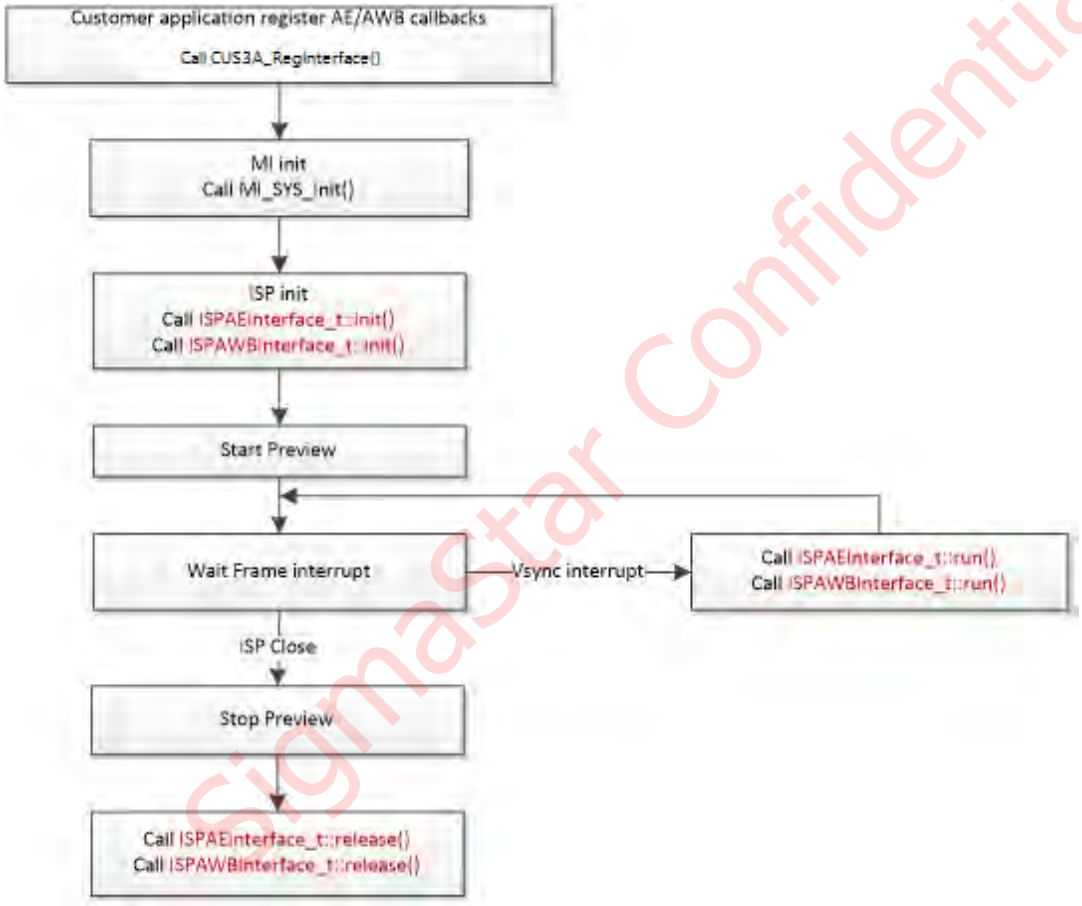
1. 系統架構

ISP 架構，除主要 ISP 核心外，另外可外掛三種模組。AWB/AE 演算法庫提透過 AWB/AE Interface 與 ISP 連結，客戶可自行開發演算法，ISP 透過 AWB/AE Interface 提供硬體統計值給演算法，演算法回報對應的結果給 ISP，ISP 核心將結果透過適當的方式寫入硬體。Sensor 驅動透過 Sensor Interface 與 ISP 連結，不同型號的 Sensor 必須實作對應的 Sensor 驅動。



2. AE/AWB 運作流程

客戶程式必須在 MI_SYS_Init()之前透過 CUS3A_RegInterface()註冊演算法庫(callback function)，之後 ISP 初始化階段，ISP 核心透過 init()通知演算法進行初始化工作。當前端 Sensor 開始取影像，ISP 核心會在每偵影像中斷時呼叫演算法 run()，並且將硬體統計值帶入演算法中。當 ISP 關閉時，透過 release()通知演算法釋放所有資源。



流程圖中紅色字體為演算法必須提供的 callback function。

3. AE/AWB STATISTIC FORMAT

AE/AWB 統計值在程式碼中定義如下，每張影像均會產生 128*90 筆取樣資料。
在 Macaron 中，AE 統計值，每張影像只會產生 32*32 筆取樣資料。

```
#define isp_3A_ROW    (128)   /**< number of 3A statistic blocks in a row
*/
#define isp_3A_COL    (90)    /**< number of 3A statistic blocks in a
column */ @brief Single AE HW statistics data*/
typedef struct
{
    U8  r;    /**< block pixel R average , 0~255*/
    U8  g;    /**< block pixel G average , 0~255, g=(Gr+Gb+1)>>1*/
    U8  b;    /**< block pixel B average , 0~255*/
    U8  y;    /**< block pixel Y average , 0~255, \
         y=(Ravg*9 +Gavg*19 +Bavg*4 + 16)>>5 */
}__attribute__((packed, aligned(1))) ISP_AE_SAMPLE;

/*! @brief Single AWB HW statistics data*/
typedef struct
{
    U8  r;    /**<center pixel R value , 0~255   */
    U8  g;    /**<center pixel G value , 0~255   */
    U8  b;    /**<center pixel B value , 0~255   */
}__attribute__((packed, aligned(1))) ISP_AWB_SAMPLE;
```

ISP 會在每次 Frame End 將 AE/AWB 統計值傳給演算法端，資料排列方式如下表。ISP 將影像切為 isp_3A_ROW x isp_3A_COL 區域，並計算每個區域統計值填入
ISP_AWB_INFO::data[isp_3A_ROW*isp_3A_COL]
ISP_AE_INFO::data[isp_3A_ROW*isp_3A_COL]

0	1	2	3	.....	isp_3A_ROW-1
isp_3A_ROW	isp_3A_ROW +1	isp_3A_ROW +2	isp_3A_ROW +3	. . . . .	isp_3A_ROW*2-1
isp_3A_ROW*2					
.					
isp_3A_ROW* (isp_3A_COL-1)	isp_3A_ROW* (isp_3A_COL-1)+1				isp_3A_ROW* isp_3A_COL-1

4. AF STATISTIC FORMAT

設置 AF 統計 window(ISP_AF_RECT),提供 16*n(n = 1~16)個 window 可供設置(af_stats_win[16]),每禎會將統計結果 output, output 結構如 ISP_AF_STATS,每個 Window 會 output 6 種 filter 的統計值-

各 filter bit 數表示如下

Filter Bit & Chip Type	Twinkie	Pretzel	Macaron
IIR / Sobel	34	35	35
Luma	34	32	32
YSat	24	22	22

AF ROI Mode = AF_ROI_MODE_NORMAL: 只看CusAFStats_t.stParaAPI[0]
AF ROI Mode = AF_ROI_MODE_MATRIX: 可看CusAFStats_t.stParaAPI[0~15]

```
typedef struct {
    MI_U32 u32StartX;   /*range : 0~1023*/
    MI_U32 u32StartY;   /*range : 0~1023*/
    MI_U32 u32EndX;     /*range : 0~1023*/
    MI_U32 u32EndY;     /*range : 0~1023*/
} AF_WINDOW_PARAM_t;

typedef struct {
    MI_U8 u8WindowIndex;
    AF_WINDOW_PARAM_t stParaAPI;
} CusAFWin_t;

typedef struct{
    MI_U8 high_iir[5*16];
    MI_U8 low_iir[5*16];
    MI_U8 luma[4*16];
    MI_U8 sobel_v[5*16];
    MI_U8 sobel_h[5*16];
    MI_U8 ysat[3*16];
} AF_STATS_PARAM_t ;

typedef struct{
    AF_STATS_PARAM_t stParaAPI[16];
stParaAPI;
} CusAFStats_t;
```

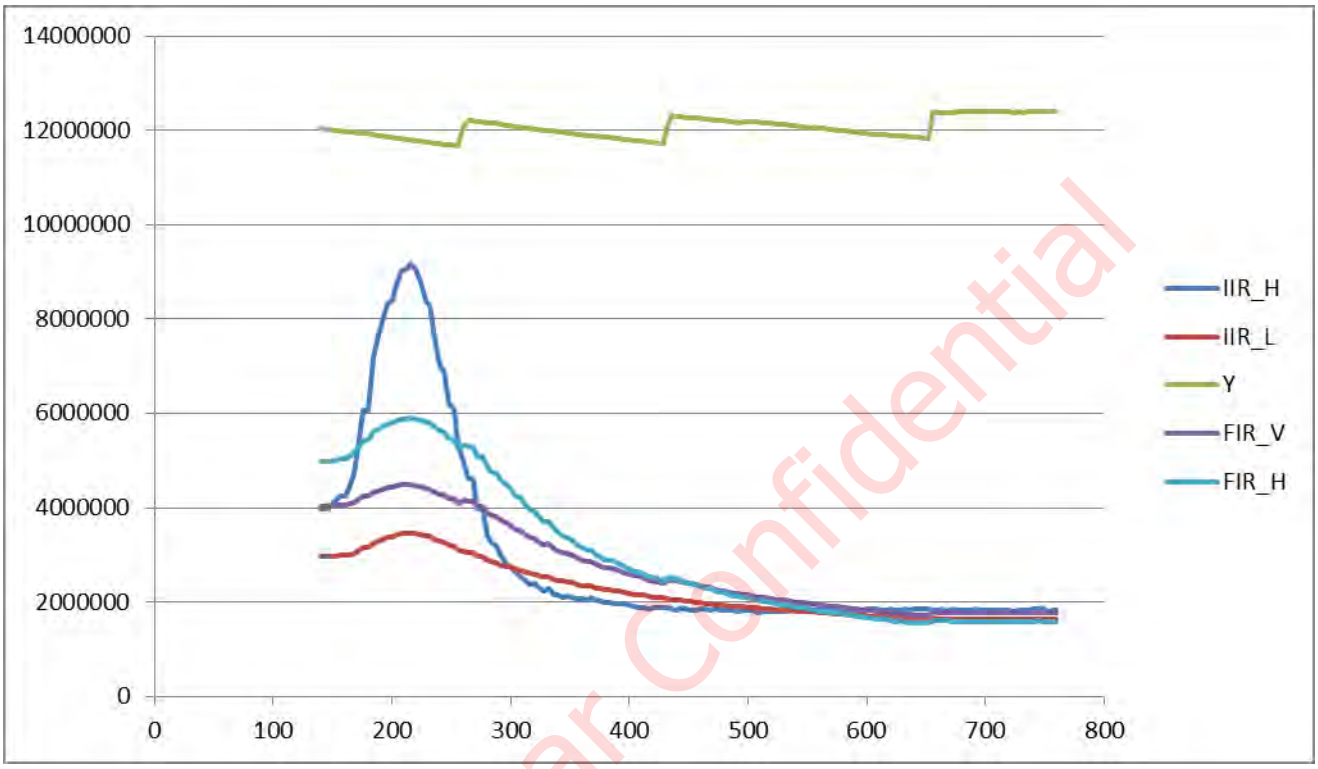
```
typedef struct{
    MI_U8 high_iir[6*16];
    MI_U8 low_iir[6*16];
    MI_U8 luma[6*16];
    MI_U8 sobel_v[6*16];
    MI_U8 sobel_h[6*16];
    MI_U8 ysat[3*16];
} AF_STATS_PARAM_t ;

typedef struct{
    AF_STATS_PARAM_t

} CusAFStats_t;
```


4.1. 各 AF Filter 說明

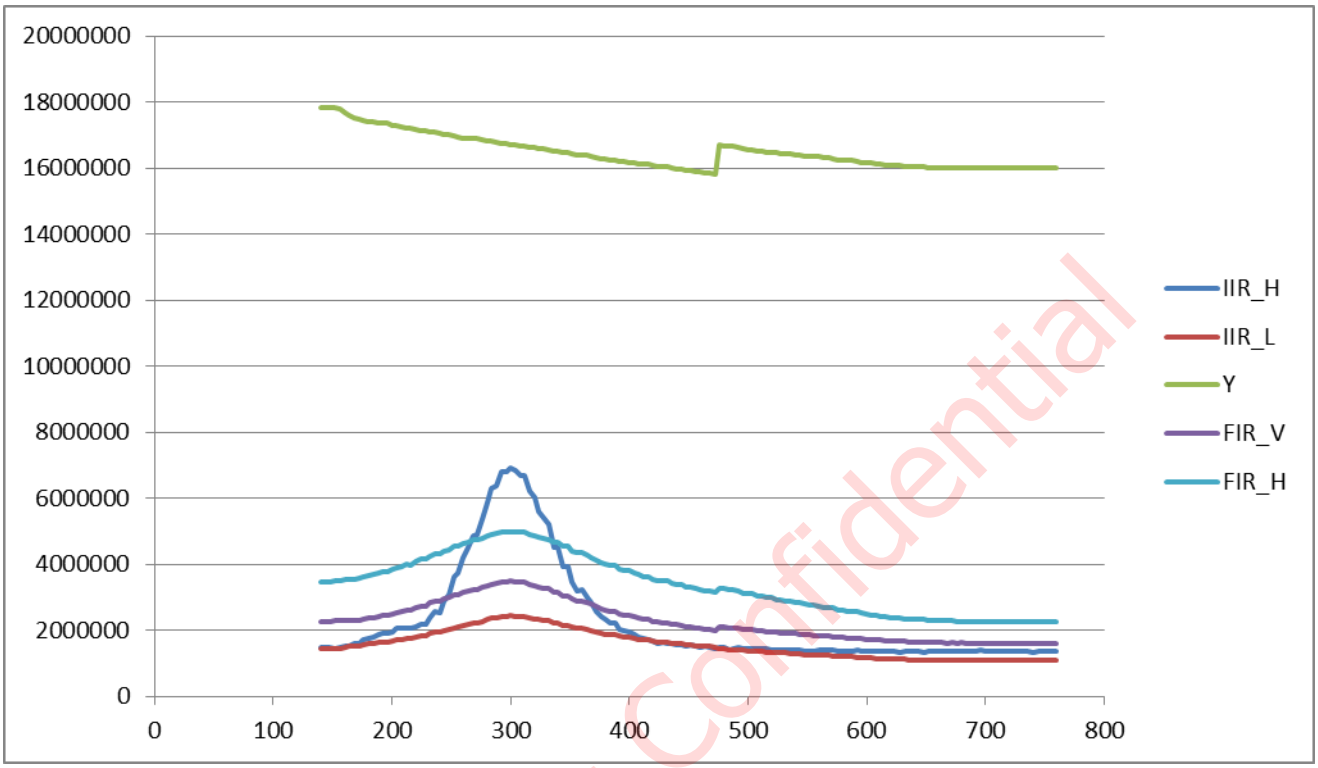
參數名稱	描述	輸出/輸入
IIR_1	分區間統計的 AF 水準方向 IIR 濾波器統計值	輸出
IIR_2	分區間統計的 AF 水準方向 IIR 濾波器統計值	輸出
Luma	分區間統計的 AF 亮度統計值	輸出
FIR_H	分區間統計的 AF 水準方向 FIR 濾波器統計值	輸出
FIR_V	分區間統計的 AF 垂直方向 FIR 濾波器統計值	輸出
YSat	分區間統計的 AF FIR 濾波器，大於 YThd 的統計值個數	輸出

資料範例:

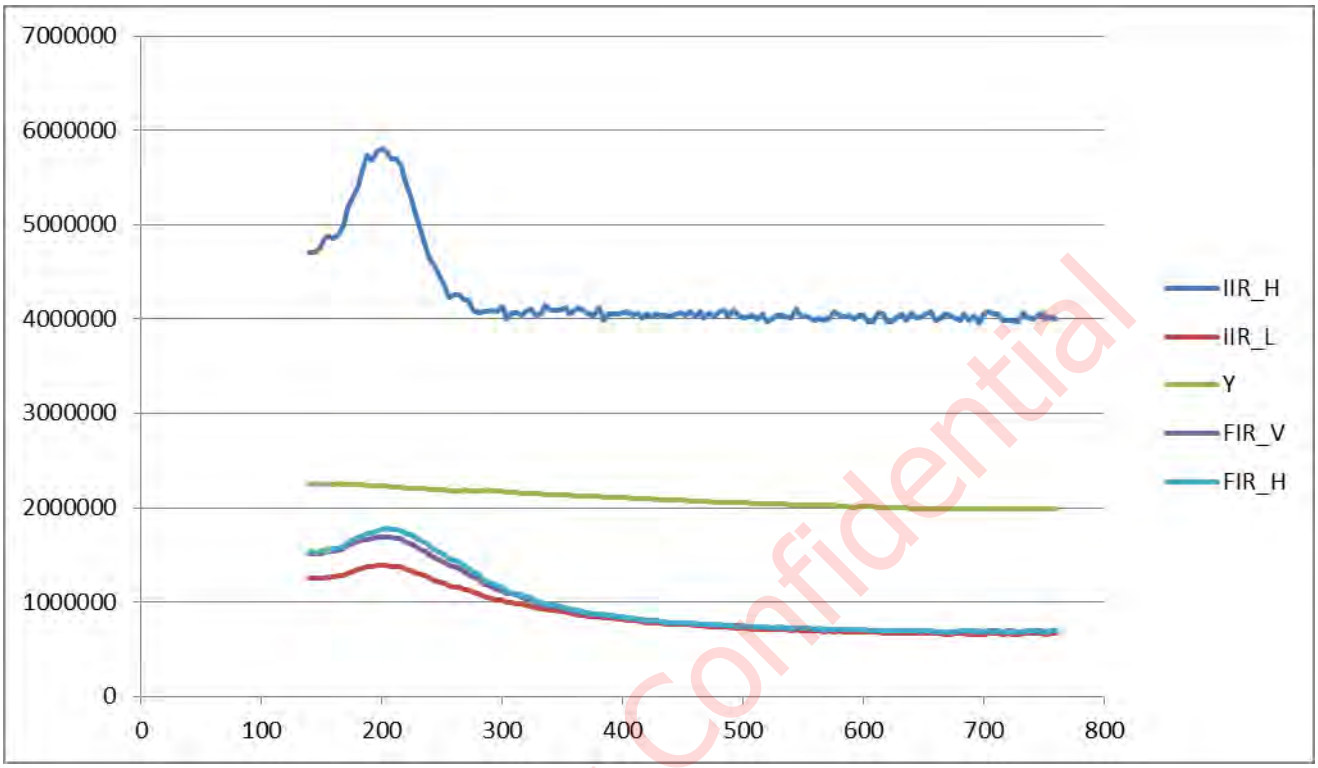
一般場景遠焦:



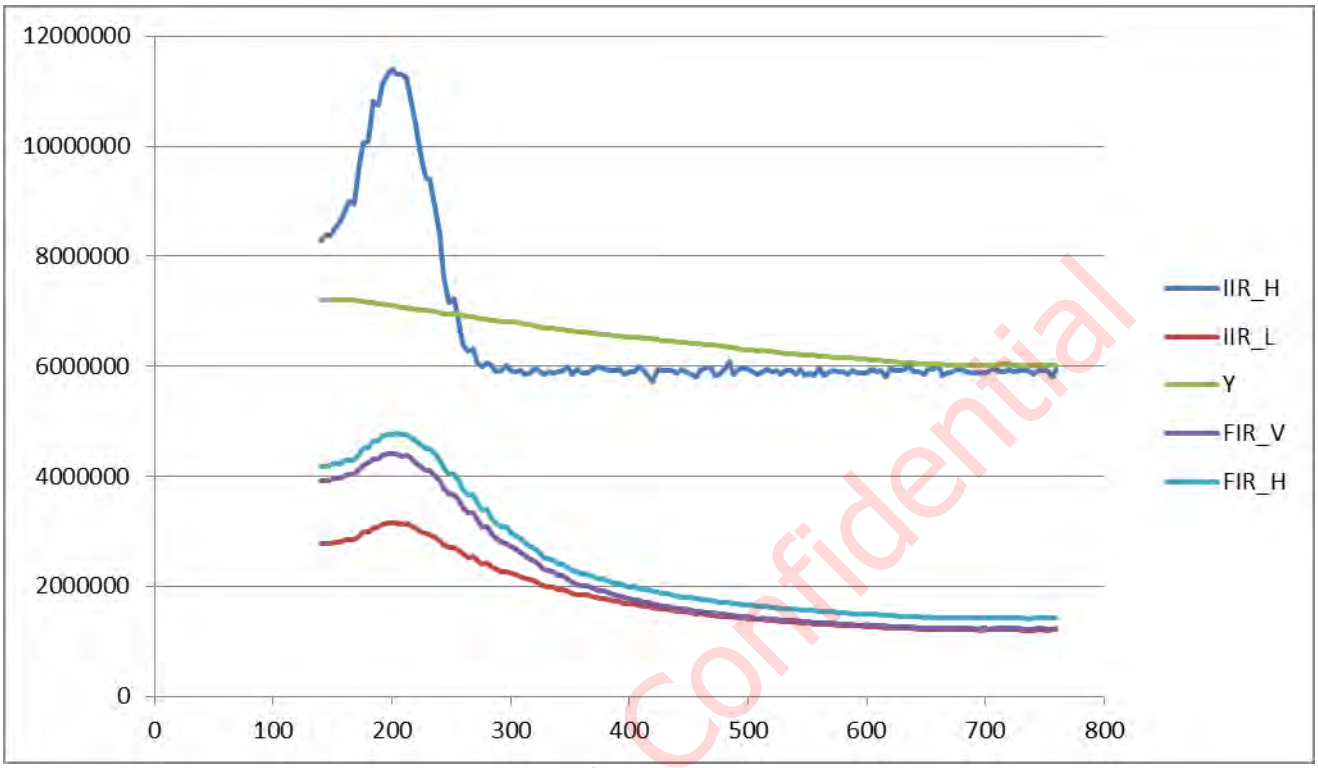
一般場景近焦:



低亮度場景遠焦:



低亮度場景遠焦+燈源:



5. AE RESULT

當 ISP 呼叫 AE 演算法 run()後，AE 演算法必須提供相對應的結果給 ISP，格式如下。

```
/*! @brief ISP ae algorithm result*/
typedef struct
{
    U32 Size;       /**< struct size*/
    U32 Change;    /**< if true, apply this result to hw register*/
    U32 Shutter;    /**< Shutter in us */
    U32 SensorGain; /**< Sensor gain, 1X = 1024 */
    U32 IspGain;    /**< ISP gain, 1X = 1024 */
    U32 ShutterHdrShort;    /**< Shutter in us */
    U32 SensorGainHdrShort; /**< Sensor gain, 1X = 1024 */
    U32 IspGainHdrShort;    /**< ISP gain, 1X = 1024 */
    U32 u4BVx16384; /**< Bv * 16384 in APEX system, EV = Av + Tv = Sv + Bv */
    U32 AvgY;       /**< frame brightness */
    U32 HdrRatio;   /**< hdr ratio, 1X = 1024 */
}__attribute__((packed, aligned(1))) ISP_AE_RESULT;
```

Size：ISP_AE_RESULT 長度，用於處理版本向下相容問題。
Change: 若該值為 1，ISP 將此次設定值寫入硬體，若為 0 ISP 會忽略此次的結果。
Shutter: 設定 Sensor 曝光時間。
SensorGain: 設定 Sensor gain，Digital/Analog gain 分配率由 Sensor 驅動決定。
IspGain: 設定 ISP digital gain。
ShutterHdrShort: 設定短曝 Sensor 曝光時間。
SensorGainHdrShort: 設定短曝 Sensor gain，Digital/Analog gain 分配率由 Sensor 驅動決定。
IspGainHdrShort: 設定短曝 ISP digital gain。
u4BVx16384: 目前場景的亮度值，此處計算公式比照 APEX。此設定值將會被 IQ 演算法參考。
AvgY：此張影像的平均灰階亮度，此設定值將會被 IQ 演算法參考。
HdrRatio：此張影像的 HDR 長短曝比例，此設定值將會被 IQ 演算法參考。

6. AWB RESULT

當 ISP 呼叫 AWB 演算法 run()後，AWB 演算法必須提供相對應的結果給 ISP，格式如下。

```
/*! @brief AWB algorithm result*/
typedef struct
{
    U32 Size;       /**< struct size*/
    U32 Change;    /**< if true, apply this result to hw register*/
    U32 R_gain;     /**< AWB gain for R channel*/
    U32 G_gain;     /**< AWB gain for G channel*/
    U32 B_gain;     /**< AWB gain for B channel*/
    U32 ColorTmp;   /**< Return color temperature*/
}ISP_AWB_RESULT;
```

Size：ISP_AWB_RESULT 長度，用於處理版本向下相容問題。
Change: 若該值為 1，ISP 將此次設定值寫入硬體，若為 0，ISP 會忽略此次的結果。
R_gain: R channel 增益。
G_gain: G channel 增益。
B_gain: B channel 增益。
ColorTmp: 目前場景色溫。

7. AF RESULT

當 ISP 呼叫 AF 演算法 run()後，AF 演算法必須提供相對應的結果給 ISP，格式如下。

```
/*! @brief AF algorithm result*/
typedef struct
{
    U32 Change;    /**< if true, apply this result to hw*/
    U32 Focal_pos; /**< Focal position*/
}__attribute__((packed, aligned(1))) ISP_AF_RESULT;
```

Change: 若該值為 1，通知 AF 演算出的焦距(Focal_pos)生效
Focal_pos: AF 演算法算出的焦距結果。

8. AE,AWB,AF 庫介面

*int CUS3A_RegInterface(U32 nCh, ISP_AE_INTERFACE *pAe, ISP_AWB_INTERFACE *pAwb, ISP_AF_INTERFACE *pAf)*

*int CUS3A_AERegInterface(u32 nCh,ISP_AE_INTERFACE *pAE);*
*int CUS3A_AWBRegInterface(u32 nCh,ISP_AWB_INTERFACE *pAWB);*
*int CUS3A_AFRegInterface(u32 nCh,ISP_AF_INTERFACE *pAF);*

說明:
ISP 提供 AE,AWB,AF 庫註冊介面
可使用 *CUS3A_RegInterface* 一次註冊 AE, AWB, AF
或使用 *CUS3A_AERegInterface*、*CUS3A_AWBRegInterface*、*CUS3A_AFRegInterface* 來分別註冊 AE, AWB, AF

參數:

參數名稱	描述	輸出/輸入
nCh	nCh = 0	輸入
pAe	演算法實作端提供 AE 庫指針，若不需註冊，則填 NULL	輸入
pAwb	演算法實作端提供 AWB 庫指針，若不需註冊，則填 NULL	輸入
pAf	演算法實作端提供 AF 庫指針，若不需註冊，則填 NULL	輸入

返回值:

返回值	描述
0	註冊成功
<0	註冊失敗

9. AE 回檔介面

*int (*init)(void* pdata,ISP_AE_INIT_PARAM *init_state)*

說明:
ISP 透過此介面通知 AE 庫進行初始化

參數:

參數名稱	描述	輸出/輸入
pdata	演算法私有資料指標	輸入
init_state	AE 初始值	輸入

返回值:

返回值	描述
0	初始化成功
-1	初始化失敗

*void (*release)(void* pdata)*

說明:
ISP 透過此介面通知 AE 庫進行釋放程式

參數:

參數名稱	描述	輸出/輸入
pdata	演算法私有資料指標	輸入

*void (*run)(void* pdata,const ISP_AE_INFO *info,ISP_AE_RESULT *result)*

說明:
ISP 透過此介面呼叫 AE 庫進行運算

參數:

參數名稱	描述	輸出/輸入
pdata	演算法私有資料指標	輸入
info	ISP AE 統計值資料結構指標	輸入
result	ISP AE 計算結果資料結構指標	輸出

*int (*ctrl)(void* pdata,ISP_AE_CTRL_CMD cmd,void* param)*

說明:
ISP 透過此介面控制 AE 參數

參數:

參數名稱	描述	輸出/輸入
pdata	演算法私有資料指標	輸入
cmd	AE 控制指令	輸入
param	AE 控制指令參數	輸入

10. AE 資料結構

ISP_AE_INTERFACE

說明:

AE 庫註冊結構體，演算法需實作並且填寫 init,release,run,ctrl 回檔指針，供 ISP 呼叫。

宣告:

```
/**@brief ISP AE interface*/
typedef struct
{
    void *pdata; /**< Private data for AE algorithm.*/
    int (*init)(void* pdata, ISP_AE_INIT_PARAM *init_state);
    void (*release)(void* pdata);
    void (*run)(void* pdata, const ISP_AE_INFO *info, ISP_AE_RESULT *result);
    int (*ctrl)(void* pdata, ISP_AE_CTRL_CMD cmd, void* param);
} ISP_AE_INTERFACE;
```

成員:

名稱	描述
pdata	AE 庫私有數據指標
init	AE 庫初始化回檔指標
release	AE 庫釋放回檔指針
run	AE 庫計算回檔指標，當 ISP 收到一偵影像統計值時，透過該指標要求 AE 演算法進行計算。
ctrl	AE 庫控制回檔指標

ISP_AE_INIT_PARAM

說明:
ISP 透過此結構，告知 AE 演算法相關硬體初始狀態

宣告:

```
typedef struct
{
    U32 Size;                 /**< struct size*/
    char sensor_id[32];           /**< sensor module id*/
    U32 shutter;              /**< shutter Shutter in us*/
    U32 shutter_step;          /**< shutter Shutter step ns*/
    U32 shutter_min;           /**< shutter Shutter min us*/
    U32 shutter_max;           /**< shutter Shutter max us*/
    U32 sensor_gain;           /**< sensor_gain Sensor gain, 1X = 1024*/
    U32 sensor_gain_max;       /**< sensor_gain_max Maximum Sensor gain, 1X = 1024*/
    U32 isp_gain;             /**< isp_gain Isp digital gain , 1X = 1024 */
    U32 isp_gain_max;          /**< isp_gain Maximum Isp digital gain , 1X = 1024 */
    U32 FNx10;                /**< F number * 10*/
    U32 fps;                  /**< initial frame per second*/
}ISP_AE_INIT_PARAM;
```

成員:

參數名稱	描述	輸出/輸入
Size	ISP_AE_INIT_PARAM 結構長度	輸入
sensor_id[32]	Image sensor 名稱	輸入
shutter	曝光時間初始值 us	輸入
shutter_step	曝光時間步距 ns	輸入
shutter_min	最小曝光時間 us	輸入
shutter_max	最長曝光時間 us	輸入
sensor_gain	Sensor 增益初始值	輸入
sensor_gain_max	Sensor 增益最大值	輸入
isp_gain	Isp 增益初始值	輸入
isp_gain_max	Isp 增益最大值	輸入
FNx10	光圈值*10	輸入
fps	每秒影像偵數	輸入

ISP_AE_INFO

說明:
ISP AE 硬體統計值

宣告:

```
/*! @brief ISP report to AE, hardware statistic */
typedef struct
{
    U32 Size;       /**< struct size*/
    ISP_HIST  *hist1;   /**< HW statistic histogram 1*/
    ISP_HIST  *hist2;   /**< HW statistic histogram 2*/
    U32 AvgBlkX;  /**< HW statistics average block number*/
    U32 AvgBlkY;  /**< HW statistics average block number*/
    ISP_AE_SAMPLE *avgs; /**< HW statistics average block data*/
    U32 Shutter;                    /**< Current shutter in us*/
    U32 SensorGain;                 /**< Current Sensor gain, 1X = 1024 */
    U32 IspGain;                    /**< Current ISP gain, 1X = 1024*/
    U32 ShutterHDRShort;            /**< Current shutter in us*/
    U32 SensorGainHDRShort;         /**< Current Sensor gain, 1X = 1024 */
    U32 IspGainHDRShort;            /**< Current ISP gain, 1X = 1024*/
} __attribute__((packed, aligned(1))) ISP_AE_INFO;
```

成員:

參數名稱	描述	輸出/輸入
Size	ISP_AE_INFO 結構長度	輸入
hist1	Histogram 1	輸入
hist2	Histogram 2 (Macaron not support)	輸入
AvgBlkX	橫向切割數量	輸入
AvgBlkY	縱向切割數量	輸入
avgs	區塊亮度統計值	輸入
Shutter	統計值發生時的曝光時間 us	輸入
SensorGain	統計值發生時的 Sensor Gain 1X=1024	輸入
IspGain	統計值發生時的 Isp Gain 1X=1024	輸入
ShutterHDRShort	統計值發生時的曝光時間 us (HDR short)	輸入
SensorGainHDRShort	統計值發生時的 Sensor Gain 1X=1024 (HDR short)	輸入
IspGainHDRShort	統計值發生時的 Isp Gain 1X=1024 (HDR short)	輸入

ISP_AE_RESULT

說明:
ISP AE 演算法計算結果

宣告:

```
/*! @brief ISP ae algorithm result*/
typedef struct
{
      U32 Size;        /**< struct size*/
      U32 Change;           /**< if true, apply this result to hw register*/
      U32 Shutter;         /**< Shutter in us */
      U32 SensorGain;      /**< Sensor gain, 1X = 1024 */
      U32 IspGain;         /**< ISP gain, 1X = 1024 */
      U32 ShutterHdrShort;     /**< Shutter in us */
      U32 SensorGainHdrShort;  /**< Sensor gain, 1X = 1024 */
      U32 IspGainHdrShort;     /**< ISP gain, 1X = 1024 */
      U32 u4BVx16384;      /**< Bv * 16384 in APEX system, EV = Av + Tv = Sv + Bv */
      U32 AvgY;           /**< frame brightness */
      U32 HdrRatio;   /**< hdr ratio, 1X = 1024 */
}__attribute__((packed, aligned(1))) ISP_AE_RESULT;
```

成員:

參數名稱	描述	輸出/輸入
Size	ISP_AE_RESULT 結構長度	輸入
Change	通知 ISP 此次計算結果是否要套用到硬體	輸出
Shutter	AE 演算法回報曝光時間 us	輸出
SensorGain	AE 演算法回報 Sensor 增益 1X=1024	輸出
IspGain	AE 演算法回報 ISP 增益 1X=1024	輸出
ShutterHdrShort	AE 演算法回報曝光時間 us (HDR Short)	輸出
SensorGainHdrShort	AE 演算法回報 Sensor 增益 1X=1024 (HDR Short)	輸出
IspGainHdrShort	AE 演算法回報 ISP 增益 1X=1024 (HDR Short)	輸出
u4Bvx16384	AE 演算法回報目前場景亮度估測值	輸出
AvgY	AE 演算法回報目前影像平均亮度	輸出
HdrRatio	AE 演算法回報目前影像 HDR 長短曝比例 1x=1024	輸出

ISP_AE_CTRL_CMD

說明:

AE 控制指令，ISP 透過回檔介面

int (*ctrl)(void* pdata,ISP_AE_CTRL_CMD cmd,void* param)

設定 AE 參數，參數 param 型態隨不同的控制命令改變。

宣告:

```
typedef enum
{
    ISP_AE_FPS_SET, /**< ISP notify AE sensor FPS has changed*/
}ISP_AE_CTRL_CMD;
```

數值:

名稱	描述	參數型態
ISP_AE_FPS_SET	ISP 通知 AE 庫目前的 FPS 更動	U32

11. AWB 回檔介面

*int (*init)(void* pdata)*

說明:
ISP 透過此介面通 AWB 庫進行初始化

參數:

參數名稱	描述	輸出/輸入
pdata	演算法私有資料指標	輸入

返回值:

返回值	描述
0	初始化成功
-1	初始化失敗

*void (*release)(void* pdata)*

說明:
ISP 透過此介面通知 AWB 庫進行釋放程式

參數:

參數名稱	描述	輸出/輸入
pdata	演算法私有資料指標	輸入

*void (*run)(void* pdata,const ISP_AWB_INFO *awb_info,const ISP_AE_INFO *ae_info,ISP_AWB_RESULT *result)*

說明:
ISP 透過此介面呼叫 AWB 庫進行運算

參數:

參數名稱	描述	輸出/輸入
pdata	演算法私有資料指標	輸入
awb_info	ISP AWB 統計值資料結構指標	輸入
ae_info	ISP AE 統計值資料結構指標	輸入
result	ISP AWB 計算結果資料結構指標	輸出

*int (*ctrl)(void* pdata,ISP_AWB_CTRL_CMD cmd,void* param)*

說明:
ISP 透過此介面控制 AWB 參數

返回值:

返回值	描述
0	控制指令成功
<0	註冊失敗

參數:

參數名稱	描述	輸出/輸入
pdata	演算法私有資料指標	輸入
cmd	AWB 控制指令	輸入
param	AWB 控制指令參數	輸入

12. AWB 資料結構

ISP_AWB_INTERFACE

說明:
AWB 庫註冊結構體，演算法需實作並且填寫 init,release,run,ctrl 回檔指針，供 ISP 呼叫。

宣告:

```
typedef struct
{
    void *pdata; /**< Private data for AE algorithm.*/
    int (*init)(void *pdata);
    void (*release)(void *pdata);
    void (*run)(void *pdata, const ISP_AWB_INFO *awb_info, const ISP_AE_INFO *ae_info,
ISP_AWB_RESULT *result);
    int (*ctrl)(void *pdata, ISP_AWB_CTRL_CMD cmd, void* param);
} ISP_AWB_INTERFACE
```

成員:

名稱	描述
pdata	AWB 庫私有數據指標
init	AWB 庫初始化回檔指標
release	AWB 庫釋放回檔指針
run	AWB 庫計算回檔指標，當 ISP 收到一偵影像統計值時，透過該指標要求 AWB 演算法進行計算。
ctrl	AWB 庫控制回檔指標

ISP_AWB_INFO

說明:
ISP AWB 硬體統計值

宣告:

```
/*! @brief AWB HW statistics data*/
typedef struct
{
    U32 AvgBlkX;
    U32 AvgBlkY;
    U32 CurRGain;
    U32 CurGGain;
    U32 CurBGain;
    ISP_AWB_SAMPLE *avgs;
}__attribute__((packed, aligned(1))) ISP_AWB_INFO;
```

成員:

參數名稱	描述	輸出/輸入
AvgBlkX	橫向切割數量	輸入
AvgBlkY	縱向切割數量	輸入
avgs	ISP AWB 區塊 RGB 統計值	輸入

ISP_AWB_RESULT

說明:
ISP AWB 演算法計算結果

宣告:

```
/*! @brief AWB algorithm result*/
typedef struct
{
    U32 Size;       /**< struct size*/
    U32 Change;    /**< if true, apply this result to hw register*/
    U32 R_gain;     /**< AWB gain for R channel*/
    U32 G_gain;     /**< AWB gain for G channel*/
    U32 B_gain;     /**< AWB gain for B channel*/
    U32 ColorTmp;   /**< Return color temperature*/
} ISP_AWB_RESULT;
```

成員:

參數名稱	描述	輸出/輸入
Size	ISP_AWB_RESULT 結構長度	輸入
Change	通知 ISP 此次計算結果是否要套用到硬體	輸出
R_gain	AWB 演算法回報 AWB R 增益 1X = 1024	輸出
G_gain	AWB 演算法回報 AWB G 增益 1X = 1024	輸出
B_gain	AWB 演算法回報 AWB B 增益 1X = 1024	輸出
ColorTmp	AWB 演算法回報目前場景色溫 K	輸出

ISP_AWB_CTRL_CMD

說明:

AWB 控制指令，ISP 透過回檔介面

int (*ctrl)(void* pdata,ISP_AWB_CTRL_CMD cmd,void* param)

設定 AE 參數，參數 param 型態隨不同的控制命令改變。

宣告:

```
typedef enum
{
    ISP_AWB_MODE_SET,
}ISP_AWB_CTRL_CMD;
```

數值:

名稱	描述	參數型態
ISP_AWB_MODE_SET	ISP 通知 AWB 庫目前的 AWB 模式更動	U32

13. AF 回檔介面

*int (*init)(void* pdata, ISP_AF_INIT_PARAM *param)*

說明:
ISP 透過此介面通 AF 庫進行初始化

參數:

參數名稱	描述	輸出/輸入
pdata	演算法私有資料指標	輸入
param	AF 初始設定	輸入/出

返回值:

返回值	描述
0	初始化成功
-1	初始化失敗

*void (*release)(void* pdata)*

說明:
ISP 透過此介面通知 AF 庫進行釋放程式

參數:

參數名稱	描述	輸出/輸入
pdata	演算法私有資料指標	輸入

*void (*run)(void* pdata,const ISP_AF_INFO *af_info ,ISP_AF_RESULT *result)*

說明:
ISP 透過此介面呼叫 AF 庫進行運算

參數:

參數名稱	描述	輸出/輸入
pdata	演算法私有資料指標	輸入
af_info	ISP AF 統計值資料結構指標	輸入
result	ISP AF 計算結果資料結構指標	輸出

*int (*ctrl)(void* pdata,ISP_AF_CTRL_CMD cmd,void* param) (Reserve)*

說明:
ISP 透過此介面控制 AF 參數

返回值:

返回值	描述
0	控制指令成功
<0	註冊失敗

參數:

參數名稱	描述	輸出/輸入
pdata	演算法私有資料指標 (Reserve)	輸入
cmd	AF 控制指令 (Reserve)	輸入
param	AF 控制指令參數 (Reserve)	輸入

14. AF 資料結構

ISP_AF_INTERFACE

說明:

AF 庫註冊結構體，演算法需實作並且填寫 init,release,run,ctrl 回檔指針，供 ISP 呼叫。

宣告:

```
typedef struct isp_af_interface
{
    void *pdata; /**< Private data for AF algorithm.*/
    int (*init)(void *pdata);
    void (*release)(void *pdata);
    void (*run)(void *pdata,const ISP_AF_INFO *af_info, ISP_AF_RESULT *result);
    int (*ctrl)(void *pdata,ISP_AF_CTRL_CMD cmd,void* param);
}ISP_AF_INTERFACE;
```

成員:

名稱	描述
pdata	AF 庫私有數據指標
init	AF 庫初始化回檔指標
release	AF 庫釋放回檔指針
run	AF 庫計算回檔指標，當 ISP 收到一偵影像統計值時，透過該指標要求 AF 演算法進行計算。
ctrl	AF 庫控制回檔指標

ISP_AF_INFO

說明:
ISP AF 硬體統計值

宣告:

```
/*! @brief AF HW statistics data*/
typedef struct
{
        U32 Size;          /**< struct size*/
        ISP_AF_STATS af_stats; /**< AF statistic*/
}   attribute   ((packed, aligned(1))) ISP_AF_INFO;
```

成員:

參數名稱	描述	輸出/輸入
Size	ISP_AF_INFO 資料結構長度	輸入
af_stats	ISP AF 統計值	輸出

ISP_AF_STATS

說明:
ISP AF 硬體統計值

宣告:

```
typedef struct{
        U32 iir_1[16]; /*AF horizontal IIR filter statistic*/
        U32 :32;
        U32 :32;
        U32 iir_2[16]; /*AF horizontal IIR filter statistic*/
        U32 :32;
        U32 :32;
        U32 luma[16]; /*AF luma statistic*/
        U32 :32;
        U32 :32;
        U32 FIR_v[16]; /*AF FIR vertical filter statistic*/
        U32 :32;
        U32 :32;
        U32 FIR_h[16]; /*AF FIR horizontal filter statistic*/
        U32 :32;
        U32 :32;
}ISP_AF_STATS;
```

成員:

參數名稱	描述	輸出/輸入
IIR_1	分區間統計的 AF 水準方向 IIR 濾波器統計值	輸出
IIR_2	分區間統計的 AF 水準方向 IIR 濾波器統計值	輸出
Luma	分區間統計的 AF 亮度統計值	輸出
FIR_h	分區間統計的 AF 水準方向 FIR 濾波器統計值	輸出
FIR_v	分區間統計的 AF 垂直方向 FIR 濾波器統計值	輸出
YSat	分區間統計的 AF FIR 濾波器，大於 YThd 的統計值個數	輸出

* IIR_1 default 為 IIR Low，IIR_2 default 為 IIR High
* 每個 filter 都是 16 組，前面九個預設對應畫面的九宮格分割位置，第 16 個是自訂使用的，通常是拿來做 touch AF。
* IIR FIR 數值越高代表畫面就越清晰。Luma 表示亮度。

使用參考:

區間加權權重: 建議"中央權重"。

統計值: 建議常時使用 IIR_1 (IIR Low)。
IIR_2 (IIR High) 在清晰時會有明顯變化，但模糊時幾乎不變動，適合做最清晰位置判斷。
FIR 會有偏性，僅適合做強度變化參考。

ISP_AF_RESULT

說明:

ISP AF 演算法計算結果

宣告:

```
/*! @brief AF algorithm result*/
typedef struct
{
    U32 Change;     /**< if true, apply this result to hw*/
    U32 Focal_pos; /**< Focal position*/
}__attribute__((packed, aligned(1))) ISP_AF_RESULT;
```

成員:

參數名稱	描述	輸出/輸入
Change	通知 ISP 此次計算結果是否要套用到硬體	輸出
Focal_pos	AF 演算法演算出的對焦 position	輸出

15. SAMPLE CODE

```
#include <mi_isp.h>

//// AE INTERFACE TEST ////

int ae_init(void* pdata, ISP_AE_INIT_PARAM *init_state)
{
    printf("******
ae_init ,shutter=%d,shutter_step=%d,sensor_gain_min=%d,sensor_gain_max=%d *******\n",
          (int)init_state->shutter,
          (int)init_state->shutter_step,
          (int)init_state->sensor_gain,
          (int)init_state->sensor_gain_max
         );
    return 0;
}

void ae_run(void* pdata, const ISP_AE_INFO *info, ISP_AE_RESULT *result)
{
#define log_info 1

    // Only one can be chosen (the following three define)
#define shutter_test 0
#define gain_test 0
#define AE_sample 1

#if (shutter_test) || (gain_test)
    static int AE_period = 4;
#endif
    static unsigned int fcount = 0;
    unsigned int max = info->AvgBlkY * info->AvgBlkX;
    unsigned int avg = 0;
    unsigned int n;
#if gain_test
    static int tmp = 0;
    static int tmp1 = 0;
#endif

    result->Change             = 0;
    result->u4BVx16384         = 16384;
    result->HdrRatio           = 10; //infinity5 //TBD //10 * 1024;   //user define hdr
exposure ratio
    result->IspGain            = 1024;
    result->SensorGain         = 4096;
    result->Shutter            = 20000;
    result->IspGainHdrShort    = 1024;
    result->SensorGainHdrShort = 1024;
    result->ShutterHdrShort    = 1000;
    //result->Size         = sizeof(CusAEResult_t);
```

```
    for(n = 0; n < max; ++n)
    {
        avg += info->avgs[n].y;
    }
    avg /= max;

    result->AvgY        = avg;

#if shutter_test // shutter test under constant sensor gain
    int Shutter_Step = 100; //per frame
    int Shutter_Max = 33333;
    int Shutter_Min = 150;
    int Gain_Constant = 10240;

    result->SensorGain = Gain_Constant;
    result->Shutter = info->Shutter;

    if(++fcount % AE_period == 0)
    {
        if (tmp == 0)
        {
            result->Shutter = info->Shutter + Shutter_Step * AE_period;
            //printf("[shutter-up] result->Shutter = %d \n", result->SensorGain);
        }
        else
        {
            result->Shutter = info->Shutter - Shutter_Step * AE_period;
            //printf("[shutter-down] result->Shutter = %d \n", result->SensorGain);
        }
        if (result->Shutter >= Shutter_Max)
        {
            result->Shutter = Shutter_Max;
            tmp = 1;
        }
        if (result->Shutter <= Shutter_Min)
        {
            result->Shutter = Shutter_Min;
            tmp = 0;
        }
    }
#if log_info
    printf("fcount = %d, Image avg = 0x%X \n", fcount, avg);
    printf("tmp = %d, Shutter: %d -> %d \n", tmp, info->Shutter, result->Shutter);
#endif
#endif

#if gain_test // gain test under constant shutter
    int Gain_Step = 1024; //per frame
    int Gain_Max = 1024 * 100;
    int Gain_Min = 1024 * 2;
    int Shutter_Constant = 20000;

    result->SensorGain = info->SensorGain;
    result->Shutter = Shutter_Constant;
```

```
    if(++fcount % AE_period == 0)
    {
        if (tmp1 == 0)
        {
            result->SensorGain = info->SensorGain + Gain_Step * AE_period;
            //printf("[gain-up] result->SensorGain = %d \n", result->SensorGain);
        }
        else
        {
            result->SensorGain = info->SensorGain - Gain_Step * AE_period;
            //printf("[gain-down] result->SensorGain = %d \n", result->SensorGain);
        }
        if (result->SensorGain >= Gain_Max)
        {
            result->SensorGain = Gain_Max;
            tmp1 = 1;
        }
        if (result->SensorGain <= Gain_Min)
        {
            result->SensorGain = Gain_Min;
            tmp1 = 0;
        }
    }
#if log_info
    printf("fcount = %d, Image avg = 0x%X \n", fcount, avg);
    printf("tmp = %d, SensorGain: %d -> %d \n", tmp, info->SensorGain, result-
>SensorGain);
#endif
#endif

#if AE_sample
    int y_lower = 0x28;
    int y_upper = 0x38;
    int change_ratio = 10; // percentage
    int Gain_Min = 1024 * 2;
    int Gain_Max = 1024 * 1000;
    int Shutter_Min = 150;
    int Shutter_Max = 33333;

    result->SensorGain = info->SensorGain;
    result->Shutter = info->Shutter;

    if(avg < y_lower)
    {
        if (info->Shutter < Shutter_Max)
        {
            result->Shutter = info->Shutter + (info->Shutter * change_ratio / 100);
            if (result->Shutter > Shutter_Max) result->Shutter = Shutter_Max;
        }
        else
        {
            result->SensorGain = info->SensorGain + (info->SensorGain * change_ratio /
100);
            if (result->SensorGain > Gain_Max) result->SensorGain = Gain_Max;
        }
        result->Change = 1;
```

```
    }
    else if(avg > y_upper)
    {
        if (info->SensorGain > Gain_Min)
        {
            result->SensorGain = info->SensorGain - (info->SensorGain * change_ratio /
100);
            if (result->SensorGain < Gain_Min) result->SensorGain = Gain_Min;
        }
        else
        {
            result->Shutter = info->Shutter - (info->Shutter * change_ratio / 100);
            if (result->Shutter < Shutter_Min) result->Shutter = Shutter_Min;
        }
        result->Change = 1;
    }

#if 0 //infinity5 //TBD
    //hdr demo code
    result->SensorGainHdrShort = result->SensorGain;
    result->ShutterHdrShort = result->Shutter * 1024 / result->HdrRatio;
#endif

#if log_info
    printf("fcount = %d, Image avg = 0x%X \n", fcount, avg);
    printf("SensorGain: %d -> %d \n", (int)info->SensorGain, (int)result->SensorGain);
    printf("Shutter: %d -> %d \n", (int)info->Shutter, (int)result->Shutter);
#endif

#endif
}

void ae_release(void* pdata)
{
    printf("************* ae_release *************\n");
}


//// AWB INTERFACE TEST ////

int awb_init(void *pdata)
{
    printf("************ awb_init **********\n");
    return 0;
}
```

```
void awb_run(void* pdata, const ISP_AWB_INFO *info, ISP_AWB_RESULT *result)
{
#define log_info 1

    static u32 count = 0;
    int avg_r = 0;
    int avg_g = 0;
    int avg_b = 0;
    int tar_rgain = 1024;
    int tar_bgain = 1024;
    int x = 0;
    int y = 0;

    result->R_gain = info->CurRGain;
    result->G_gain = info->CurGGain;
    result->B_gain = info->CurBGain;
    result->Change = 0;
    result->ColorTmp = 6000;

    if (++count % 4 == 0)
    {
        //center area YR/G/B avg
        for (y = 30; y < 60; ++y)
        {
            for (x = 32; x < 96; ++x)
            {
                avg_r += info->avgs[info->AvgBlkX * y + x].r;
                avg_g += info->avgs[info->AvgBlkX * y + x].g;
                avg_b += info->avgs[info->AvgBlkX * y + x].b;
            }
        }
        avg_r /= 30 * 64;
        avg_g /= 30 * 64;
        avg_b /= 30 * 64;

        if (avg_r < 1)
            avg_r = 1;
        if (avg_g < 1)
            avg_g = 1;
        if (avg_b < 1)
            avg_b = 1;

#if log_info
        printf("AVG R / G / B = %d, %d, %d \n", avg_r, avg_g, avg_b);
#endif

        // calculate Rgain, Bgain
        tar_rgain = avg_g * 1024 / avg_r;
        tar_bgain = avg_g * 1024 / avg_b;

        if (tar_rgain > info->CurRGain)
        {
            if (tar_rgain - info->CurRGain < 384)
                result->R_gain = tar_rgain;
            else
                result->R_gain = info->CurRGain + (tar_rgain - info->CurRGain) / 10;
```

```
        }
        else
        {
            if (info->CurRGain - tar_rgain < 384)
                result->R_gain = tar_rgain;
            else
                result->R_gain = info->CurRGain - (info->CurRGain - tar_rgain) / 10;
        }

        if (tar_bgain > info->CurBGain)
        {
            if (tar_bgain - info->CurBGain < 384)
                result->B_gain = tar_bgain;
            else
                result->B_gain = info->CurBGain + (tar_bgain - info->CurBGain) / 10;
        }
        else
        {
            if (info->CurBGain - tar_bgain < 384)
                result->B_gain = tar_bgain;
            else
                result->B_gain = info->CurBGain - (info->CurBGain - tar_bgain) / 10;
        }

        result->Change = 1;
        result->G_gain = 1024;

#if log_info
        printf("[current] r=%d, g=%d, b=%d \n", (int)info->CurRGain, (int)info->CurGGain,
(int)info->CurBGain);
        printf("[result] r=%d, g=%d, b=%d \n", (int)result->R_gain, (int)result->G_gain,
(int)result->B_gain);
#endif
    }
}

void awb_release(void *pdata)
{
    printf("************ awb_release **********\n");
}

//// AF INTERFACE TEST ////

int af_init(void *pdata, ISP_AF_INIT_PARAM *param)
{
    MI_U32 u32ch = 0;
    MI_U8 u8win_idx = 16;
    CusAFRoiMode_t taf_roimode;
    CusAFWin_t taf_win;

    printf("************ af_init **********\n");
#if 0
    //Init Normal mode setting
    taf_roimode.mode = AF_ROI_MODE_NORMAL;
    taf_roimode.u32_vertical_block_number = 1;
```

```
    MI_ISP_CUS3A_SetAFRoiMode(u32ch, &taf_roimode);

    static CusAFWin_t afwin[16] =
    {
        { 0, {   0,    0,  255,  255}},
        { 1, { 256,    0,  511,  255}},
        { 2, { 512,    0,  767,  255}},
        { 3, { 768,    0, 1023,  255}},
        { 4, {   0,  256,  255,  511}},
        { 5, { 256,  256,  511,  511}},
        { 6, { 512,  256,  767,  511}},
        { 7, { 768,  256, 1023,  511}},
        { 8, {   0,  512,  255,  767}},
        { 9, { 256,  512,  511,  767}},
        {10, { 512,  512,  767,  767}},
        {11, { 768,  512, 1023,  767}},
        {12, {   0,  768,  255, 1023}},
        {13, { 256,  768,  511, 1023}},
        {14, { 512,  768,  767, 1023}},
        {15, { 768,  768, 1023, 1023}}
    };
    for(u8win_idx = 0; u8win_idx < 16; ++u8win_idx){
        MI_ISP_CUS3A_SetAFWindow(u32ch, &afwin[u8win_idx]);
    }

#else
    //Init Matrix mode setting
    taf_roimode.mode = AF_ROI_MODE_MATRIX;
    taf_roimode.u32_vertical_block_number = 16; //16xN, N=16
    MI_ISP_CUS3A_SetAFRoiMode(u32ch, &taf_roimode);

    static CusAFWin_t afwin[16] =
    {
        #if 1
        //full image, equal divide to 16x16
        {0, {0, 0, 63, 63}},
        {1, {64, 64, 127, 127}},
        {2, {128, 128, 191, 191}},
        {3, {192, 192, 255, 255}},
        {4, {256, 256, 319, 319}},
        {5, {320, 320, 383, 383}},
        {6, {384, 384, 447, 447}},
        {7, {448, 448, 511, 511}},
        {8, {512, 512, 575, 575}},
        {9, {576, 576, 639, 639}},
        {10, {640, 640, 703, 703}},
        {11, {704, 704, 767, 767}},
        {12, {768, 768, 831, 831}},
        {13, {832, 832, 895, 895}},
        {14, {896, 896, 959, 959}},
        {15, {960, 960, 1023, 1023}}
        #else
        //use two row only => 16x2 win
        {0, {0, 0, 63, 63}},
        {1, {64, 64, 127, 127}},
        {2, {128, 0, 191, 2}},      //win2 v_str, v_end doesn't use, set to (0, 2)
```

```
        {3, {192, 0, 255, 2}},
        {4, {256, 0, 319, 2}},
        {5, {320, 0, 383, 2}},
        {6, {384, 0, 447, 2}},
        {7, {448, 0, 511, 2}},
        {8, {512, 0, 575, 2}},
        {9, {576, 0, 639, 2}},
        {10, {640, 0, 703, 2}},
        {11, {704, 0, 767, 2}},
        {12, {768, 0, 831, 2}},
        {13, {832, 0, 895, 2}},
        {14, {896, 0, 959, 2}},
        {15, {960, 0, 1023, 2}}
        #endif
    };

    for(u8win_idx = 0; u8win_idx < 16; ++u8win_idx){
        MI_ISP_CUS3A_SetAFWindow(u32ch, &afwin[u8win_idx]);
    }
#endif

    static CusAFFilter_t affilter =
    {
        //filter setting with sign value
        //{63, -126, 63, -109, 48, 0, 320, 0, 1023},
        //{63, -126, 63, 65, 55, 0, 320, 0, 1023}

        //convert to hw format (sign bit with msb)
        63, 126+1024, 63, 109+128, 48, 0, 320, 0, 1023,
        63, 126+1024, 63, 65, 55, 0, 320, 0, 1023,

    };

    MI_ISP_CUS3A_SetAFFilter(0, &affilter);
    return 0;
}

void af_run(void *pdata, const ISP_AF_INFO *af_info, ISP_AF_RESULT *result)
{
    #define af_log_info 1

#if af_log_info
    int i=0,x=0;

    printf("\n\n");

    //print row0 16wins
    x=0;
    for (i=0; i<16; i++){
        printf("[AF]win%d-%d iir0: 0x%02x%02x%02x%02x%02x, iir1:0x%02x%02x%02x%02x%02x,
luma:0x%02x%02x%02x%02x, sobelh:0x%02x%02x%02x%02x%02x, sobelv:0x%02x%02x%02x%02x%02x
ysat:0x%02x%02x%02x\n",
            x, i,
            af_info->af_stats.stParaAPI[x].high_iir[4+i*5],af_info-
>af_stats.stParaAPI[x].high_iir[3+i*5],af_info-
>af_stats.stParaAPI[x].high_iir[2+i*5],af_info-
```

```
>af_stats.stParaAPI[x].high_iir[1+i*5],af_info->af_stats.stParaAPI[x].high_iir[0+i*5],
            af_info->af_stats.stParaAPI[x].low_iir[4+i*5],af_info-
>af_stats.stParaAPI[x].low_iir[3+i*5],af_info-
>af_stats.stParaAPI[x].low_iir[2+i*5],af_info-
>af_stats.stParaAPI[x].low_iir[1+i*5],af_info->af_stats.stParaAPI[x].low_iir[0+i*5],
            af_info->af_stats.stParaAPI[x].luma[3+i*4],af_info-
>af_stats.stParaAPI[x].luma[2+i*4],af_info->af_stats.stParaAPI[x].luma[1+i*4],af_info-
>af_stats.stParaAPI[x].luma[0+i*4],
            af_info->af_stats.stParaAPI[x].sobel_h[4+i*5],af_info-
>af_stats.stParaAPI[x].sobel_h[3+i*5],af_info-
>af_stats.stParaAPI[x].sobel_h[2+i*5],af_info-
>af_stats.stParaAPI[x].sobel_h[1+i*5],af_info->af_stats.stParaAPI[x].sobel_h[0+i*5],
            af_info->af_stats.stParaAPI[x].sobel_v[4+i*5],af_info-
>af_stats.stParaAPI[x].sobel_v[3+i*5],af_info-
>af_stats.stParaAPI[x].sobel_v[2+i*5],af_info-
>af_stats.stParaAPI[x].sobel_v[1+i*5],af_info->af_stats.stParaAPI[x].sobel_v[0+i*5],
            af_info->af_stats.stParaAPI[x].ysat[2+i*3],af_info-
>af_stats.stParaAPI[x].ysat[1+i*3],af_info->af_stats.stParaAPI[x].ysat[0+i*3]
            );
    }

    //print row15 16wins
    x=15;
    for (i=0; i<16; i++){
        printf("[AF]win%d-%d iir0: 0x%02x%02x%02x%02x%02x, iir1:0x%02x%02x%02x%02x%02x,
luma:0x%02x%02x%02x%02x, sobelh:0x%02x%02x%02x%02x%02x, sobelv:0x%02x%02x%02x%02x%02x
ysat:0x%02x%02x%02x\n",
            x, i,
            af_info->af_stats.stParaAPI[x].high_iir[4+i*5],af_info-
>af_stats.stParaAPI[x].high_iir[3+i*5],af_info-
>af_stats.stParaAPI[x].high_iir[2+i*5],af_info-
>af_stats.stParaAPI[x].high_iir[1+i*5],af_info->af_stats.stParaAPI[x].high_iir[0+i*5],
            af_info->af_stats.stParaAPI[x].low_iir[4+i*5],af_info-
>af_stats.stParaAPI[x].low_iir[3+i*5],af_info-
>af_stats.stParaAPI[x].low_iir[2+i*5],af_info-
>af_stats.stParaAPI[x].low_iir[1+i*5],af_info->af_stats.stParaAPI[x].low_iir[0+i*5],
            af_info->af_stats.stParaAPI[x].luma[3+i*4],af_info-
>af_stats.stParaAPI[x].luma[2+i*4],af_info->af_stats.stParaAPI[x].luma[1+i*4],af_info-
>af_stats.stParaAPI[x].luma[0+i*4],
            af_info->af_stats.stParaAPI[x].sobel_h[4+i*5],af_info-
>af_stats.stParaAPI[x].sobel_h[3+i*5],af_info-
>af_stats.stParaAPI[x].sobel_h[2+i*5],af_info-
>af_stats.stParaAPI[x].sobel_h[1+i*5],af_info->af_stats.stParaAPI[x].sobel_h[0+i*5],
            af_info->af_stats.stParaAPI[x].sobel_v[4+i*5],af_info-
>af_stats.stParaAPI[x].sobel_v[3+i*5],af_info-
>af_stats.stParaAPI[x].sobel_v[2+i*5],af_info-
>af_stats.stParaAPI[x].sobel_v[1+i*5],af_info->af_stats.stParaAPI[x].sobel_v[0+i*5],
            af_info->af_stats.stParaAPI[x].ysat[2+i*3],af_info-
>af_stats.stParaAPI[x].ysat[1+i*3],af_info->af_stats.stParaAPI[x].ysat[0+i*3]
            );
    }
#endif}

void af_release(void *pdata)
{
    printf("************ af_release **********\n");
```

```
}

int main(int argc,char** argv)
{
    //TO DO: Register 3rd party AE/AWB library
    ISP_AE_INTERFACE tAeIf;
    ISP_AWB_INTERFACE tAwbIf;

    CUS3A_Init();

    /*AE*/
    tAeIf.ctrl = NULL;
    tAeIf.pdata = NULL;
    tAeIf.init = ae_init;
    tAeIf.release = ae_release;
    tAeIf.run = ae_run;

    /*AWB*/
    tAwbIf.ctrl = NULL;
    tAwbIf.pdata = NULL;
    tAwbIf.init = awb_init;
    tAwbIf.release = awb_release;
    tAwbIf.run = awb_run;

    CUS3A_RegInterface(0,&tAeIf,&tAwbIf,NULL);
    // or use below reginterface
    //CUS3A_AERegInterface(0,&tAeIf);
    //CUS3A_AWBRegInterface(0,&tAwbIf);

    //TO DO: Open MI
    MI_SYS_Init();
    while(1)
    {
    usleep(5000);
    }
}
```

16. MI_ISP 設置 AE BLOCK NUMBER

MI_ISP_CUS3A_SetAEWindowBlockNumber

【目的】
設定 AE 的區塊數量

【語法】
MI_RET MI_ISP_CUS3A_SetAEWindowBlockNumber(U32 nChannel,
MS_CUST_AE_WIN_BLOCK_NUM_TYPE_e eBlkNum);

【描述】
調用此介面設置 AE 區塊數量

【參數】

參數名稱	描述
nChannel	視訊輸入通道號碼(一般為 0)
eBlkNum	區塊個數，將整張照片切割成 X*Y 格

【返回值】

返回值	描述
MI_RET_SUCCESS	成功
MI_RET_FAIL	失敗

【需求】
header file：mi_isp_twinkie.h / mi_isp_pretzel.h / mi_isp.h (for Macaron)
.so：libmi_isp_twinkie.so / libmi_isp_pretzel.so / libmi_isp.so (for Macaron)

MS_CUST_AE_WIN_BLOCK_NUM_TYPE_e

【說明】AE區塊

【定義】

```
typedef enum {
AE_16x24 = 0,
AE_32x24,
AE_64x48,
AE_64x45,
AE_128x80,
AE_128x90
} AeWinBlockNum_e;
```

【成員】

參數名稱	描述
AE_16x24	畫面分割成16*24塊
AE_32x24	畫面分割成32*24塊
AE_64x48	畫面分割成64*48塊
AE_64x45	畫面分割成64*45塊
AE_128x80	畫面分割成128*80塊
AE_128x90	畫面分割成128*90塊

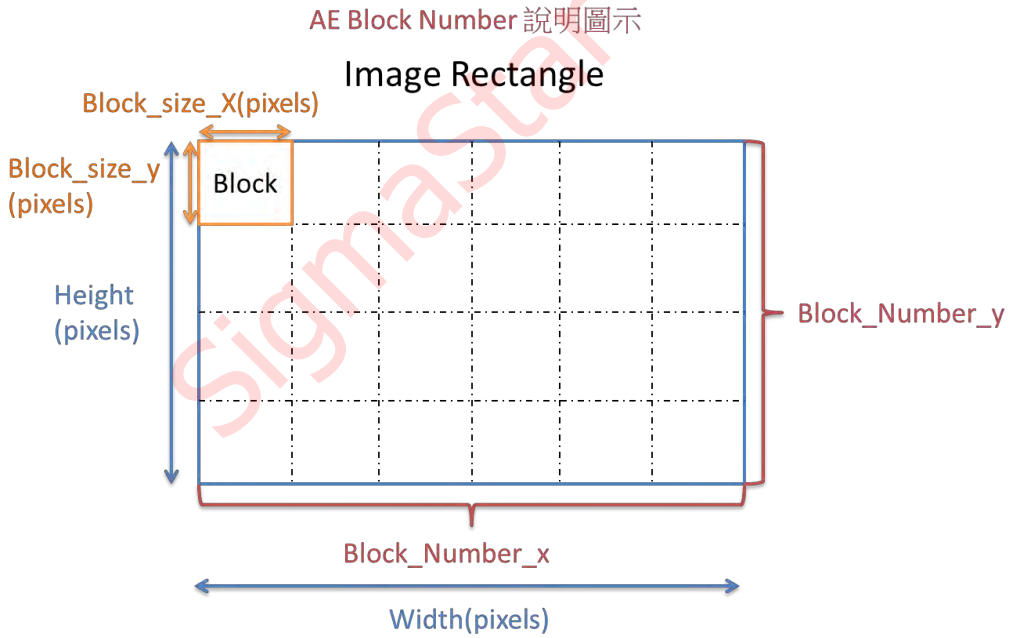


Image resolution = Width * Height
Block_size_X = Width / Block_Number_x
Block_size_Y = Height / Block_Number_y

【備註】

Macaron 不支援這支 API，畫面強制分割成 32*32 塊。

17. MI_ISP 設置 AE HISTOGRAM WINDOW

MI_ISP_CUS3A_SetAEHistogramWindow

【目的】
AE 亮度分佈統計值的視窗區域設定

【語法】
MI_RET MI_ISP_CUS3A_SetAEHistogramWindow(MI_U32 Channel, CusAEHistWin_t *data);

【描述】
調用此介面設置 AE 亮度分佈統計值的視窗區域位置及大小

【成員】

參數名稱	描述
nChannel	視訊輸入通道號碼(一般為 0)
data	亮度分佈統計值的視窗區域位置及大小，視窗編號(0 或 1)

【返回值】

返回值	描述
MI_RET_SUCCESS	成功
MI_RET_FAIL	失敗

【需求】
header file：mi_isp_twinkie.h / mi_isp_pretzel.h / mi_isp.h (for Macaron)
.so：libmi_isp_twinkie.so / libmi_isp_pretzel.so / libmi_isp.so (for Macaron)

【注意】
目前只能支援同時設定兩個視窗(0/1)。視窗區域可以重迭。

HistWin_t

【說明】AE 亮度分佈統計值的視窗區域設定

【定義】

```
typedef struct {
    MI_U16 u2Stawin_x_offset;
    MI_U16 u2Stawin_x_size;
    MI_U16 u2Stawin_y_offset;
    MI_U16 u2Stawin_y_size;
    MI_U16 u2WinIdx;
} CusAEHistWin_t;
```

【成員】

參數名稱	描述
u2Stawin_x_offset	窗口起始座標 x 軸偏移
u2Stawin_x_size	x 軸方向窗口大小
u2Stawin_y_offset	窗口起始座標 y 軸偏移
u2Stawin_y_size	y 軸方向窗口大小
u2WinIdx	窗口編號(0 或 1)

【參數範圍】

X 軸偏移：0~127
Y 軸偏移：0~89
X 軸(偏移+大小) <= 128
Y 軸(偏移+大小) <= 90

【備註】

在 Macaron 中，參數範圍如下。
X 軸偏移：0~31
Y 軸偏移：0~31
X 軸(偏移+大小) <= 32
Y 軸(偏移+大小) <= 32

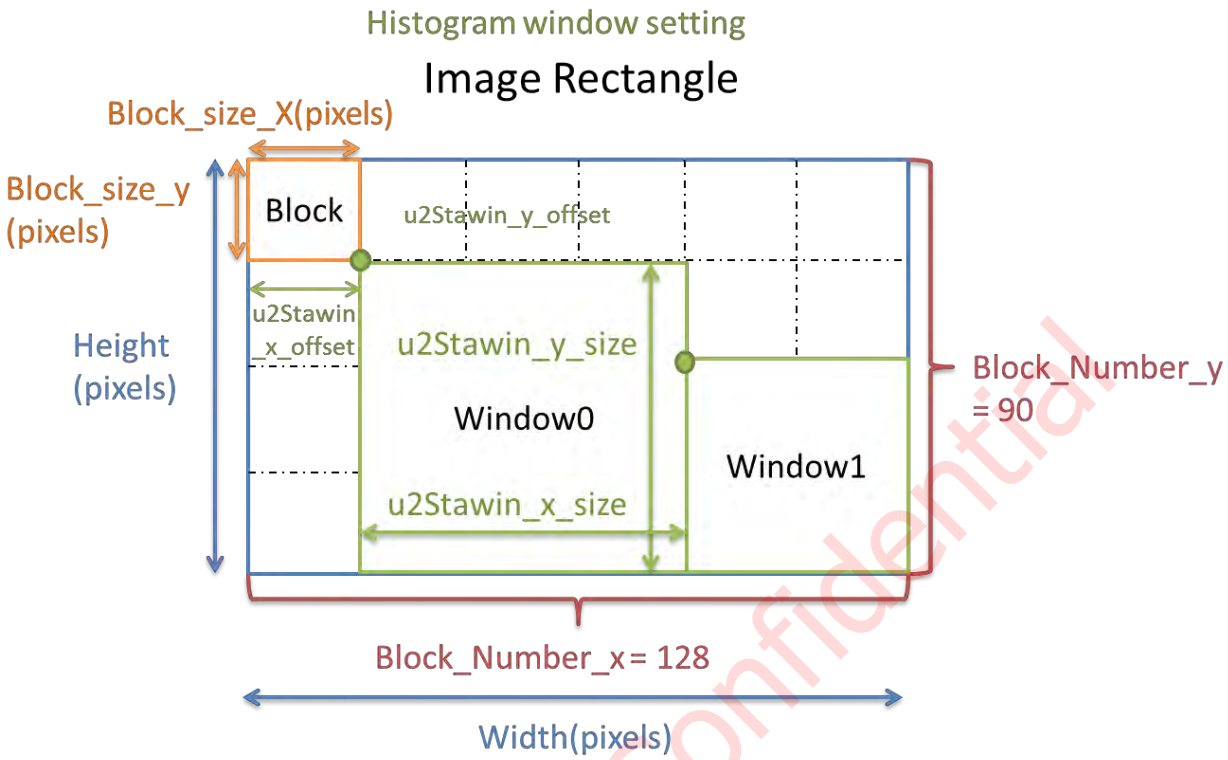


Image Size = Width * Height
Block_size_X = Width / 128
Block_size_Y = Height / 90

18. MI_ISP 設置 AWB SAMPLING

MI_ISP_CUS3A_SetAwbSampling

【目的】
設定 AWB 取樣大小

【語法】
MI_RET MI_ISP_CUS3A_SetAwbSampling(U32 nChannel, CusAWBSample_t *data)

【描述】
調用此介面設置 AWB 取樣大小

【成員】

參數名稱	描述
nChannel	視訊輸入通道號碼(一般為 0)
SizeX	X 軸上取多少個取樣點
SizeY	Y 軸上取多少個取樣點
IncRatio	對統計值乘上一個比例

【返回值】

返回值	描述
MI_RET_SUCCESS	成功
MI_RET_FAIL	失敗

【需求】
header file：mi_isp_twinkie.h / mi_isp_pretzel.h / mi_isp.h (for Macaron)
.so：libmi_isp_twinkie.so / libmi_isp_pretzel.so / libmi_isp.so (for Macaron)

【注意事項】
整張照片已經切割成 128*90 格，每個格子中取其中 SizeX * SizeY 個 pixels 做 AWB 取樣。
SizeX：SizeX >=4 && SizeX <= Block_size_X
SizeY：SizeY >=2 && SizeY <= Block_size_Y

當 AE 亮度很低的時候，可將統計值乘上一個比例(IncRatio)，避免統計值為 0

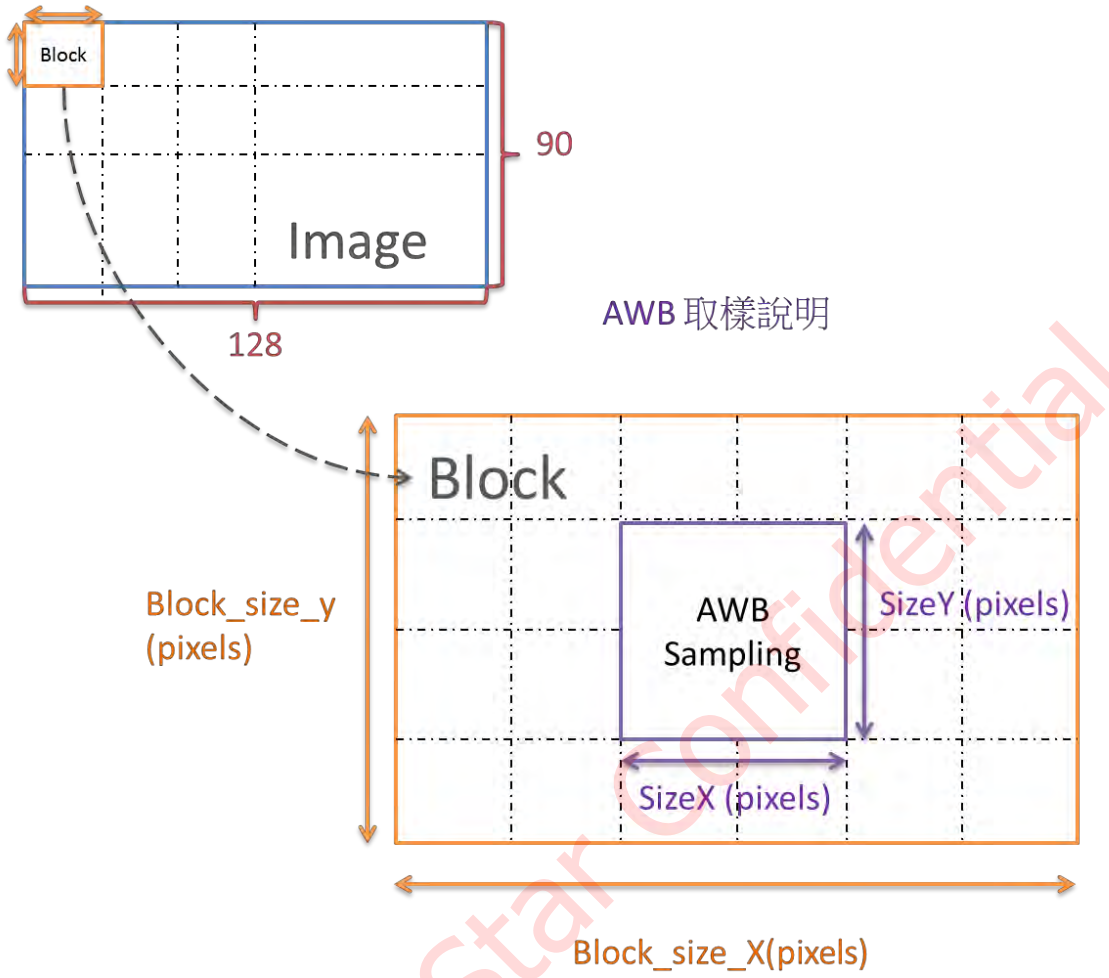


Image resolution = Width * Height
Block_size_X = Width / 128
Block_size_Y = Height / 90

19. MI_ISP 設置 AF FILTER

MI_ISP_CUS3A_SetAFFilter

【目的】
設定 AF Filter 參數

【語法】
MI_RET MI_ISP_CUS3A_SetAFFilter (MI_U32 Channel, CusAFFilter_t *data);

【描述】
調用此介面設置 AF Filter 參數

【參數】

參數名稱	描述
nChannel	視訊輸入通道號碼(一般為 0)
data	AF Filter 參數

【返回值】

返回值	描述
MI_RET_SUCCESS	成功
MI_RET_FAIL	失敗

CusAFFilter_t

【說明】ISP AF 硬體統計值

【定義】

```
typedef struct
{
    MI_U16 u16IIR1_a0;
    MI_U16 u16IIR1_a1;
    MI_U16 u16IIR1_a2;
    MI_U16 u16IIR1_b1;
    MI_U16 u16IIR1_b2;
    MI_U16 u16IIR1_1st_low_clip;
    MI_U16 u16IIR1_1st_high_clip;
    MI_U16 u16IIR1_2nd_low_clip;
    MI_U16 u16IIR1_2nd_high_clip;
    MI_U16 u16IIR2_a0;
    MI_U16 u16IIR2_a1;
    MI_U16 u16IIR2_a2;
    MI_U16 u16IIR2_b1;
    MI_U16 u16IIR2_b2;
    MI_U16 u16IIR2_1st_low_clip;
    MI_U16 u16IIR2_1st_high_clip;
    MI_U16 u16IIR2_2nd_low_clip;
    MI_U16 u16IIR2_2nd_high_clip;
} CusAFFilter_t;
```

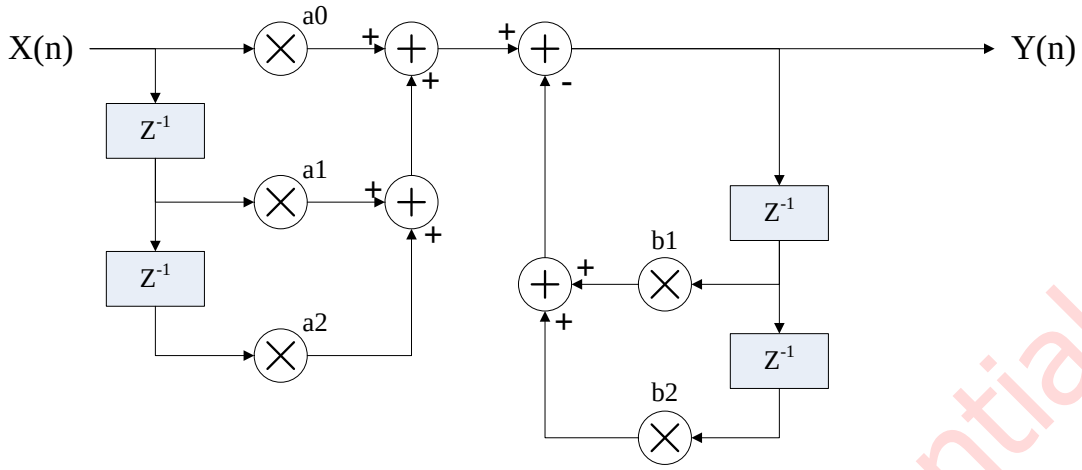
【成員】

IIR參數如下

名稱	bit 表示	描述	IIR1 default	IIR2 default
a0	S+9	a0 乘法器	63	63
a1	S+10	a1 乘法器	-126	-126
a2	S+9	a2 乘法器	63	63
b1	S+7	b1 乘法器	65	-109
b2	S+7	b2 乘法器	55	48
1st_low_clip	10	X(n) input low clip	0	0
1st_high_clip	10	X(n) input high clip	320	320
2nd_low_clip	10	Y(n) output low clip	0	0
2nd_high_clip	10	Y(n) output high clip	1023	1023

** IIR1 default 為 IIR High，IIR2 default 為 IIR Low

IIR 係數，架構圖如下：



【需求】

header file：mi_isp_twinkie.h / mi_isp_pretzel.h / mi_isp.h (for Macaron)

.so：libmi_isp_twinkie.so / libmi_isp_pretzel.so / libmi_isp.so (for Macaron)

20. MI_ISP 設置 AF FILTER SQUARE

MI_ISP_CUS3A_SetAFFilterSq

【目的】
設定 AF Filter Square 參數

【語法】
MI_RET MI_ISP_CUS3A_SetAFFilterSq (MI_U32 Channel, CusAFFilterSq_t *data);

【描述】
調用此介面設置 AF Filter Square 參數

【參數】

參數名稱	描述
nChannel	視訊輸入通道號碼(一般為 0)
data	AF Filter Square 參數

【返回值】

返回值	描述
MI_RET_SUCCESS	成功
MI_RET_FAIL	失敗

CusAFFilterSq_t

【說明】ISP AF Square 硬體統計值

【定義】

```
typedef struct
{
    MI_BOOL bFIRYSatEn;
    MI_U16 u16FIRYThd;

    MI_BOOL bIIRSquareAccEn;
    MI_BOOL bFIRSquareAccEn;

    MI_U16 u16IIR1Thd;
    MI_U16 u16IIR2Thd;
```

```
    MI_U16 u16FIRHThd;
    MI_U16 u16FIRVThd;
    MI_U8 u8AFTblX[12];
    MI_U16 u16AFTblY[13];
} CusAFFilter_t;
```

【成員】

名稱	描述
bFIRYSatEn	FIR Filter Y 閥值控制
u16FIRYThd	If bFIRYSatEn = 1 則 pixel 亮度小於 u16FIRYThd 時，就會列入 fir filter 計算中，且回傳大於 u16FIRYThd 的 pixel 個數(於 AF 統計值中)，數值範圍: 0 ~ 1023。
bIIRSquareAccEn	IIR Filter 增強控制
bFIRSquareAccEn	FIR Filter 增強控制
u16IIR1Thd	IIR Filter Output = IIR Filter Output – IIRThd。數值範圍: 0 ~ 1023。
u16IIR2Thd	IIR Filter Output = IIR Filter Output - IIRThd。數值範圍: 0 ~ 1023。
u16FIRHThd	FIR Filter Output = FIR Filter Output - FIR Thd。數值範圍: 0 ~ 1023。
u16FIRVThd	FIR Filter Output = FIR Filter Output - FIR Thd。數值範圍: 0 ~ 1023。
u8AFTblX[12]	針對 IIR 與 FIR Filter，做一個 non-linear 的 mapping u8AFTblX 為橫軸，節點為二的冪次方累加，累加起來需大於 1024。數值範圍: 0 ~ 15。
u16AFTblY[13]	針對 IIR 與 FIR Filter，做一個 non-linear 的 mapping u8AFTblY 為縱軸，數值範圍: 0 ~ 8191。

SquareACC 使用時機，針對中高對比的粗邊反差，做放大的動作。且低於 Thd 的值，會砍為 0，用來抗低照下的噪點。

參考設定 (統計值放大效果)
u8AFTblX = 6,7,7,6,6,6,7,6,6,7,6,6
u16AFTblY = 0,154,486,1178,1638,2099,2560,3635,4198,4813,6451,7270,8191

參考設定 (統計值保持原狀)
u8AFTblX = 6,7,7,6,6,6,7,6,6,7,6,6
u16AFTblY = 0,64,192,320,384,448,512,640,704,768,896,960,1023

【需求】
header file：mi_isp_twinkie.h / mi_isp_pretzel.h / mi_isp.h (for Macaron)
.so：libmi_isp_twinkie.so / libmi_isp_pretzel.so / libmi_isp.so (for Macaron)

【備註】
Twinkie 不支援這支 API。

21. MI_ISP 設置 AF ROI MODE

MI_ISP_CUS3A_SetAFRoiMode

【目的】
設定 AF 統計值的模式

【語法】
MI_RET MI_ISP_CUS3A_SetAFRoiMode (U32 nChannel, CusAFRoiMode_t *data);

【描述】
調用此介面設置 AWB 統計值來源

【參數】

參數名稱	描述
nChannel	視訊輸入通道號碼(一般為 0)
data	AF 統計值的模式

【返回值】

返回值	描述
MI_RET_SUCCESS	成功
MI_RET_FAIL	失敗

CusAFRoiMode_t

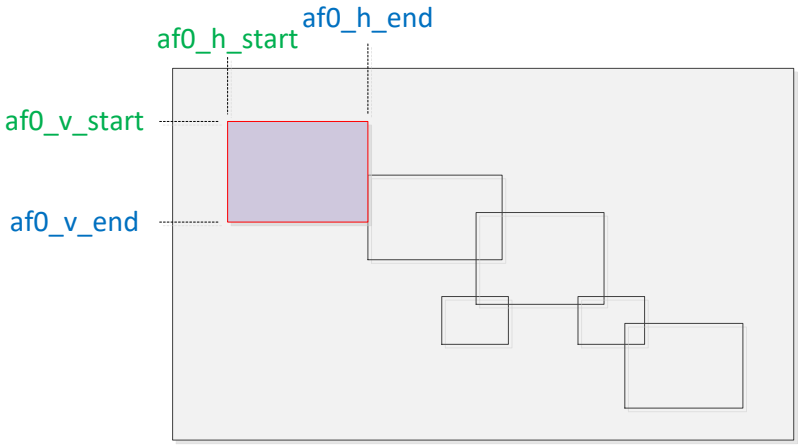
【描述】
設定 AF 統計值的模式

【定義】
```
typedef enum __attribute__ ((aligned (1)))
{
    AF_ROI_MODE_NORMAL,
    AF_ROI_MODE_MATRIX
} ISP_AF_ROI_MODE_e;

typedef struct
{
    ISP_AF_ROI_MODE_e mode;
    MI_U32 u32_vertical_block_number;
} CusAFRoiMode_t;
```

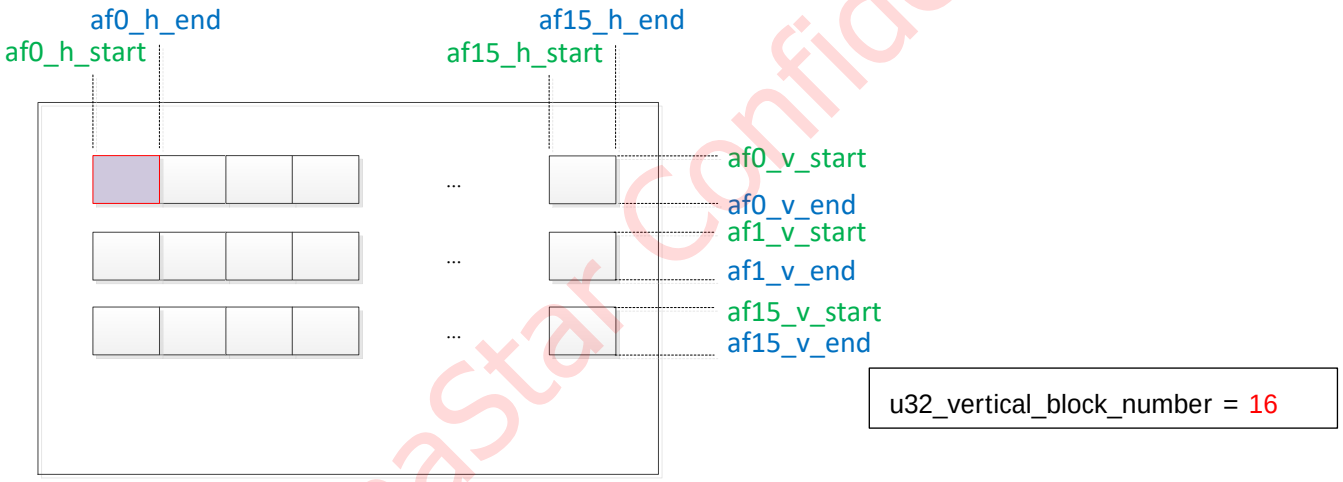
【成員】

名稱	描述
mode	AF_ROI_MODE_NORMAL: 可切為 16 組 ROI，window size 與位置可隨意分割，預設為此模式。 AF_ROI_MODE_MATRIX: 可切為16 * N組ROI，window size與位置稍有限制，但可切較多區塊。(N= u32_vertical_block_number)
u32_vertical_block_number	If mode = AF_ROI_MODE_MATRIX 可切為 16 * N 組 ROI。(N= u32_vertical_block_number, 1~16)

AF_ROI_MODE_NORMAL: AF ROI切割方式



AF_ROI_MODE_MATRIX: AF ROI切割方式



横軸，固定有16個
縱軸，會依據u32_vertical_block_number，來決定有幾組16個ROI，上圖為u32_vertical_block_number=16，就有 16*16 組ROI

【需求】
header file：mi_isp_twinkie.h / mi_isp_pretzel.h / mi_isp.h (for Macaron)
.so：libmi_isp_twinkie.so / libmi_isp_pretzel.so / libmi_isp.so (for Macaron)

【備註】
Twinkie 不支援這支 API。

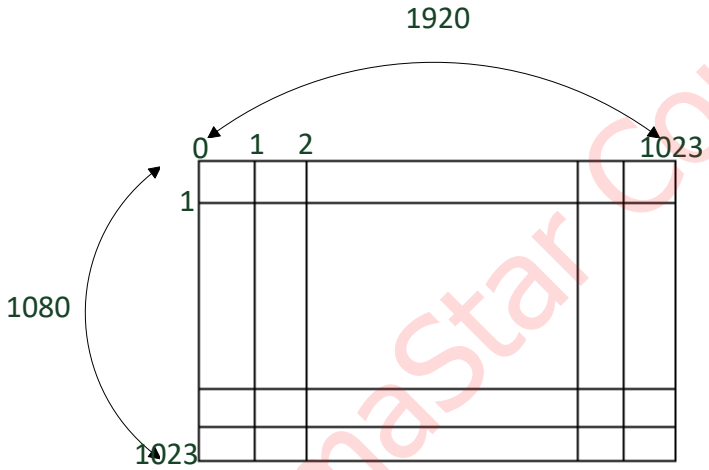
22. MI_ISP 設置 AF WINDOW

MI_ISP_CUS3A_SetAFWindow

【目的】
設定 AF Window

【語法】
MI_RET MI_ISP_CUS3A_SetAFWindow (U32 nChannel, CusAFWin_t *data);

【描述】
AF window 在 AF_Init 階段設置,將 CMOS sensor image 切成 1024*1024 座標格來設置 AF window



座標與實際 crop range 換算公式:

Image width * X /1024 = 真實 X 座標
Image_height * Y/1024 = 真實 Y 座標

Ex:
AF window 設置為 :
Start_X = 458
End_X = 566
Start_Y = 418
End_Y = 606

則在 1920x1080 的 CMOS sensor 下設置的 AF window 座標即為:
Real_Start_X = 1920 *458/1024 = 858
Real_End_X = 1920 *566/1024 = 1061
Real_Start_Y = 1080 *418/1024 = 440
Real_End_Y = 1080 *606/1024 = 639

【參數】

參數名稱	描述
nChannel	視訊輸入通道號碼(一般為 0)
data	AF Window 設置

【返回值】

返回值	描述
MI_RET_SUCCESS	成功
MI_RET_FAIL	失敗

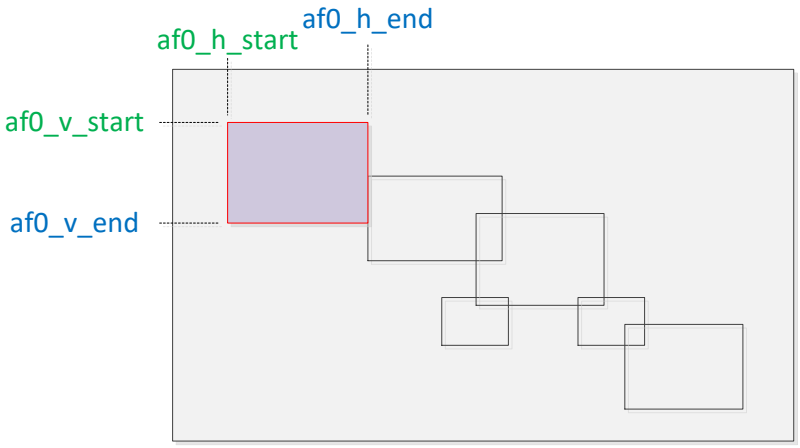
CusAFWin_t

```
typedef struct AF_WINDOW_PARAM_s
{
    MI_U32 u32StartX;                    /*range : 0~1023*/
    MI_U32 u32StartY;                    /*range : 0~1023*/
    MI_U32 u32EndX;                      /*range : 0~1023*/
    MI_U32 u32EndY;                      /*range : 0~1023*/
} AF_WINDOW_PARAM_t;

typedef struct
{
    MI_U8 u8WindowIndex;
    AF_WINDOW_PARAM_t stParaAPI;
} CusAFWin_t;
```

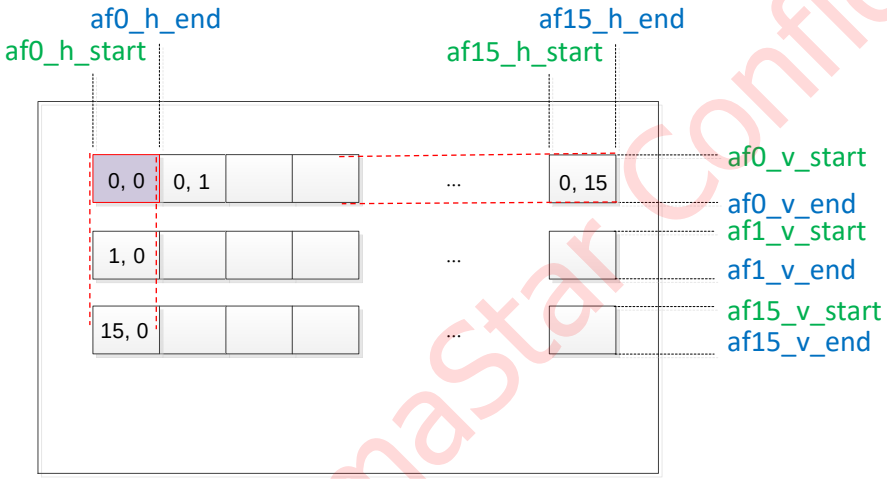
【參數】

參數名稱	描述
u8WindowIndex	AF ROI Index (0~15), for NORMAL & MARIX mode
stParaAPI	AF ROI position

【描述】
在 AF ROI Mode = AF_ROI_MODE_NORMAL 時，StrX, StrY, EndX, EndY 就是代表該 Window Index 的位置



在 AF ROI Mode = AF_ROI_MODE_MATRIX時，StrX, EndX,就是代表橫軸window切割的位置，StrY, EndY就是代表縱軸切割的位置



使用限制:

AF ROI Mode = AF_ROI_MODE_NORMAL

- 各自 window 寬高設置上，可互相重迭
- h/v end > h/v start

AF ROI Mode = AF_ROI_MODE_MATRIX

- 各自 window 寬設置上，可互相重迭，但高不可重迭
- h/v end > h/v start
- win0_v_start < win0_v_end < win1_v_start < win1_v_end < win2...
- win(0,0), win(1,0), ..., win(15,0) 使用共同的 h_start, h_end 設定，由 af0_h_start, h_end 決定
- win(0,1), win(1,1), ..., win(15,1) 使用共同的 h_start, h_end 設定，由 af1_h_start, h_end 決定，其他以此推論
- win(0,0), win(0,1), ..., win(0,15) 使用共同的 v_start, v_end 設定，由 af0_v_start, v_end 決定
- win(1,0), win(1,1), ..., win(1,15) 使用共同的 v_start, v_end 設定，由 af1_v_start, v_end 決定，其他以此推論

【需求】
header file：mi_isp_twinkie.h / mi_isp_pretzel.h / mi_isp.h (for Macaron)
.so：libmi_isp_twinkie.so / libmi_isp_pretzel.so / libmi_isp.so (for Macaron)

23. TWINKIE/PRETZEL/MACARON 支援差異列表

此章節描述 Twinkie，Pretzel 與 Macaron 所支援的功能列表

參數名稱	Twinkie	Pretzel	Macaron
AE 統計值	128*90	128*90	32*32
AE hist2	支援	支援	不支援
AF 統計值	IIR(34), FIR(34), Luma(34), YSat(24)	IIR(35), FIR(35), Luma(32), YSat(22)	IIR(35), FIR(35), Luma(32), YSat(22)
MI_ISP_CUS3A_SetAEWindowBlockNumber	支援	支援	不支援
MI_ISP_CUS3A_SetAEHistogramWindow	偏移+大小限制: 最大 128*90	偏移+大小限制: 最大 128*90	偏移+大小限制: 最大 32*32
MI_ISP_CUS3A_SetAFFilter	支援	支援	支援
MI_ISP_CUS3A_SetAFFilterSq	不支援	支援	支援
MI_ISP_CUS3A_SetAFRoiMode	不支援	支援	支援