SigmaStar
High Performance IPCAM/PCCAM System-on-Chip

AE/AWB/AF Interface Version 1.3

REVISION HISTORY

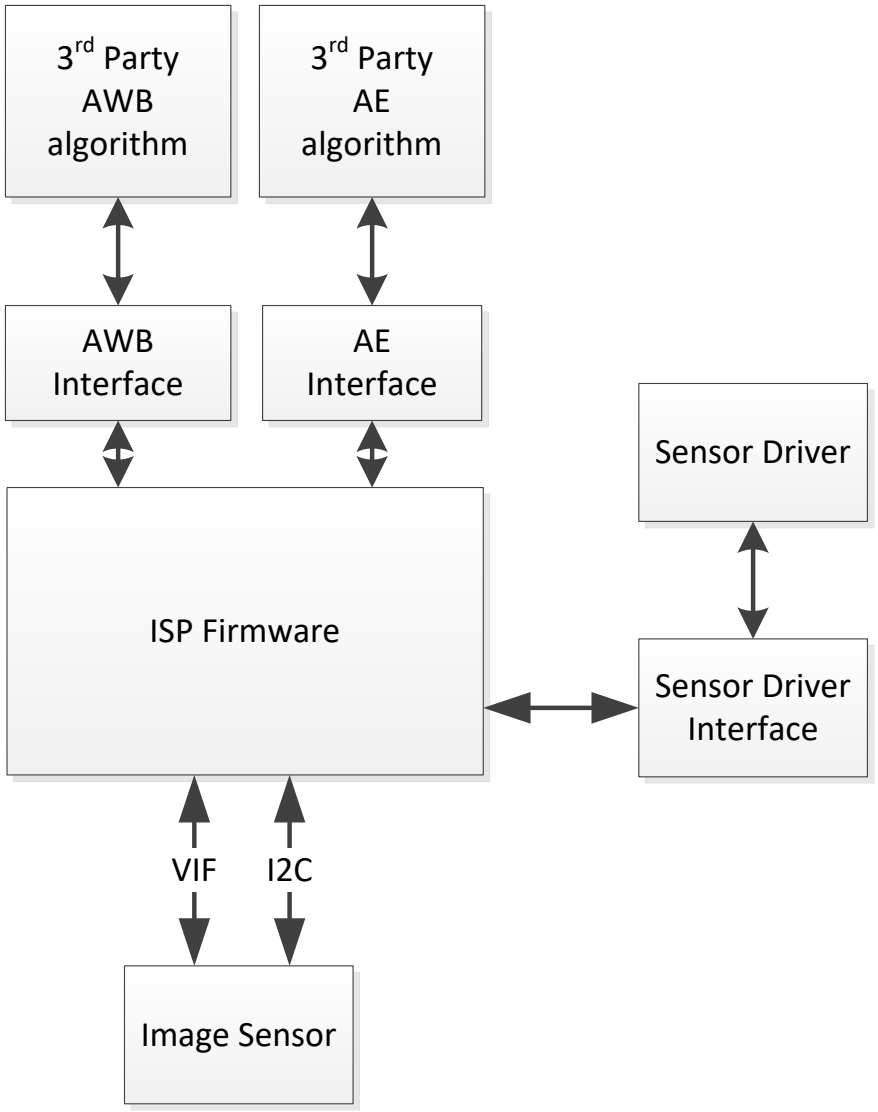
Revision No.	Description	Date
1.0 (cus3a v1.0)	• Formal release	02/19/2019
1.1 (cus3a v1.0)	• Added 3A RegInterface	03/06/2019
1.2 (cus3a v1.0)	• Updated AF Windows Description	03/14/2019
1.3 (cus3a v1.1)	• Added AF Filter Parameter for Pudding • Updated Cust3A to ver 1.1 (updated AE Init, Info, Result, AWB Info) • Added Get Cus3a Version API	12/04/2019

TABLE OF CONTENTS

REVISION HISTORY i
TABLE OF CONTENTS ii
1. 系统架构 1
2. AE/AWB 运作流程 2
3. AE/AWB STATISTIC FORMAT 3
4. AF STATISTIC FORMAT 4
4.1. 各 AF Filter 说明 5
5. AE RESULT 10
6. AWB RESULT 11
7. AF RESULT 12
8. AE,AWB,AF 库界面 13
9. AE 回调界面 14
10. AE 数据结构 16
11. AWB 回调界面 24
12. AWB 数据结构 26
13. AF 回调界面 30
14. AF 数据结构 32
15. SAMPLE CODE 35
16. MI_ISP 设置 AE BLOCK NUMBER 45
17. MI_ISP 设置 AE HISTOGRAM WINDOW 47
18. MI_ISP 设置 AWB SAMPLING 50
19. MI_ISP 设置 AF FILTER 52
20. MI_ISP 设置 AF FILTER SQUARE 56
21. MI_ISP 设置 AF ROI MODE 58
22. MI_ISP 设置 AF WINDOW 61
23. GET CUST3A VERSION 65
24. TWINKIE/PRETZEL/MACARON/PUDDING 支援差异列表 66

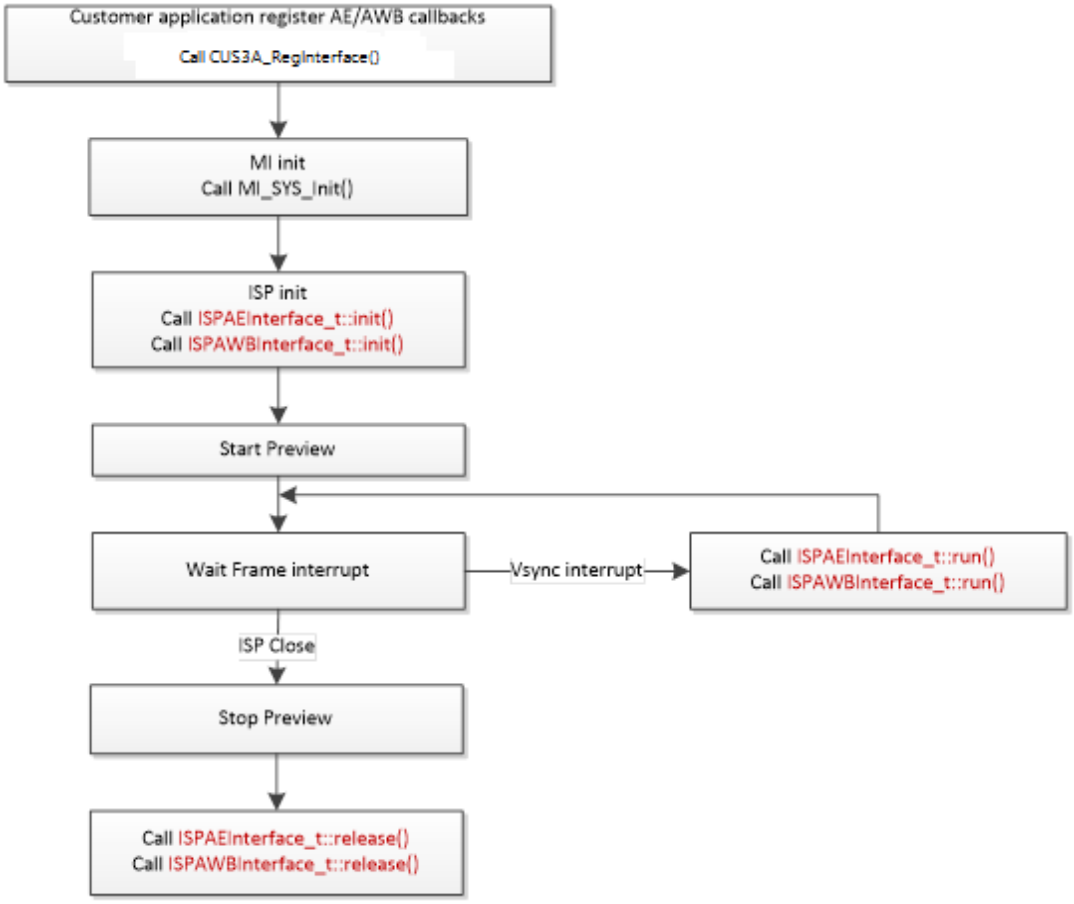
1. 系统架构

ISP 架构，除主要 ISP 核心外，另外可外挂三种模块。AWB/AE 算法库提透过 AWB/AE Interface 与 ISP 连结，客户可自行开发算法，ISP 透过 AWB/AE Interface 提供硬件统计值给算法，算法回报对应的结果给 ISP，ISP 核心将结果透过适当的方式写入硬件。Sensor 驱动透过 Sensor Interface 与 ISP 链接，不同型号的 Sensor 必须实作对应的 Sensor 驱动。



2. AE/AWB 运作流程

客户程序必须在 MI_SYS_Init()之前透过 CUS3A_RegInterface()注册算法库(callback function)，之后 ISP 初始化阶段，ISP 核心透过 init()通知算法进行初始化工作。当前端 Sensor 开始取影像，ISP 核心会在每侦影像中断时呼叫算法 run()，并且将硬件统计值带入算法中。当 ISP 关闭时，透过 release()通知算法释放所有资源。



流程图中红色字体为算法必须提供的 callback function。

3. AE/AWB STATISTIC FORMAT

AE/AWB 统计值在程序代码中定义如下，每张影像均会产生 128*90 笔取样数据。
在 Macaron, Pudding 中，AE 统计值，每张影像只会产生 32*32 笔取样数据。

```
#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 会在每次 Frame End 将 AE/AWB 统计值传给算法端，数据排列方式如下表。ISP 将影像切为 isp_3A_ROW x isp_3A_COL 区域，并计算每个区域统计值填入
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

设置 AF 统计 window(ISP_AF_RECT),提供 16*n(n = 1~16)个 window 可供设置(af_stats_win[16]),每祯会将统计结果 output, output 结构如 ISP_AF_STATS,每个 Window 会 output 6 种 filter 的统计值-

各 filter bit 数表示如下

Filter Bit & Chip Type	Twinkie	Pretzel	Macaron/Pudding
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;

//for Pretzel, Macaron, Pudding
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];
} CusAFStats_t;
```

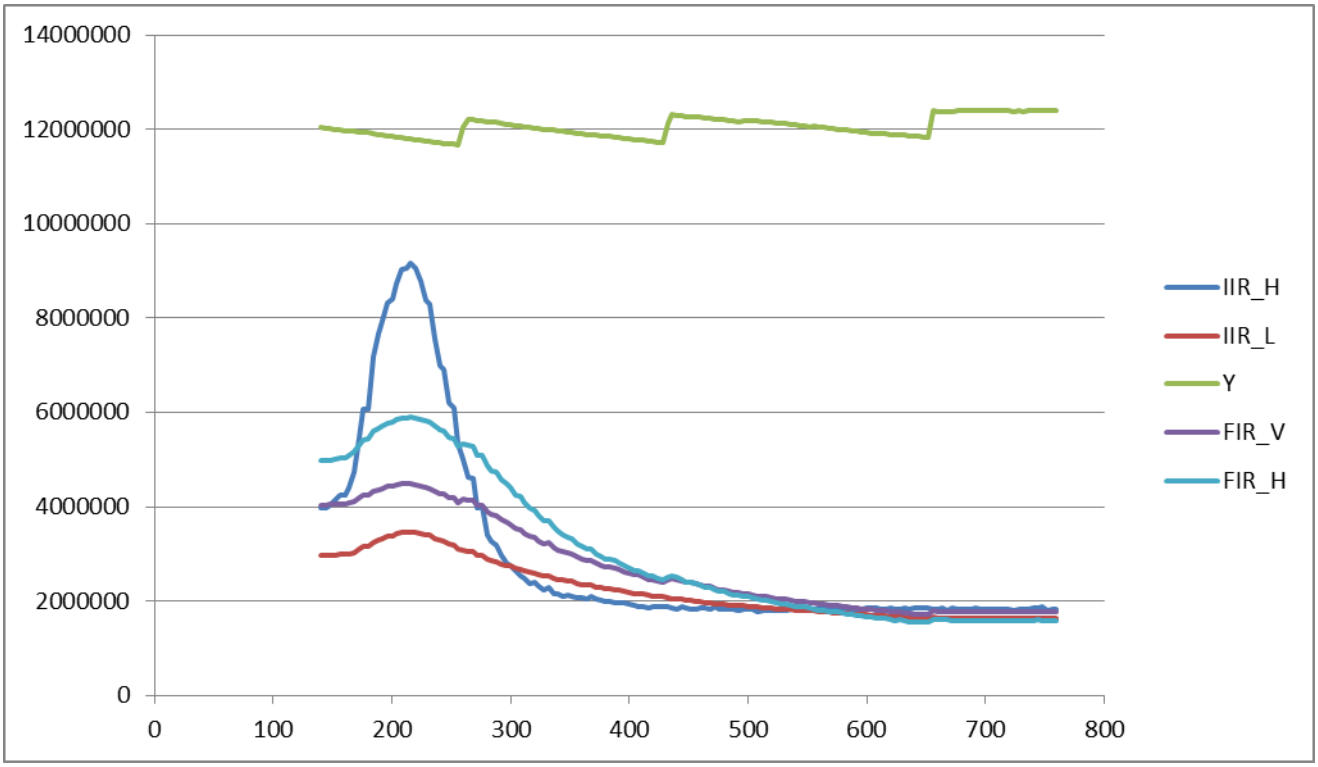
```
//for Twinkie
typedef struct{
    MI_U8 high_iir[6*16];
    MI_U8 low_iir[6*16];
    MI_U8 luma[6*16];
    MI_U8 sobel_v[6*16];
    MI_U8 sobel_h[6*16];
    MI_U8 ysat[3*16];
} AF_STATS_PARAM_t ;

typedef struct{
    AF_STATS_PARAM_t stParaAPI;
} CusAFStats_t;
```

4.1. 各 AF Filter 说明

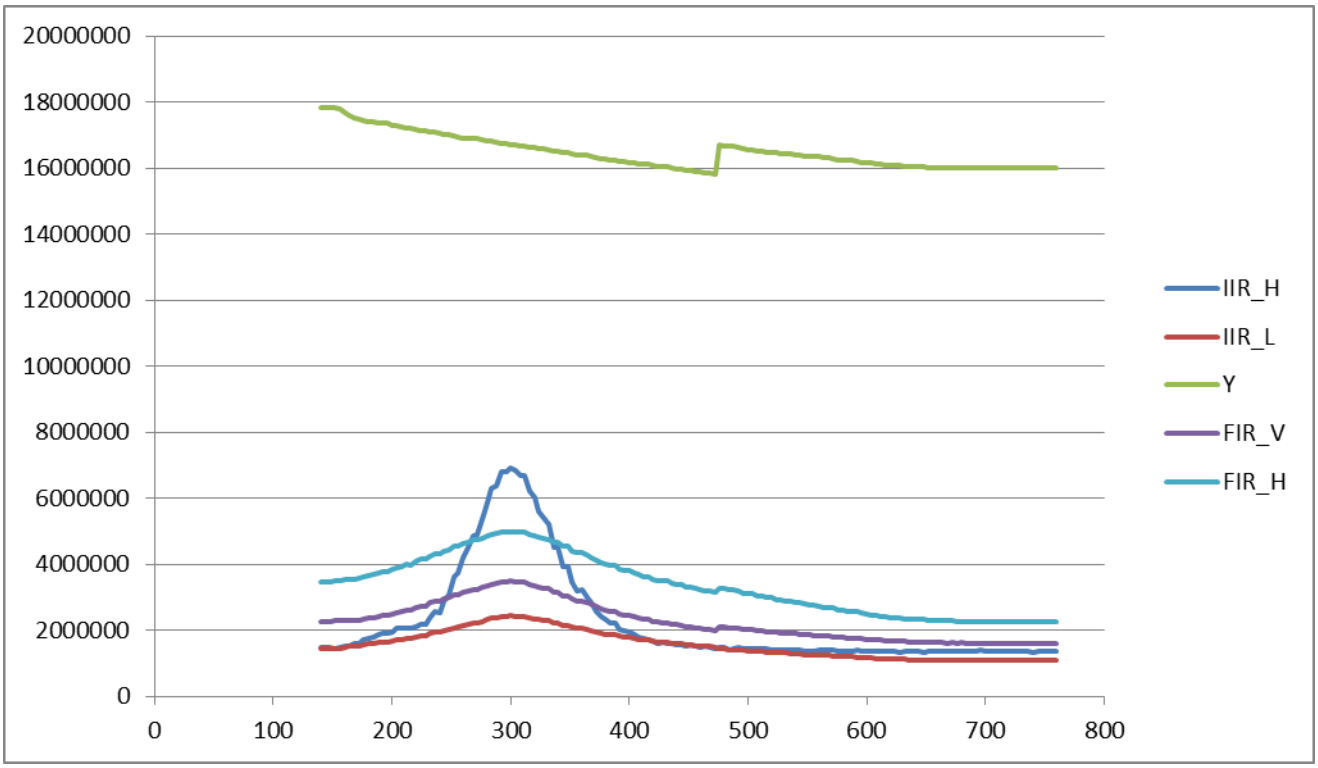
参数名称	描述	输出/输入
IIR_1	分区间统计的 AF 水平方向 IIR 滤波器统计值	输出
IIR_2	分区间统计的 AF 水平方向 IIR 滤波器统计值	输出
Luma	分区间统计的 AF 亮度统计值	输出
FIR_H	分区间统计的 AF 水平方向 FIR 滤波器统计值	输出
FIR_V	分区间统计的 AF 垂直方向 FIR 滤波器统计值	输出
YSat	分区间统计的 AF FIR 滤波器，大于 YThd 的统计值个数	输出

数据范例:

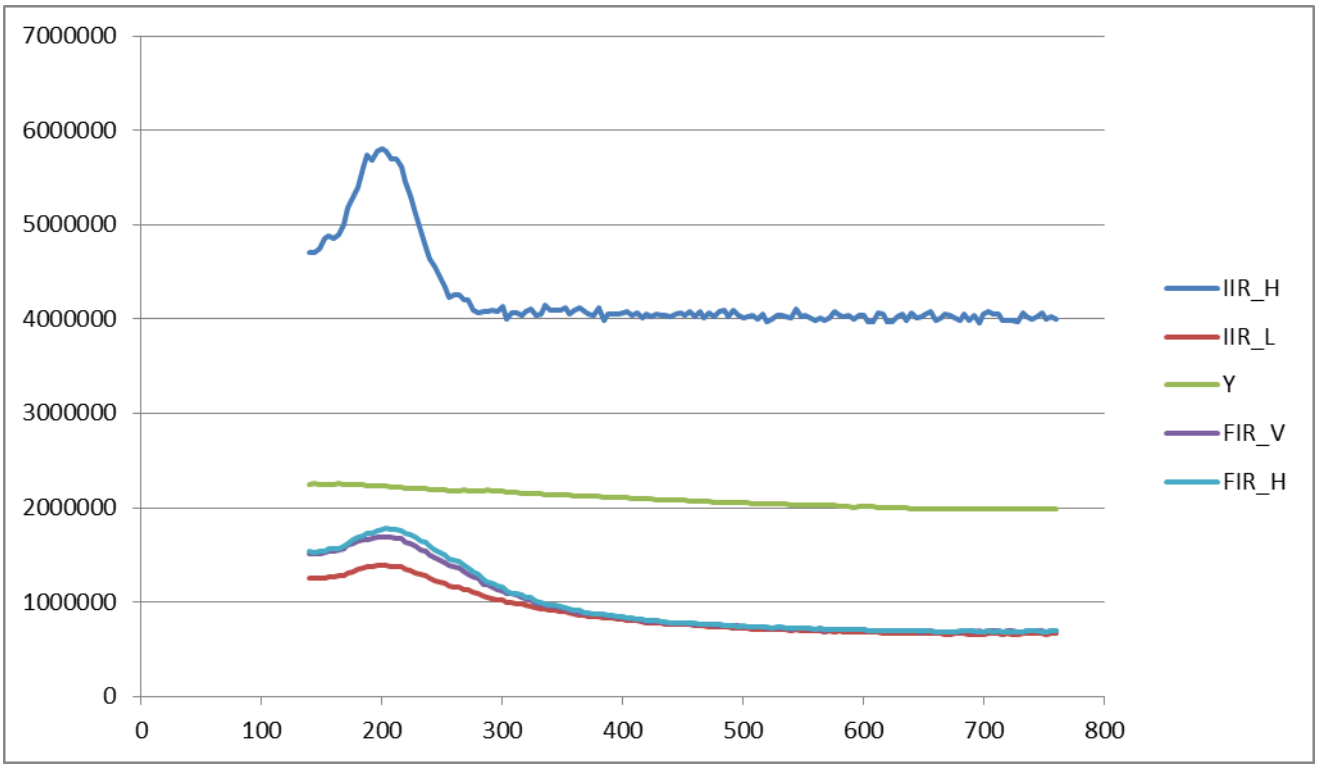
一般场景远焦:



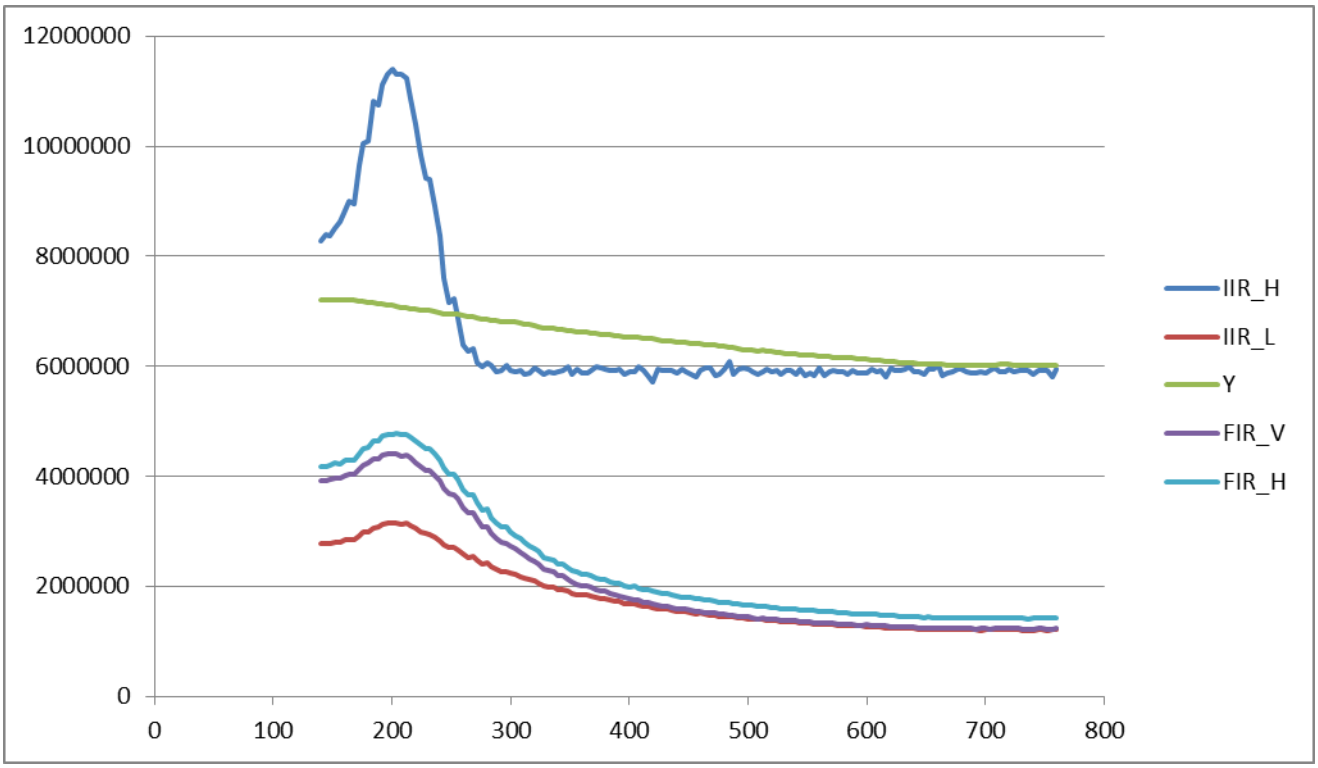
一般场景近焦:



低亮度场景远焦:



低亮度场景远焦+灯源:



5. AE RESULT

当 ISP 呼叫 AE 算法 run()后，AE 算法必须提供相对应的结果给 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 */

    /*CUS3A v1.1*/
    U32 FNx10;          /**< F number * 10 */
    U32 DebandFPS;      /**< Target fps when running auto debanding*/
    U32 WeightY;        /**< frame brightness with ROI weight*/

}__attribute__((packed, aligned(1))) ISP_AE_RESULT;
```

Size : ISP_AE_RESULT 长度，用于处理版本向下兼容问题。
Change: 若该值为 1，ISP 将此次设定值写入硬件，若为 0 ISP 会忽略此次的结果。
Shutter: 设定 Sensor 曝光时间。
SensorGain: 设定 Sensor gain，Digital/Analog gain 分配率由 Sensor 驱动决定。
IspGain: 设定 ISP digital gain。
ShutterHdrShort: 设定短曝 Sensor 曝光时间。
SensorGainHdrShort: 设定短曝 Sensor gain，Digital/Analog gain 分配率由 Sensor 驱动决定。
IspGainHdrShort: 设定短曝 ISP digital gain。
u4BVx16384: 目前场景的亮度值，此处计算公式比照 APEX。此设定值将会被 IQ 算法参考。
AvgY：此张影像的平均灰阶亮度，此设定值将会被 IQ 算法参考。
HdrRatio：此张影像的 HDR 长短曝比例，此设定值将会被 IQ 算法参考。
Fnx10: 设定 FN Numbe
DebandFPS: 设定 FPS
WeightY: 此张影像经过 ROI weighting 的平均灰阶亮度，此设定值将会被 IQ 算法参考。

6. AWB RESULT

当 ISP 呼叫 AWB 算法 run()后，AWB 算法必须提供相对应的结果给 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 长度，用于处理版本向下兼容问题。
Change: 若该值为 1，ISP 将此次设定值写入硬件，若为 0，ISP 会忽略此次的结果。
R_gain: R channel 增益。
G_gain: G channel 增益。
B_gain: B channel 增益。
ColorTmp: 目前场景色温。

7. AF RESULT

当 ISP 呼叫 AF 算法 run()后，AF 算法必须提供相对应的结果给 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: 若该值为 1，通知 AF 演算出的焦距(Focal_pos)生效
Focal_pos: AF 算法算出的焦距结果。

8. AE,AWB,AF 库界面

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);

说明:
ISP 提供 AE,AWB,AF 库注册接口
可使用 ***CUS3A_RegInterface*** 一次注册 AE, AWB, AF
或使用 ***CUS3A_AERegInterface***、***CUS3A_AWBRegInterface***、***CUS3A_AFRegInterface*** 来分别注册 AE, AWB, AF

参数:

参数名称	描述	输出/输入
nCh	nCh = 0	输入
pAe	算法实作端提供 AE 库指针，若不需注册，则填 NULL	输入
pAwb	算法实作端提供 AWB 库指针，若不需注册，则填 NULL	输入
pAf	算法实作端提供 AF 库指针，若不需注册，则填 NULL	输入

返回值:

返回值	描述
0	注册成功
<0	注册失败

9. AE 回调界面

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

说明:
ISP 透过此接口通知 AE 库进行初始化

参数:

参数名称	描述	输出/输入
pdata	算法私有数据指针	输入
init_state	AE 初始值	输入

返回值:

返回值	描述
0	初始化成功
-1	初始化失败

void (*release)(void* pdata)

说明:
ISP 透过此接口通知 AE 库进行释放程序

参数:

参数名称	描述	输出/输入
pdata	算法私有数据指针	输入

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

说明:
ISP 透过此接口呼叫 AE 库进行运算

参数:

参数名称	描述	输出/输入
pdata	算法私有数据指针	输入
info	ISP AE 统计值数据结构指针	输入
result	ISP AE 计算结果数据结构指针	输出

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

说明:
ISP 透过此接口控制 AE 参数

参数:

参数名称	描述	输出/输入
pdata	算法私有数据指针	输入
cmd	AE 控制指令	输入
param	AE 控制指令参数	输入

10. AE 数据结构

ISP_AE_INTERFACE

说明:

AE 库注册结构体，算法需实作并且填写 init,release,run,ctrl 回调指针，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;
```

成员:

名称	描述
pdata	AE 库私有数据指针
init	AE 库初始化回调指标
release	AE 库释放回调指针
run	AE 库计算回调指标，当 ISP 收到一侦影像统计值时，透过该指针要求 AE 算法进行计算。
ctrl	AE 库控制回调指标

ISP_AE_INIT_PARAM

说明:

ISP 透过此结构，告知 AE 算法相关硬件初始状态

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_min;       /**< sensor_gain_min Minimum 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*/
    U32 shutterHDRShort_step;          /**< shutter Shutter step ns*/
    U32 shutterHDRShort_min;           /**< shutter Shutter min us*/
    U32 shutterHDRShort_max;           /**< shutter Shutter max us*/
    U32 sensor_gainHDRShort_min;       /**< sensor_gain_min Minimum Sensor gain, 1X = 1024*/
    U32 sensor_gainHDRShort_max;       /**< sensor_gain_max Maximum Sensor gain, 1X = 1024*/

    /*CUS3A v1.1*/
    U32 AvgBlkX;                /**< HW statistics average block number*/
    U32 AvgBlkY;                /**< HW statistics average block number*/
}ISP_AE_INIT_PARAM;
```

成员:

参数名称	描述	输出/输入
Size	ISP_AE_INIT_PARAM 结构长度	输入
sensor_id[32]	Image sensor 名称	输入
shutter	曝光时间初始值 us	输入
shutter_step	曝光时间步距 ns	输入
shutter_min	最小曝光时间 us	输入
shutter_max	最长曝光时间 us	输入
sensor_gain	Sensor 增益初始值	输入

参数名称	描述	输出/输入
sensor_gain_min	Sensor 增益最小值	输入
sensor_gain_max	Sensor 增益最大值	输入
isp_gain	Isp 增益初始值	输入
isp_gain_max	Isp 增益最大值	输入
FNx10	光圈值*10	输入
fps	每秒影像侦数	输入
shutterHDRShort_step	HDR 短曝，曝光时间步距 ns	输入
shutterHDRShort_min	HDR 短曝，最小曝光时间 us	输入
shutterHDRShort_max	HDR 短曝，最长曝光时间 us	输入
sensor_gainHDRShort_min	HDR 短曝，Sensor 增益最小值	输入
sensor_gainHDRShort_max	HDR 短曝，Sensor 增益最大值	输入
AvgBlkX	AE 统计值横向切割数量	输入
AvgBlkY	AE 统计值纵向切割数量	输入

ISP_AE_INFO

说明:

ISP AE 硬件统计值

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*/

    /*CUS3A v1.1*/
    U32 PreAvgY;            /**< Previous frame brightness */
    U32 HDRCtlMode;         /**< 0 = HDR off;*/
                            /**< 1 = Separate shutter & Separate sensor gain settings*/
                            /**< 2 = Separate shutter & Share sensor gain settings*/
                            /**< 3 = Share shutter & Separate sensor gain settings*/
    U32 FNx10;              /**< Aperture in FNx10 */
    U32 CurFPS;             /**< Current sensor FPS */
    U32 PreWeightY;         /**< Previous frame brightness with ROI weight */
} __attribute__((packed, aligned(1))) ISP_AE_INFO;
```

成员:

参数名称	描述	输出/输入
Size	ISP_AE_INFO 结构长度	输入
hist1	Histogram 1	输入
hist2	Histogram 2 (Macaron/Pudding not support)	输入
AvgBlkX	横向切割数量	输入
AvgBlkY	纵向切割数量	输入
avgs	区块亮度统计值	输入

参数名称	描述	输出/输入
Shutter	统计值发生时的曝光时间 us	输入
SensorGain	统计值发生时的 Sensor Gain 1X=1024	输入
IspGain	统计值发生时的 Isp Gain 1X=1024	输入
ShutterHDRShort	统计值发生时的曝光时间 us (HDR short)	输入
SensorGainHDRShort	统计值发生时的 Sensor Gain 1X=1024 (HDR short)	输入
IspGainHDRShort	统计值发生时的 Isp Gain 1X=1024 (HDR short)	输入
PreAvgY	上一次 AE 结果的平均亮度 8bit*10	输入
HDRCtlMode	HDR mode 时，sensor 回传该 sensor 对于 shutter/gain 所支持的格式	输入
FNx10	统计值发生时的 FN	输入
CurFPS	当前 FPS	输入
PreWeightY	上一次 AE 经过 ROI weighting 的平均亮度 8bit*10	输入

ISP_AE_RESULT

说明:

ISP AE 算法计算结果

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 */

     /*CUS3A v1.1*/
     U32 FNx10;           /**< F number * 10 */
     U32 DebandFPS;      /**< Target fps when running auto debanding*/
     U32 WeightY;        /**< frame brightness with ROI weight*/
}__attribute__((packed, aligned(1))) ISP_AE_RESULT;
```

成员:

参数名称	描述	输出/输入
Size	ISP_AE_RESULT 结构长度	输入
Change	通知 ISP 此次计算结果是否要套用到硬件	输出
Shutter	AE 算法回报曝光时间 us	输出
SensorGain	AE 算法回报 Sensor 增益 1X=1024	输出
IspGain	AE 算法回报 ISP 增益 1X=1024	输出
ShutterHdrShort	AE 算法回报曝光时间 us (HDR Short)	输出
SensorGainHdrShort	AE 算法回报 Sensor 增益 1X=1024 (HDR Short)	输出
IspGainHdrShort	AE 算法回报 ISP 增益 1X=1024 (HDR Short)	输出
u4Bvx16384	AE 算法回报目前场景亮度估测值	输出
AvgY	AE 算法回报目前影像平均亮度	输出

参数名称	描述	输出/输入
HdrRatio	AE 算法回报目前影像 HDR 长短曝比例 1x=1024	输出
FNx10	AE 算法回报目前影像 FN Number	输出
DebandFPS	AE 算法回报目前影像 FPS	输出
WeightY	AE 算法回报目前影像经过 ROI weighting 的平均亮度	输出

ISP_AE_CTRL_CMD

说明:

AE 控制指令，ISP 透过回调接口

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

设定 AE 参数，参数 param 型态随不同的控制命令改变。

宣告:

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

数值:

名称	描述	参数型态
ISP_AE_FPS_SET	ISP 通知 AE 库目前的 FPS 更动	U32

11. AWB 回调界面

int (*init)(void* pdata)

说明:
ISP 透过此接口通 AWB 库进行初始化

参数:

参数名称	描述	输出/输入
pdata	算法私有数据指针	输入

返回值:

返回值	描述
0	初始化成功
-1	初始化失败

void (*release)(void* pdata)

说明:
ISP 透过此接口通知 AWB 库进行释放程序

参数:

参数名称	描述	输出/输入
pdata	算法私有数据指针	输入

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

说明:
ISP 透过此接口呼叫 AWB 库进行运算

参数:

参数名称	描述	输出/输入
pdata	算法私有数据指针	输入
awb_info	ISP AWB 统计值数据结构指针	输入
ae_info	ISP AE 统计值数据结构指针	输入
result	ISP AWB 计算结果数据结构指针	输出

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

说明:
ISP 透过此接口控制 AWB 参数

返回值:

返回值	描述
0	控制指令成功
<0	注册失败

参数:

参数名称	描述	输出/输入
pdata	算法私有数据指针	输入
cmd	AWB 控制指令	输入
param	AWB 控制指令参数	输入

12. AWB 数据结构

ISP_AWB_INTERFACE

说明:

AWB 库注册结构体，算法需实作并且填写 init,release,run,ctrl 回调指针，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
```

成员:

名称	描述
pdata	AWB 库私有数据指针
init	AWB 库初始化回调指标
release	AWB 库释放回调指针
run	AWB 库计算回调指标，当 ISP 收到一侦影像统计值时，透过该指针要求 AWB 算法进行计算。
ctrl	AWB 库控制回调指标

ISP_AWB_INFO

说明:

ISP AWB 硬件统计值

宣告:

```
/*! @brief AWB HW statistics data*/
typedef struct
{
      U32 AvgBlkX;
      U32 AvgBlkY;
      U32 CurRGain;
      U32 CurGGain;
      U32 CurBGain;
      void *avgs;

     /*CUS3A v1.1*/
     U8 HDRMode;       /**< Noramal or HDR mode */
     void* pAwbStatisShort;      /**< Short Shutter AWB statistic data*/
     U32 u4BVx16384;    /**< From AE output, Bv * 16384 in APEX system, EV = Av + Tv = Sv + Bv  */
     S32 WeightY;       /**< frame brightness with ROI weight0 */
}__attribute__((packed, aligned(1))) ISP_AWB_INFO;
```

成员:

参数名称	描述	输出/输入
AvgBlkX	横向切割数量	输入
AvgBlkY	纵向切割数量	输入
CurRGain	反应在目前影像的 AWB Gain	输入
CurGGain	反应在目前影像的 AWB Gain	输入
CurBGain	反应在目前影像的 AWB Gain	输入
avgs	ISP AWB 区块 RGB 统计值	输入
HDRMode	目前影像是否为 HDR mode	输入
pAwbStatisShort	ISP AWB 区块 RGB 统计值 (HDR mode 下的短曝信息)	输入
U4BVx16384	AE 回传的 BV 信息	输入
WeightY	AE 回传的 ROI weighting 的平均亮度	输入

ISP_AWB_RESULT

说明:

ISP AWB 算法计算结果

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;
```

成员:

参数名称	描述	输出/输入
Size	ISP_AWB_RESULT 结构长度	输入
Change	通知 ISP 此次计算结果是否要套用到硬件	输出
R_gain	AWB 算法回报 AWB R 增益 1X = 1024	输出
G_gain	AWB 算法回报 AWB G 增益 1X = 1024	输出
B_gain	AWB 算法回报 AWB B 增益 1X = 1024	输出
ColorTmp	AWB 算法回报目前场景色温 K	输出

ISP_AWB_CTRL_CMD

说明:

AWB 控制指令，ISP 透过回调接口

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

设定 AE 参数，参数 param 型态随不同的控制命令改变。

宣告:

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

数值:

名称	描述	参数型态
ISP_AWB_MODE_SET	ISP 通知 AWB 库目前的 AWB 模式更动	U32

13. AF 回调界面

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

说明:
ISP 透过此接口通 AF 库进行初始化

参数:

参数名称	描述	输出/输入
pdata	算法私有数据指针	输入
param	AF 初始设定	输入/出

返回值:

返回值	描述
0	初始化成功
-1	初始化失败

void (*release)(void* pdata)

说明:
ISP 透过此接口通知 AF 库进行释放程序

参数:

参数名称	描述	输出/输入
pdata	算法私有数据指针	输入

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

说明:
ISP 透过此接口呼叫 AF 库进行运算

参数:

参数名称	描述	输出/输入
pdata	算法私有数据指针	输入
af_info	ISP AF 统计值数据结构指针	输入
result	ISP AF 计算结果数据结构指针	输出

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

说明:
ISP 透过此接口控制 AF 参数

返回值:

返回值	描述
0	控制指令成功
<0	注册失败

参数:

参数名称	描述	输出/输入
pdata	算法私有数据指针 (Reserve)	输入
cmd	AF 控制指令 (Reserve)	输入
param	AF 控制指令参数 (Reserve)	输入

14. AF 数据结构

ISP_AF_INTERFACE

说明:

AF 库注册结构体，算法需实作并且填写 init,release,run,ctrl 回调指针，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;
```

成员:

名称	描述
pdata	AF 库私有数据指针
init	AF 库初始化回调指标
release	AF 库释放回调指针
run	AF 库计算回调指标，当 ISP 收到一侦影像统计值时，透过该指针要求 AF 算法进行计算。
ctrl	AF 库控制回调指标

ISP_AF_INFO

说明:
ISP AF 硬件统计值

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;
```

成员:

参数名称	描述	输出/输入
Size	ISP_AF_INFO 数据结构长度	输入
af_stats	ISP AF 统计值	输出

ISP_AF_STATS

说明:
ISP AF 硬件统计值

宣告:

```
//for Pretzel, Macaron, Pudding
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;
```

```
//for Twinkie
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;
```

成员:

参数名称	描述	输出/输入
IIR_1	分区间统计的 AF 水平方向 IIR 滤波器统计值	输出
IIR_2	分区间统计的 AF 水平方向 IIR 滤波器统计值	输出
Luma	分区间统计的 AF 亮度统计值	输出
FIR_h	分区间统计的 AF 水平方向 FIR 滤波器统计值	输出
FIR_v	分区间统计的 AF 垂直方向 FIR 滤波器统计值	输出
YSat	分区间统计的 AF FIR 滤波器，大于 YThd 的统计值个数	输出

* IIR_1 default 为 IIR Low，IIR_2 default 为 IIR High
* 每个 filter 都是 16 组，前面九个预设对应画面的九宫格分割位置，第 16 个是自定义使用的，通常是拿来做 touch AF。
* IIR FIR 数值越高代表画面就越清晰。Luma 表示亮度。

使用参考:

区间加权权重: 建议"中央权重"。

统计值: 建议常时使用 IIR_1 (IIR Low)。
IIR_2 (IIR High) 在清晰时会有明显变化，但模糊时几乎不变动，适合做最清晰位置判断。
FIR 会有偏性，仅适合做强度变化参考。

ISP_AF_RESULT

说明:

ISP AF 算法计算结果

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;
```

成员:

参数名称	描述	输出/输入
Change	通知 ISP 此次计算结果是否要套用到硬件	输出
Focal_pos	AF 算法演算出的对焦 position	输出

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 =
    {
#if Twinkie || Pretzel || Macaron
        //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,
#elseif Pudding
        //filter setting with sign value
        //{s9, s10, s9, s7, s7}
        //high: 37, 0, -37, 83, 40; 37, 0, -37, -54, 34; 32, 0, -32, 14, 0
        //low:  15, 0, -15, -79, 44; 15, 0, -15, -115, 55; 14, 0, -14, -91, 37

        //convert to hw format (sign bit with msb)
        37, 0, 37+512, 83, 40, 0, 1023, 0, 1023, //high
        15, 0, 15+512, 79+128, 44, 0, 1023, 0, 1023, //low
          1, 37, 0, 37+512,  54+128, 34, 1, 32, 0, 32+512, 14, 0,  //high-e1, e2
          1, 15, 0, 15+512, 115+128, 55, 1, 14, 0, 14+512, 91+128, 37,  //low-e1, e2
#endif
    };

    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_i
nfo->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_i
nfo->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_i
nfo->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_i
nfo->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 设置 AE BLOCK NUMBER

MI_ISP_CUS3A_SetAEWindowBlockNumber

【目的】
设定 AE 的区块数量

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

【描述】
调用此界面设置 AE 区块数量

【参数】

参数名称	描述
nChannel	视讯输入信道号码(一般为 0)
eBlkNum	区块个数，将整张照片切割成 X*Y 格

【返回值】

返回值	描述
MI_RET_SUCCESS	成功
MI_RET_FAIL	失败

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

MS_CUST_AE_WIN_BLOCK_NUM_TYPE_e

【说明】AE区块

【定义】

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

【成员】

参数名称	描述
AE_16x24	画面分割成16*24块
AE_32x24	画面分割成32*24块
AE_64x48	画面分割成64*48块
AE_64x45	画面分割成64*45块
AE_128x80	画面分割成128*80块
AE_128x90	画面分割成128*90块

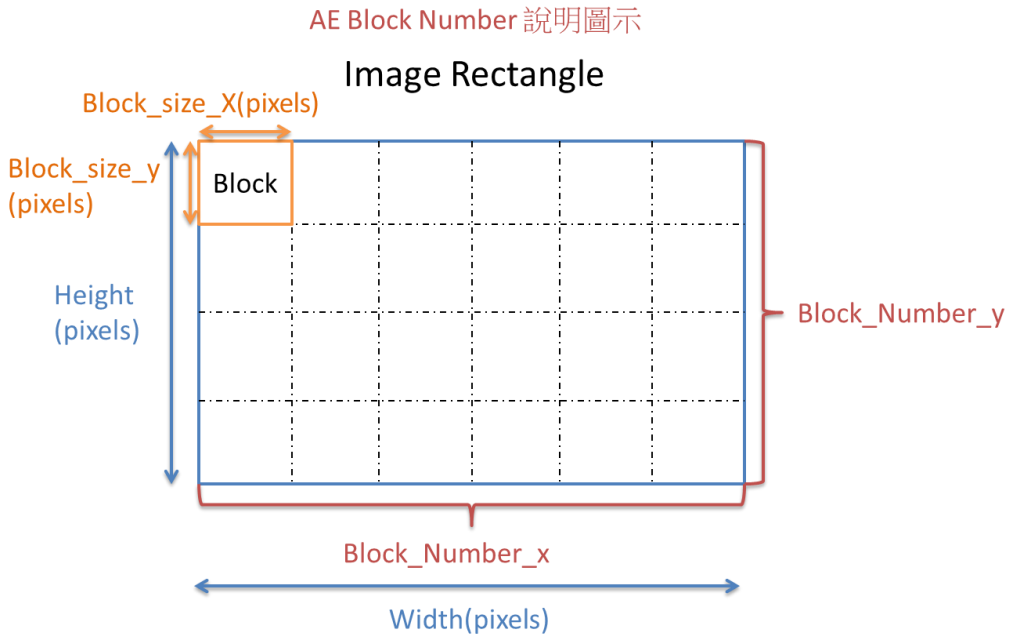


Image resolution = Width * Height
Block_size_X = Width / Block_Number_x
Block_size_Y = Height / Block_Number_y

【备注】

Macaron, Pudding 不支持这支 API，画面强制分割成 32*32 块。

17. MI_ISP 设置 AE HISTOGRAM WINDOW

MI_ISP_CUS3A_SetAEHistogramWindow

【目的】
AE 亮度分布统计值的窗口区域设定

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

【描述】
调用此接口设置 AE 亮度分布统计值的窗口区域位置及大小

【成员】

参数名称	描述
nChannel	视讯输入信道号码(一般为 0)
data	亮度分布统计值的窗口区域位置及大小，窗口编号(0 或 1)

【返回值】

返回值	描述
MI_RET_SUCCESS	成功
MI_RET_FAIL	失败

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

【注意】
目前只能支持同时设定两个窗口(0/1)。窗口区域可以重迭。

HistWin_t

【说明】AE 亮度分布统计值的窗口区域设定

【定义】

```
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;
```

【成员】

参数名称	描述
u2Stawin_x_offset	窗口起始坐标 x 轴偏移
u2Stawin_x_size	x 轴方向窗口大小
u2Stawin_y_offset	窗口起始坐标 y 轴偏移
u2Stawin_y_size	y 轴方向窗口大小
u2WinIdx	窗口编号(0 或 1)

【参数范围】

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

【备注】

在 Macaron, Pudding 中，参数范围如下。
X 轴偏移：0~31
Y 轴偏移：0~31
X 轴(偏移+大小) <= 32
Y 轴(偏移+大小) <= 32

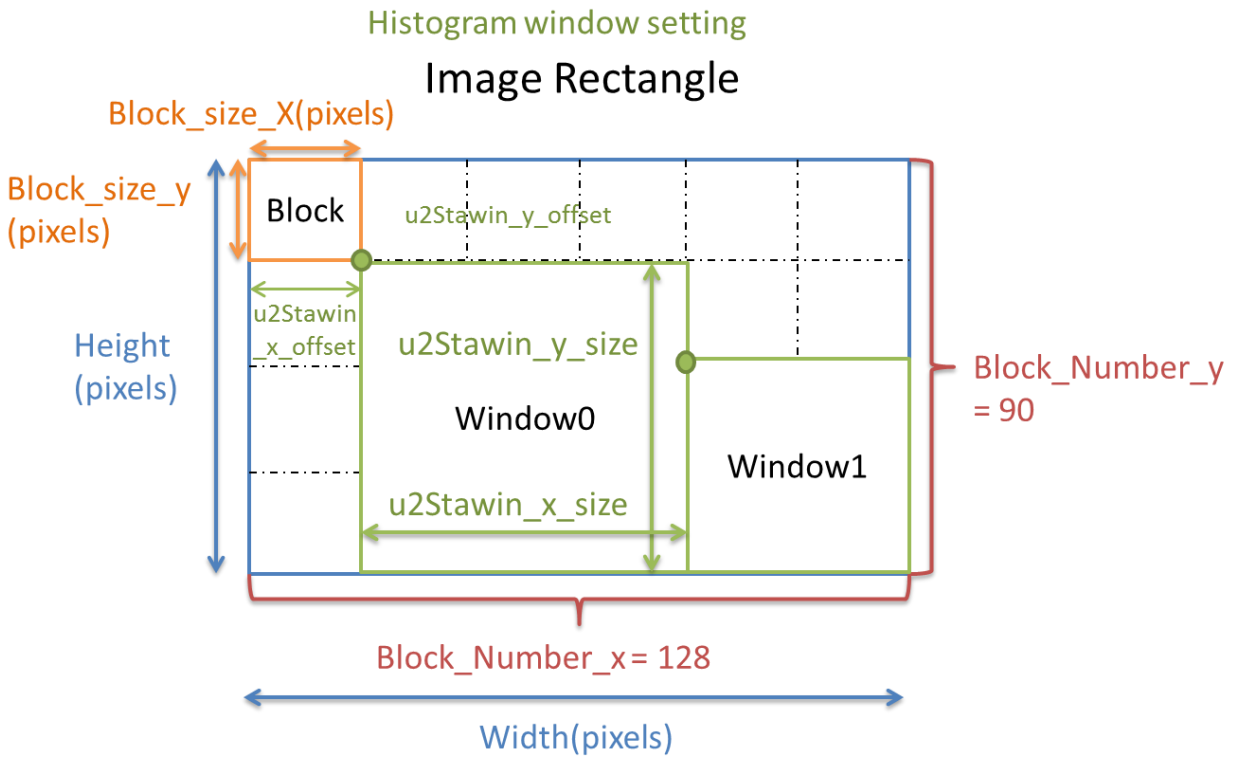


Image Size = Width * Height
Block_size_X = Width / 128
Block_size_Y = Height / 90

18. MI_ISP 设置 AWB SAMPLING

MI_ISP_CUS3A_SetAwbSampling

【目的】
设定 AWB 取样大小

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

【描述】
调用此界面设置 AWB 取样大小

【成员】

参数名称	描述
nChannel	视讯输入信道号码(一般为 0)
SizeX	X 轴上取多少个取样点
SizeY	Y 轴上取多少个取样点
IncRatio	对统计值乘上一个比例

【返回值】

返回值	描述
MI_RET_SUCCESS	成功
MI_RET_FAIL	失败

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

【注意事项】
整张照片已经切割成 128*90 格，每个格子中取其中 SizeX * SizeY 个 pixels 做 AWB 取样。
SizeX：SizeX >=4 && SizeX <= Block_size_X
SizeY：SizeY >=2 && SizeY <= Block_size_Y

当 AE 亮度很低的时候，可将统计值乘上一个比例(IncRatio)，避免统计值为 0

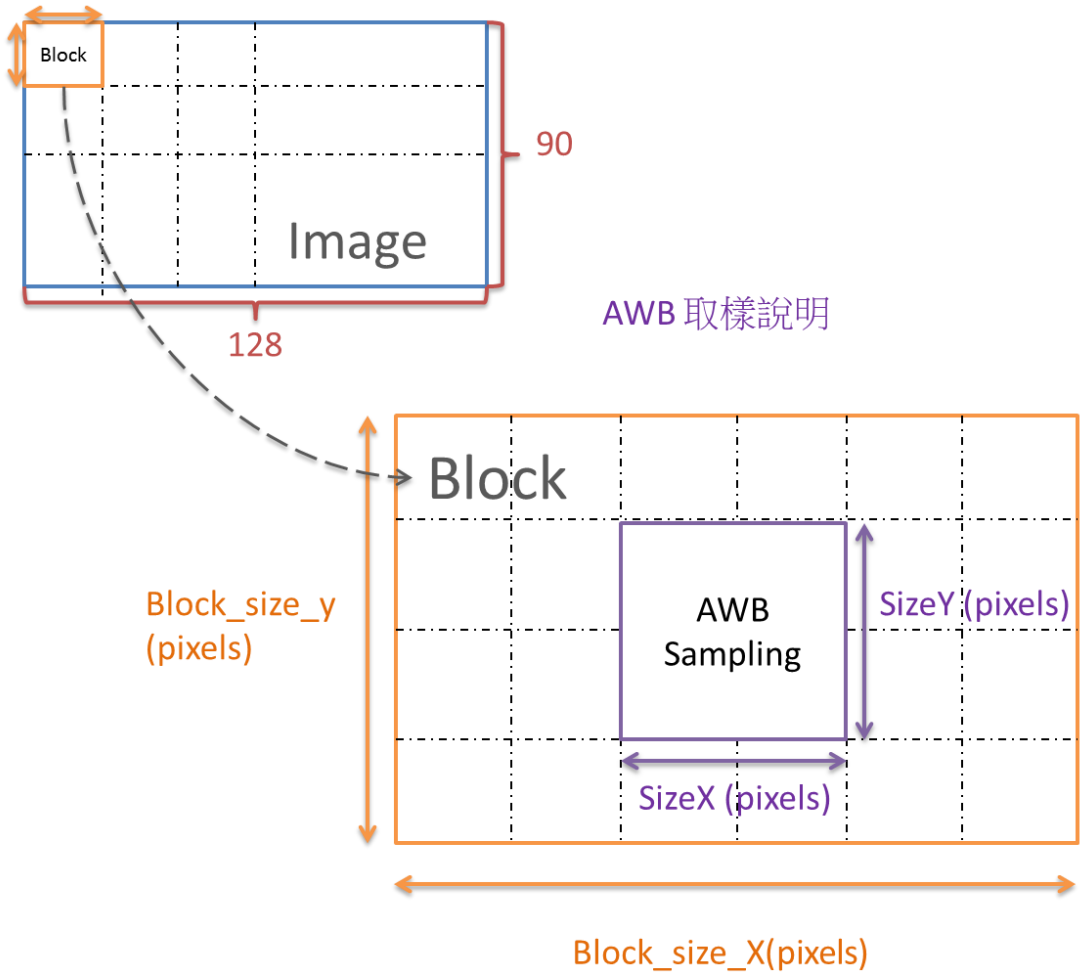


Image resolution = Width * Height
Block_size_X = Width / 128
Block_size_Y = Height / 90

19. MI_ISP 设置 AF FILTER

MI_ISP_CUS3A_SetAFFilter

【目的】
设定 AF Filter 参数

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

【描述】
调用此界面设置 AF Filter 参数

【参数】

参数名称	描述
nChannel	视讯输入信道号码(一般为 0)
data	AF Filter 参数

【返回值】

返回值	描述
MI_RET_SUCCESS	成功
MI_RET_FAIL	失败

CusAFFilter_t

【说明】ISP AF 硬件统计值

【定义】

```
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;

     //for Pudding only
    MI_U16 u16IIR1_e1_en;
    MI_U16 u16IIR1_e1_a0;
    MI_U16 u16IIR1_e1_a1;
    MI_U16 u16IIR1_e1_a2;
    MI_U16 u16IIR1_e1_b1;
    MI_U16 u16IIR1_e1_b2;
    MI_U16 u16IIR1_e2_en;
    MI_U16 u16IIR1_e2_a0;
    MI_U16 u16IIR1_e2_a1;
    MI_U16 u16IIR1_e2_a2;
    MI_U16 u16IIR1_e2_b1;
    MI_U16 u16IIR1_e2_b2;

    MI_U16 u16IIR2_e1_en;
    MI_U16 u16IIR2_e1_a0;
    MI_U16 u16IIR2_e1_a1;
    MI_U16 u16IIR2_e1_a2;
    MI_U16 u16IIR2_e1_b1;
    MI_U16 u16IIR2_e1_b2;
    MI_U16 u16IIR2_e2_en;
    MI_U16 u16IIR2_e2_a0;
```

```
    MI_U16 u16IIR2_e2_a1;
    MI_U16 u16IIR2_e2_a2;
    MI_U16 u16IIR2_e2_b1;
    MI_U16 u16IIR2_e2_b2;
} CusAFFilter_t;
```

【成员】

IIR参数如下

名称	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 为 IIR High，IIR2 default 为 IIR Low

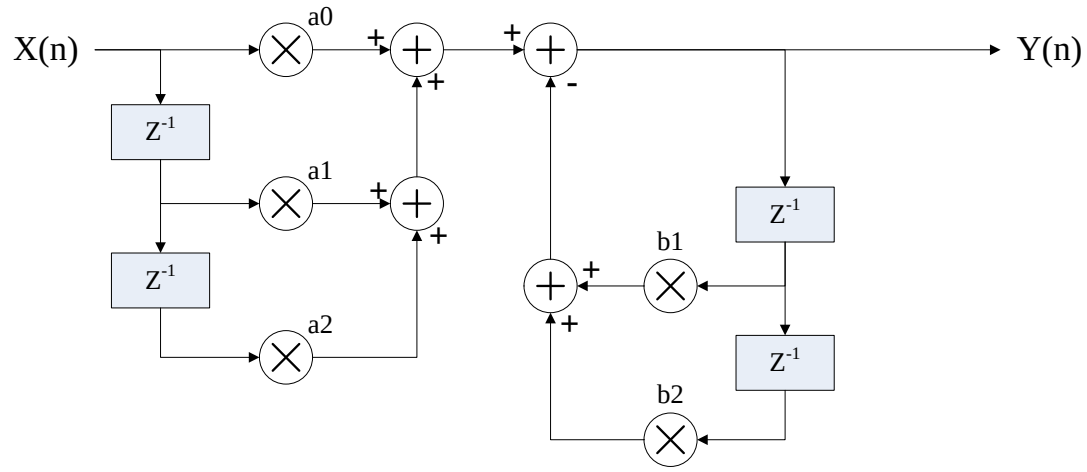
//Pudding only

名称	bit 表示	描述	IIR1 default	IIR2 default
a0	S+9	a0 乘法器	37	15
a1	S+10	a1 乘法器	0	0
a2	S+9	a2 乘法器	-37	-15
b1	S+7	b1 乘法器	83	-79
b2	S+7	b2 乘法器	40	44
1st_low_clip	10	X(n) input low clip	0	0
1st_high_clip	10	X(n) input high clip	1023	1023
2nd_low_clip	10	Y(n) output low clip	0	0
2nd_high_clip	10	Y(n) output high clip	1023	1023
e1_en	1	Extra1 enable	1	1
e1_a0	S+9	a0 乘法器	37	15
e1_a1	S+10	a1 乘法器	0	0
e1_a2	S+9	a2 乘法器	-37	-15
e1_b1	S+7	b1 乘法器	-54	-115
e1_b2	S+7	b2 乘法器	34	55

名称	bit 表示	描述	IIR1 default	IIR2 default
e2_en	1	Extra2 enable	1	1
e2_a0	S+9	a0 乘法器	32	14
e2_a1	S+10	a1 乘法器	0	0
e2_a2	S+9	a2 乘法器	-32	-14
e2_b1	S+7	b1 乘法器	14	-91
e2_b2	S+7	b2 乘法器	0	37

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

IIR 系数，架构图如下：



【需求】

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

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

20. MI_ISP 设置 AF FILTER SQUARE

MI_ISP_CUS3A_SetAFFilterSq

【目的】
设定 AF Filter Square 参数

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

【描述】
调用此界面设置 AF Filter Square 参数

【参数】

参数名称	描述
nChannel	视讯输入信道号码(一般为 0)
data	AF Filter Square 参数

【返回值】

返回值	描述
MI_RET_SUCCESS	成功
MI_RET_FAIL	失败

CusAFFilterSq_t

【说明】ISP AF Square 硬件统计值

【定义】

```
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;
```

【成员】

名称	描述
bFIRYSatEn	FIR Filter Y 阀值控制
u16FIRYThd	If bFIRYSatEn = 1 则 pixel 亮度小于 u16FIRYThd 时，就会列入 fir filter 计算中，且回传大于 u16FIRYThd 的 pixel 个数(于 AF 统计值中)，数值范围: 0 ~ 1023。
bIIRSquareAccEn	IIR Filter 增强控制
bFIRSquareAccEn	FIR Filter 增强控制
u16IIR1Thd	IIR Filter Output = IIR Filter Output – IIRThd。数值范围: 0 ~ 1023。
u16IIR2Thd	IIR Filter Output = IIR Filter Output - IIRThd。数值范围: 0 ~ 1023。
u16FIRHThd	FIR Filter Output = FIR Filter Output - FIR Thd。数值范围: 0 ~ 1023。
u16FIRVThd	FIR Filter Output = FIR Filter Output - FIR Thd。数值范围: 0 ~ 1023。
u8AFTblX[12]	针对 IIR 与 FIR Filter，做一个 non-linear 的 mapping u8AFTblX 为横轴，节点为二的幂次方累加，累加起来需大于 1024。数值范围: 0 ~ 15。
u16AFTblY[13]	针对 IIR 与 FIR Filter，做一个 non-linear 的 mapping u8AFTblY 为纵轴，数值范围: 0 ~ 8191。

SquareACC 使用时机，针对中高对比度的粗边反差，做放大的动作。且低于 Thd 的值，会砍为 0，用来抗低照下的噪点。

参考设定 (统计值放大效果)
u8AFTblX = 6,7,7,6,6,6,7,6,6,7,6,6
u16AFTblY = 0,1,53,249,431,685,1023,1999,2661,3455,5487,6749,8191

参考设定 (统计值保持原状)
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, Pudding)
.so：libmi_isp_twinkie.so / libmi_isp_pretzel.so / libmi_isp.so (for Macaron, Pudding)

【备注】
Twinkie 不支援这支 API。

21. MI_ISP 设置 AF ROI MODE

MI_ISP_CUS3A_SetAFRoiMode

【目的】
设定 AF 统计值的模式

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

【描述】
调用此界面设置 AWB 统计值来源

【参数】

参数名称	描述
nChannel	视讯输入信道号码(一般为 0)
data	AF 统计值的模式

【返回值】

返回值	描述
MI_RET_SUCCESS	成功
MI_RET_FAIL	失败

CusAFRoiMode_t

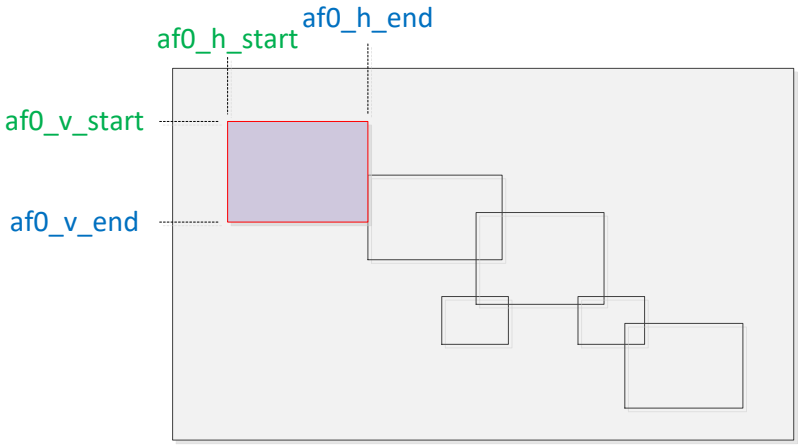
【描述】
设定 AF 统计值的模式

【定义】
```
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;
```

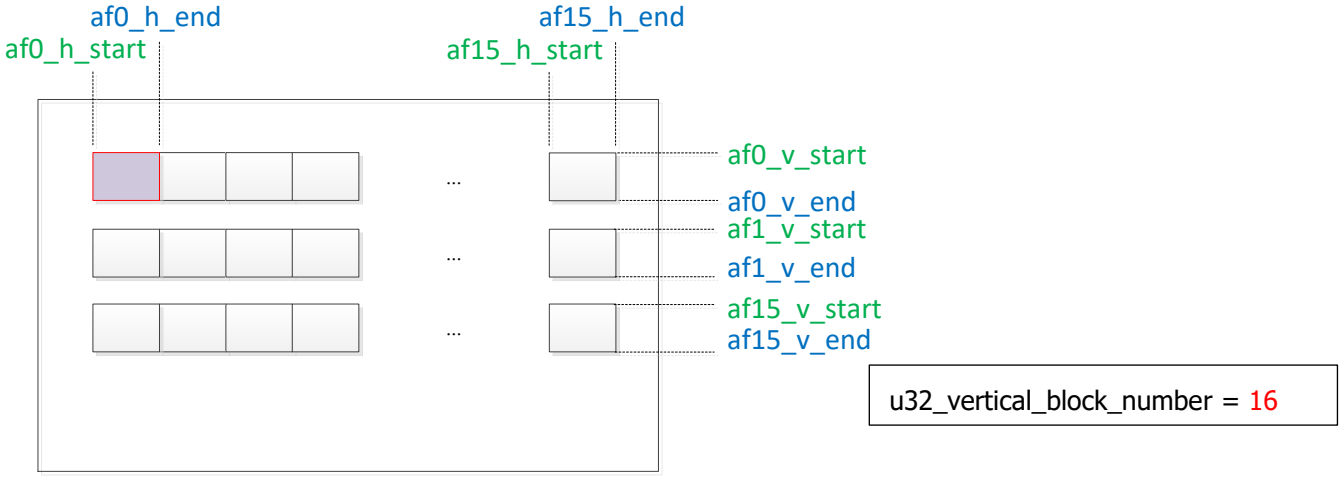
【成员】

名称	描述
mode	AF_ROI_MODE_NORMAL: 可切为 16 组 ROI，window size 与位置可随意分割，默认为此模式。 AF_ROI_MODE_MATRIX: 可切为16 * N组ROI，window size与位置稍有限制，但可切较多区块。(N= u32_vertical_block_number)
u32_vertical_block_number	If mode = AF_ROI_MODE_MATRIX 可切为 16 * N 组 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



横轴，固定有16个
纵轴，会依据u32_vertical_block_number，来决定有几组16个ROI，上图为
u32_vertical_block_number=16，就有 16*16 组ROI

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

【备注】
Twinkie 不支援这支 API。

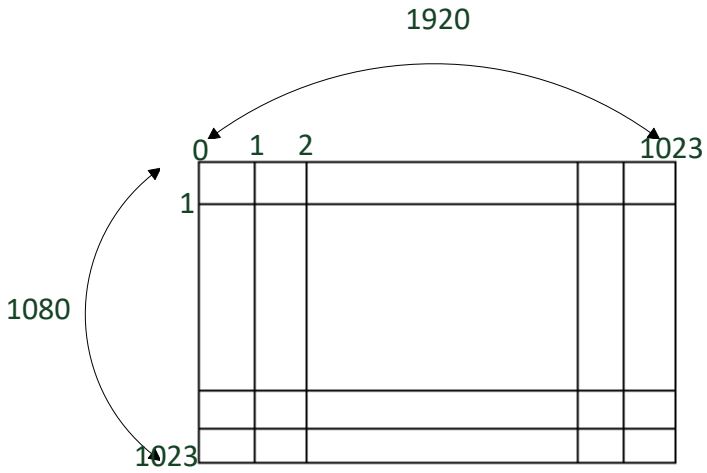
22. MI_ISP 设置 AF WINDOW

MI_ISP_CUS3A_SetAFWindow

【目的】
设定 AF Window

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

【描述】
AF window 在 AF_Init 阶段设置,将 CMOS sensor image 切成 1024*1024 坐标格来设置 AF window



坐标与实际 crop range 换算公式:

Image width * X /1024 = 真实 X 坐标
Image_height * Y/1024 = 真实 Y 坐标

Ex:
AF window 设置为 :
Start_X = 458
End_X = 566
Start_Y = 418
End_Y = 606

则在 1920x1080 的 CMOS sensor 下设置的 AF window 坐标即为:
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

【参数】

参数名称	描述
nChannel	视讯输入信道号码(一般为 0)
data	AF Window 设置

【返回值】

返回值	描述
MI_RET_SUCCESS	成功
MI_RET_FAIL	失败

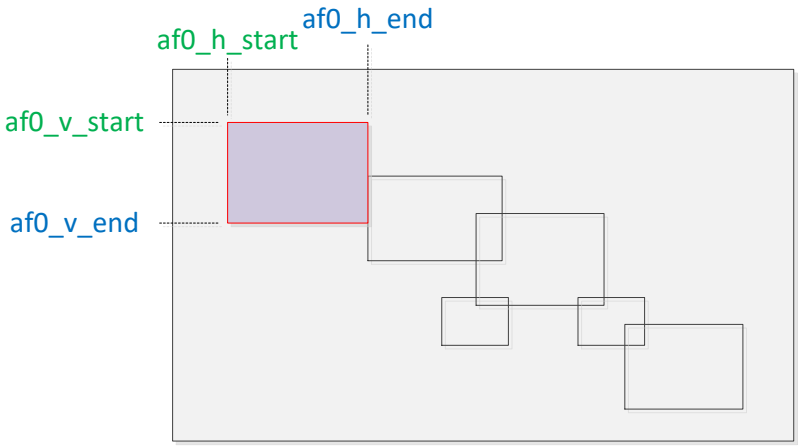
CusAFWin_t

```
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;
```

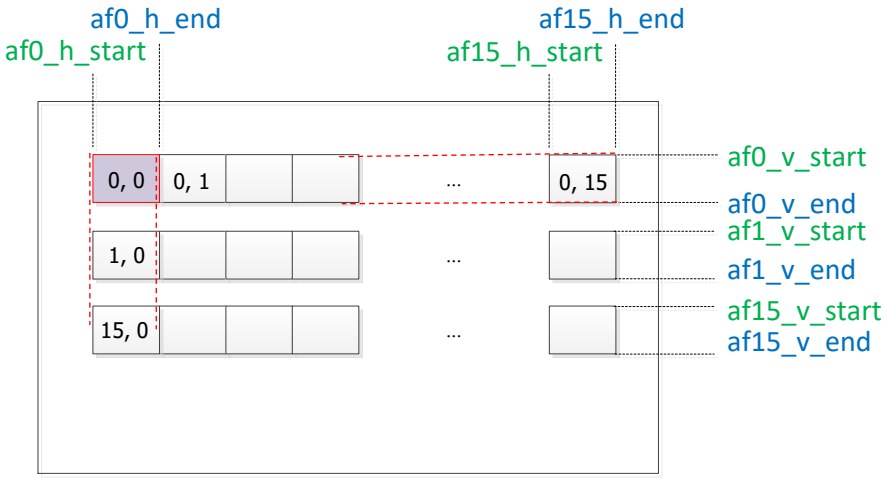
【参数】

参数名称	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 时，StrX, StrY, EndX, EndY 就是代表该 Window Index 的位置



在 AF ROI Mode = AF_ROI_MODE_MATRIX时，StrX, EndX,就是代表横轴window切割的位置，StrY, EndY就是代表纵轴切割的位置



使用限制:

AF ROI Mode = AF_ROI_MODE_NORMAL

- 各自 window 宽高设置上，可互相重迭
- h/v end > h/v start

AF ROI Mode = AF_ROI_MODE_MATRIX

- 各自 window 宽设置上，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 设定，由 af0_h_start, h_end 决定
- win(0,1), win(1,1), ..., win(15,1) 使用共同的 h_start, h_end 设定，由 af1_h_start, h_end 决定，其他以此推论
- win(0,0), win(0,1), ..., win(0,15) 使用共同的 v_start, v_end 设定，由 af0_v_start, v_end 决定
- win(1,0), win(1,1), ..., win(1,15) 使用共同的 v_start, v_end 设定，由 af1_v_start, v_end 决定，其他以此推论

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

23. GET CUST3A VERSION

参考 isp_cus3a_if.h，版号定义如下

```
#define CUS3A_VER_STR "CUS3A_V1.1"
#define CUS3A_VER_MAJOR 1
#define CUS3A_VER_MINOR 1

unsigned int CUS3A_GetVersion(char* pVerStr);
```

1. 可藉由查看 header file 获得版号
2. 可藉由 CUS3A_GetVersion 获得版号

24. TWINKIE/PRETZEL/MACARON/PUDDING 支援差异列表

此章节描述 Twinkie、Pretzel、Macaron 与 Pudding 所支持的功能列表

参数名称	Twinkie	Pretzel	Macaron
AE 统计值	128*90	128*90	32*32
AE hist2	支援	支援	不支援
AF 统计值	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

参数名称	Pudding		
AE 统计值	32*32		
AE hist2	不支援		
AF 统计值	IIR(35), FIR(35), Luma(32), YSat(22)		
MI_ISP_CUS3A_SetAEWindowBlockNumber	不支援		
MI_ISP_CUS3A_SetAEHistogramWindow	偏移+大小限制: 最大 32*32		
MI_ISP_CUS3A_SetAFFilter	支援，新增 Extra 参数		
MI_ISP_CUS3A_SetAFFilterSq	支援		
MI_ISP_CUS3A_SetAFRoiMode	支援		