

# V5 EVE Application API Reference Manual

Document Version：v1.0

Release Date： 2018-09-27

**Trademark statement**



Note

The products, services or features you purchase are subject to the terms and conditions of the company and all or part of the products, services or features described in this document may not be covered by your purchase or use. Unless otherwise agreed by the contract, Allwinner Company makes no representations or warranties, express or implied, regarding the contents of this document.

The contents of this document may be updated from time to time due to product version upgrades or other reasons. Unless otherwise agreed, this document is provided as a guide only, and all statements, information, and recommendations in this document are not warranties of any kind, express or implied.

# Foreword

## Overview

This document provides SDK guidance for EVE (Embedded Visual Engine) development and design engineers. EVE is suitable for rigid target detection and can be widely used in face, pedestrian, vehicle and other applications requiring fast target detection.

## Product version

The product version corresponding to this document.

| Chip name | Product version |
|---|---|
| V5 | V1.0 |
| | |
| | |



## Reader

This document (this guide) is intended primarily for the following engineers:

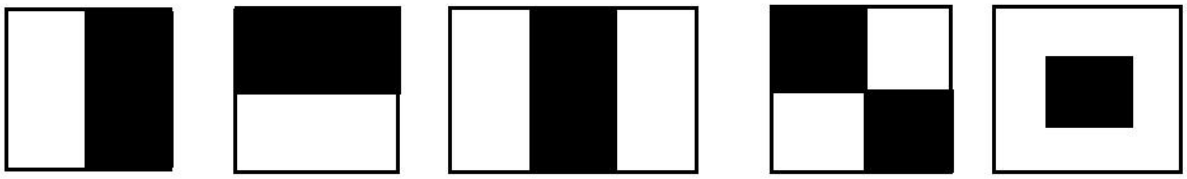
Software development engineer using EVE for development

## Glossary

API：Application Programming Interface

SDK： Software Development Kit

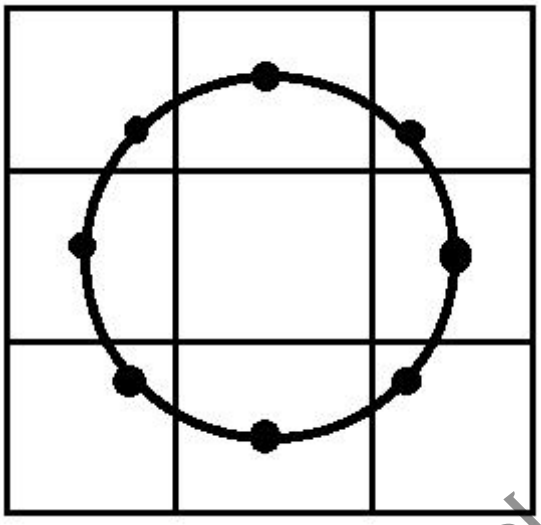
HAAR: Haar-like feature is a commonly used feature description operator in the field of computer vision.，Invented by the revelation of the one-dimensional haar wavelet，The Haar features are divided into edge features, linear features, center features and diagonal features to form feature templates. The feature templates are held in white and black, and the feature values of the template are defined as white rectangular pixels minus black rectangular pixels.。



Edge feature Edge feature Line feature Diagonal feature Center feature

LBP: local binary pattern,it is an operator that describes the local features of an image. LBP features have

significant advantages such as gray invariance and rotation invariance.。



Local binary pattern operator

EVE: Embedded Visual acceleration Engine

ICF: integral channel features，by performing various linear and non-linear changes to the input image, such as summation, histogram, haar-like, HOG, and variations thereof can be quickly calculated from the integral map.

| Version | Revision date | Modify the content |
|---|---|---|
| AW_AI_EVE_EVENT_1.0.0.20170901_Release | 2017.08.11 | Draft |
| AW_AI_EVE_EVENT_1.0.0.20170927_Release | 2017.09.27 | Added EVE DMA data source and EVE operator synchronization mode, 0-asynchronous, 1-synchronous mode |
| AW_AI_EVE_EVENT_1.0.0.20170927_Release | 2018.09.27 | Release |
| | | |

# Contents





# 1 Overview

## 1.1 Overview

EVE (Embedded Vision Engine) is a high-performance hardware detection operator for rigid target detection with fast speed, low power consumption and customizable detection targets. The algorithm provided by EVE can customize the classifier to support any small deformation rigid target detection (face, vehicle, license plate, pedestrian, head, head and shoulders and other key targets).

## 1.2 EVE Functional Description

### 1.2.1 Detail and Performance

1. 360p(640*360) detection speed>30FPS (working frequency >300MHZ)
2. Support classic HAAR feature classifier detection, the number of features up to 3200;
3. Supports maximum resolution 4K input and internal scaling to support region of interest detection;
4. Support 4-channel integral graph calculation, processing 1.3 billion features per second;
5. Support for 3-channel feature calculation;
6. Support user-defined target size, support 432 different kinds of detection frames;
7. Supports minimum 64x64 pixel single image detection;
8. Low power consumption, 360p full image detection <57mv;
9. The classifier visualizes the design and provides a complete set of training, debugging, testing and evaluation tools
10. Customizable classifiers to support arbitrary small deformation rigid target detection;
11. Supports minimum 20 pixel face detection at 360p resolution;
12. Face detection rate is greater than 95% within 30 degrees;
13. Support left and right maximum deflection 80 degrees
14. Support 16 face real-time detection and tracking (25FPS), support up to 128 faces;
15. Support LBP feature detection;
16. Support integral channel feature SICF detection;

### 1.2.2 Related concepts

1. Handle

When a user creates an EVE task using an operator, the system assigns a handle to each task, and the configuration parameter eventType is used to identify the type of the EVE event.

2. Return EVE status flags

After the EVE task is created, the user can get the information about he EVE task enabled status by setting the parameter AW_STATUS_E.



3．Return EVE result flag

After the user creates the EVE task, the parameter pstResult can be used to query whether the EVE task is completed.

### 1.2.3 Use Reference

The second chapter of the reference manual introduces the EVE function in detail, and describes the syntax rules, parameter settings, and function return values of the function in detail. The related functions are equipped with corresponding routines. The user can call the corresponding operator interface to create an EVE task according to the requirements. The third chapter gives a detailed description of the data type, value range, and data structure of the EVE operator.

# 2 EVE Application API Reference

| Face detection function list | Functional description |
|---|---|
| AW_AI_EVE_Event_GetAlgoVersion | Query the EVE event version number |
| AW_AI_EVE_Event_Init | Initialize the EVE event |
| AW_AI_EVE_Event_UnInit | Deinitialize EVE event |
| AW_AI_EVE_Event_Process | Handling a frame of EVE events |
| AW_AI_EVE_Event_SetEventParam | Set EVE event parameters |
| AW_AI_EVE_Event_SetEveSourceAddress | Set the EVE event video source physical/virtual address |
| AW_AI_EVE_Event_GetLastError | Get the error code of the EVE event |

## 2.1 AW_AI_EVE_Event_GetAlgoVersion

【Description】

Query the EVE event version number

【Grammar】

AW_EXPORTS AW_STATUS_E AW_AI_EVE_Event_GetAlgoVersion( AW_S8 *pVersion );

【Parameter】

| Parameter name | Description | Input/Output |
|---|---|---|
| pVersion | Version number string, such as AW_AI_EVE_EVENT_V5_1.0.0.20180927_release | Output |

【Return value】

| Parameter name | Description | Return value |
|---|---|---|
| AW_STATUS_E | Status | ERROR：AW_STATUS_ERROR<br>OK：AW_STATUS_OK<br>SKIP：AW_STATUS_SKIP |

【Reference header file】

aw_ai_eve_event_interface.h

【Example】

```
#include "aw_ai_eve_event_interface.h"
#include <stdio.h>
#include <stdlib.h>

    AW_S32 status;
    AW_S8 cVersion[128];
    status = AW_AI_EVE_Event_GetAlgoVersion(cVersion);
    if(AW_STATUS_ERROR == status)
    {
        printf("AW_AI_EVE_Event_GetAlgoVersion failure!\n");
        return -1;
    }
    else
    {
        printf("AW_AI_EVE_Event_GetAlgoVersion version is %s!\n", cVersion);
    }
```

【Remarks】

No

## 2.2 AW_AI_EVE_Event_Init

【Description】

Initialize the EVE event

【Grammar】

AW_EXPORTS AW_HANDLE AW_AI_EVE_Event_Init(AW_AI_EVE_CTRL_S *pEVECtrl);

【Parameter】

| Parameter name | Description | Input/Output |
|---|---|---|
| pEVECtrl | EVE operator configuration parameters | Input |

【Return value】

| Parameter name | Description | Return value |
|---|---|---|
| AW_HANDLE | Handle of the EVE event | NULL： NULL<br>Not empty: non-null |

| | | |
|---|---|---|
| | | pointer |

【Reference header file】

aw_ai_eve_event_interface.h

【Example】

```
#include "aw_ai_eve_event_interface.h"
#include <stdio.h>
#include <stdlib.h>
    //Initialize the EVE event
    AW_AI_EVE_CTRL_S sEVECtrl;
    sEVECtrl.addrInputType = AW_AI_EVE_ADDR_INPUT_VIR;//输入虚拟地址
    sEVECtrl.classifierNum = 7;
    sEVECtrl.scale_factor = 1;
    sEVECtrl.mScanStageNo = 10;
    sEVECtrl.yStep = 3;
    sEVECtrl.xStep0 =1;
    sEVECtrl.xStep1 =4;
    sEVECtrl.mRltNum = AW_AI_EVE_MAX_RESULT_NUM;
    sEVECtrl.mMidRltNum = 0;
    sEVECtrl.mMidRltStageNo = 50;
    sEVECtrl.rltType = AW_AI_EVE_RLT_OUTPUT_DETAIL;
    sEVECtrl.mDmaOut.s16Width = 640;
    sEVECtrl.mDmaOut.s16Height = 360;
    sEVECtrl.mPyramidLowestLayel.s16Width = 640;
    sEVECtrl.mPyramidLowestLayel.s16Height = 360;
    sEVECtrl.dmaSrcSize.s16Width = 640;
    sEVECtrl.dmaSrcSize.s16Height = 360;
    sEVECtrl.dmaDesSize.s16Width = 640;
    sEVECtrl.dmaDesSize.s16Height = 360;
    sEVECtrl.dmaRoi.s16X = 0;
    sEVECtrl.dmaRoi.s16Y = 0;
    sEVECtrl.dmaRoi.s16Width = sEVECtrl.dmaDesSize.s16Width;
    sEVECtrl.dmaRoi.s16Height = sEVECtrl.dmaDesSize.s16Height;
```

```
sEVECtrl.classifierNum = 1;
sEVECtrl.classifierPath[0].path = (AW_S8*)"frontface.ld";
sEVECtrl.classifierPath[0].key    = (AW_U8*)awKeyNew;

sEVECtrl.dmaCallBackFunc = &dmaCallBackFunc;
sEVECtrl.dma_pUsr = dma_pUsr;
AW_HANDLE hEveEvent = AW_AI_EVE_Event_Init(&sEVECtrl);
if(AW_NULL == hEveEvent)
{
    printf("AW_AI_EVE_Event_Init failure!\n");
    return -1;
}
else
{
    printf("AW_AI_EVE_Kernel_Init finish!\n");
}
```

【Remarks】

NO

## 2.3 AW_AI_EVE_Event_UnInit

【Description】

Deinitialize (release) the EVE event

【Grammar】

AW_EXPORTS AW_STATUS_E AW_AI_EVE_Event_UnInit( AW_HANDLE hHandle);

【Parameter】

| Parameter name | Description | Input/Output |
|---|---|---|
| hHandle | Handle of the EVE event | Input |

【Return value】

| Parameter name | Description | Return value |
|---|---|---|
| AW_STATUS_E | Status | ERROR ： AW_STATUS_ERROR<br>OK：AW_STATUS_OK |

| | | |
|---|---|---|
| | | SKIP : AW_STATUS_SKIP |

【Reference header file】

aw_ai_eve_event_interface.h

【Example】

```
#include "aw_ai_eve_event_interface.h"
#include <stdio.h>
#include <stdlib.h>

//hEveEvent is the return handle for AW_AI_EVE_Event_Init

status = AW_AI_EVE_Event_UnInit(hEveEvent);
if(AW_STATUS_ERROR == status)
{
    printf("AW_AI_EVE_Event_UnInit failure!, errorcode = %d\n",
AW_AI_EVE_Event_GetLastError(hEveEvent));
    return -1;
}
```

【Remarks】

NO

## 2.4 AW_AI_EVE_Event_Process

【Description】

Handling a frame of EVE events

【Grammar】

```
AW_EXPORTS AW_STATUS_E AW_AI_EVE_Event_Process( AW_HANDLE hHandle,
AW_IMAGE_S *pstImage, AW_S64 u32TimeStamp, AW_AI_EVE_EVENT_RESULT_S *pstResult );
```

【Parameter】

| Parameter name | Description | Input/Output |
|---|---|---|
| hHandle | Handle of the EVE event | Input |
| pstImage | One frame image | Input |
| u32TimeStamp | Current frame timestamp | Input |
| pstResult | EVE event results | Output |



【Return value】

| Parameter name | Description | Return value |
| --- | --- | --- |
| AW_STATUS_E | Status | ERROR：AW_STATUS_ERROR<br>OK：AW_STATUS_OK<br>SKIP：AW_STATUS_SKIP |

【Reference header file】

aw_ai_eve_event_interface.h

【Example】

```
#include "aw_ai_eve_event_interface.h"
#include <stdio.h>
#include <stdlib.h>

AW_IMAGE_S image; //Single frame image data
AW_U32 timestamp = 1;
AW_AI_EVE_EVENT_RESULT_S faceres; //EVE event structure

image.mWidth = 720;
image.mHeight = 576;
image.mPixelFormat = PIXEL_FORMAT_RGB_8BPP;
image.mpVirAddr[0] = (AW_PVOID)malloc(720*576*2);
image.mpVirAddr[1] = 0;
image.mpVirAddr[2] = 0;
image.mPhyAddr[0] = 0;
image.mPhyAddr[1] = 0;
image.mPhyAddr[2] = 0;
image.mStride[0] = 720;
image.mStride[1] = 0;
image.mStride[2] = 0;

//Set EVE virtual source address
AW_AI_EVE_Event_SetEveSourceAddress(pCap->hEveEvent, image.mVirAddr[0]);
```

```
//Processing single frame images，hEveEvent is the return handle for AW_AI_EVE_Event_Init
status = AW_AI_EVE_Event_Process(hEveEvent, &image,  timestamp, &faceres);
if(AW_STATUS_ERROR == status)
{
    printf("AW_AI_EVE_Event_Process failure!, errorcode = %d\n", AW_AI_EVE_Event_GetLastError(hEveEvent));
}
```

【Remarks】

NO

## 2.5 AW_AI_EVE_Event_SetEventParam

【Description】

Set EVE event parameters

【Grammar】

AW_EXPORTS AW_STATUS_E AW_AI_EVE_Event_SetEventParam( AW_HANDLE hHandle, AW_AI_EVE_EVENT_TYPE_E eventType, AW_PVOID pEventParam);

【Parameter】

| Parameter name | Description | Input/Output |
|---|---|---|
| hHandle | Handle of the EVE event | Input |
| eventType | Type of EVE event | Input, refer to AW_AI_EVE_EVENT_TYPE_E |
| pEventParam | Structure parameters corresponding to the EVE event | Input |

【Return value】

| Parameter name | Description | Return value |
|---|---|---|
| AW_STATUS_E | Status | ERROR：AW_STATUS_ERROR<br>OK：AW_STATUS_OK<br>SKIP： |

| | | |
|---|---|---|
| | | AW_STATUS_SKIP |

【Reference header file】

aw_ai_eve_event_interface.h

【Example】

```
#include "aw_ai_eve_event_interface.h"
#include <stdio.h>
#include <stdlib.h>

//set face event parameter
AW_AI_EVE_FACEDET_PARAM_S facedetparam;
    facedetparam.sRoiSet.s32RoiNum = 1;
    facedetparam.sRoiSet.sID[0] = 1;
    facedetparam.sRoiSet.sRoi[0].s16Left   = 0;
    facedetparam.sRoiSet.sRoi[0].s16Top    = 0;
    facedetparam.sRoiSet.sRoi[0].s16Right  = 640;
    facedetparam.sRoiSet.sRoi[0].s16Bottom = 360;
    facedetparam.s32ClassifyFlag = 0; //close
    facedetparam.s32MinFaceSize  = 20;
    facedetparam.s32OverLapCoeff = 20;
    facedetparam.s32MaxFaceNum = 128;
    facedetparam.s32MergeThreshold = 3;
    facedetparam.s8Cfgfile = AW_NULL;
    facedetparam.s8Weightfile = AW_NULL;

//hEveEvent is the return handle for AW_AI_EVE_Event_Init
status = AW_AI_EVE_Event_SetEventParam(hEveEvent, AW_AI_EVE_EVENT_FACEDETECT,
(AW_PVOID)&facedetparam);
if(AW_STATUS_ERROR == status)
{
    printf("AW_AI_EVE_Event_SetEventParam failure!, errorcode = %d\n",
AW_AI_EVE_Event_GetLastError(hEveEvent));
    return -1;
}
```



【Remarks】

NO

## 2.6 AW_AI_EVE_Event_SetEveSourceAddress

【Description】

Set the video source address of EVE

【Grammar】

AW_EXPORTS AW_STATUS_E AW_AI_EVE_Event_SetEveSourceAddress( AW_HANDLE hHandle, void *pSourceAddr);

【Parameter】

| Parameter name | Description | Input/Output |
|---|---|---|
| hHandle | Handle of the EVE event | Input |
| pSourceAddr | Video source physical/virtual address | Input, refer to the AW_AI_EVE_ADDR_INPUT_TYPE and AW_AI_EVE_CTRL_S structures. For example, the addrInputType variable of the AW_AI_EVE_CTRL_S structure is configured as a physical address, where the physical address must be entered. |

【Return value】

| Parameter name | Description | Return value |
|---|---|---|
| AW_STATUS_E | Status | ERROR：AW_STATUS_ERROR<br>OK：AW_STATUS_OK<br>SKIP：AW_STATUS_SKIP |

【Reference header file】

aw_ai_eve_event_interface.h



【Example】

```
#include "aw_ai_eve_event_interface.h"
#include <stdio.h>
#include <stdlib.h>

  //Initialize the EVE event
  AW_AI_EVE_CTRL_S sEVECtrl;
  sEVECtrl.addrInputType = AW_AI_EVE_ADDR_INPUT_PHY;//Set physical address
  ..... //Initialize other parameters


//Set EVE physical address
AW_AI_EVE_Event_SetEveSourceAddress(pCap->hEveEvent, pCap->image.mPhyAddr[0]);
```

【Remarks】

AW_AI_EVE_Event_SetEveSourceAddress must be called before AW_AI_EVE_Event_Process. If the video source physical or virtual address is unchanged, it can be called only once; if the video source physical or virtual address changes in polling mode, it must be called once when the physical or virtual address changes.

## 2.7 AW_AI_EVE_Event_GetLastError

【Description】

Set EVE kernel parameters

【Grammar】

```
AW_EXPORTS AW_S32    AW_AI_EVE_Event_GetLastError( AW_HANDLE hHandle );
```

【Parameter】

| Parameter name | Description | Input/Output |
|---|---|---|
| hHandle | Handle of the EVE event | Input |

【Return value】

| Parameter name | Description | Return value |
|---|---|---|
| AW_S32 | ERROR code description | Refer to the ERROR code table aw_ai_errorcode.h |

【Reference header file】

aw_ai_eve_event_interface.h



【Example】

NO

【Remarks】

NO

## 2.8 AW_AI_EVE_Event_SetEveDMAExecuteMode

【Description】

Set the EVE kernel data source DMA handling and EVE operator execution mode

【Grammar】

AW_STATUS_E AW_AI_EVE_Event_SetEveDMAExecuteMode( AW_HANDLE hHandle, AW_S32 dmaMode)

【Parameter】

| Parameter name | Description | Input/Output |
|---|---|---|
| hHandle | Handle of the EVE event | Input |
| dmaMode | Data source DMA handling and EVE operator execution mode, support asynchronous and synchronous modes, 0-asynchronous, 1-synchronous, reference complex: AW_AI_EVE_DMA_EXECUTE_TYPE | Input |

【Return value】

| Parameter name | Description | Return value |
|---|---|---|
| AW_STATUS_E | Status | ERROR：AW_STATUS_ERROR<br>OK：AW_STATUS_OK<br>SKIP：AW_STATUS_SKIP |

【Reference header file】

aw_ai_eve_event_interface.h

aw_ai_eve_type.h

【Example】

```
#include "aw_ai_eve_event_interface.h"
#include <stdio.h>
#include <stdlib.h>
    //Initialize the EVE event
    AW_AI_EVE_CTRL_S sEVECtrl;
    sEVECtrl.addrInputType = AW_AI_EVE_ADDR_INPUT_VIR;//Set virtual address
    sEVECtrl.classifierNum = 7;
    sEVECtrl.scale_factor = 1;
    sEVECtrl.mScanStageNo = 10;
    sEVECtrl.yStep = 3;
    sEVECtrl.xStep0 =1;
    sEVECtrl.xStep1 =4;
    sEVECtrl.mRltNum = AW_AI_EVE_MAX_RESULT_NUM;
    sEVECtrl.mMidRltNum = 0;
    sEVECtrl.mMidRltStageNo = 50;
    sEVECtrl.rltType = AW_AI_EVE_RLT_OUTPUT_DETAIL;
    sEVECtrl.mDmaOut.s16Width = 640;
    sEVECtrl.mDmaOut.s16Height = 360;
    sEVECtrl.mPyramidLowestLayel.s16Width = 640;
    sEVECtrl.mPyramidLowestLayel.s16Height = 360;
    sEVECtrl.dmaSrcSize.s16Width = 640;
    sEVECtrl.dmaSrcSize.s16Height = 360;
    sEVECtrl.dmaDesSize.s16Width = 640;
    sEVECtrl.dmaDesSize.s16Height = 360;
    sEVECtrl.dmaRoi.s16X = 0;
    sEVECtrl.dmaRoi.s16Y = 0;
    sEVECtrl.dmaRoi.s16Width = sEVECtrl.dmaDesSize.s16Width;
    sEVECtrl.dmaRoi.s16Height = sEVECtrl.dmaDesSize.s16Height;

    sEVECtrl.classifierPath.path[0] = "./classifier/frontfacenew.ld";
    sEVECtrl.classifierPath.path[1] = "./classifier/halfdown.ld";
```

```
sEVECtrl.classifierPath.path[2] = "./classifier/rotleft.ld";
sEVECtrl.classifierPath.path[3] = "./classifier/rotright.ld";
sEVECtrl.classifierPath.path[4] = "./classifier/fullprofileold.ld";
sEVECtrl.classifierPath.path[5] = "./classifier/faceprofileclip.ld";
sEVECtrl.classifierPath.path[6] = "./classifier/smallface.ld";

sEVECtrl.dmaCallBackFunc = &dmaCallBackFunc;
sEVECtrl.dma_pUsr = dma_pUsr;
AW_HANDLE hEveEvent = AW_AI_EVE_Event_Init(&sEVECtrl);
if(AW_NULL == hEveEvent)
{
    printf("AW_AI_EVE_Event_Init failure!\n");
    return -1;
}
else
{
    printf("AW_AI_EVE_Kernel_Init finish!\n");
}

//Set DMA mode
AW_STATUS_E status;
    Status = AW_AI_EVE_Event_SetEveDMAExecuteMode(hEveEvent,
AW_AI_EVE_DMA_EXECUTE_SYNC);//Synchronous mode
```

【Remarks】

Must be called after AW_AI_EVE_EVENT_Init



# 3 Data structure reference

## 3.1 Data type

【Description】

Data type definition

【Definition】

typedef unsigned char AW_U8;
typedef unsigned short AW_U16;
typedef unsigned int AW_U32;
typedef char AW_S8;
typedef short AW_S16;
typedef int AW_S32;
typedef float AW_FLOAT;
typedef char AW_CHAR;
typedef void* AW_HANDLE;
typedef void* AW_PVOID;

【Member】

| Variable | Type | Ranges |
|---|---|---|
| AW_U8 | unsigned char | 0~255 |
| AW_U16 | unsigned short | 0~65535 |
| AW_U32 | unsigned int | 0~0xffffffff |
| AW_S8 | char | -128~127 |
| AW_S16 | short | -32768~32767 |
| AW_S32 | int | -(0xffffffff+1)~ 0xffffffff |
| AW_FLOAT | float | -2^128 ~ +2^128 |
| AW_DOUBLE | double | -2^1024 ~ +2^1014 |
| AW_CHAR | char | -128~127 |
| AW_HANDLE | Handle | |
| AW_PVOID | pointer | |

【Remarks】

## 3.2 Macro definition

【Description】

Macro definition

【Definition】

```
#define    AW_NULL                0L
#define    AW_SUCCESS             0
#define    AW_FAILURE             (-1)
#define    AW_MAX_POLYGON_NUM     32 //max polygon nums
#define    AW_MAX_TRAJECT_LEN     40 //max trajectory nums
#define    AW_MAX_TGT_NUM         64 //max target nums
#define    AW_MAX_EVT_NUM         128    //max event nums
#define    AW_MAX_BLOB_NUM        128
#define    AW_MAX_ROI_NUM         8      //max roi number
```

【Member】

| Member | Description | Ranges |
|---|---|---|
| AW_NULL | Null pointer | 0L |
| AW_SUCCESS | OK | 0 |
| AW_FAILURE | Failure | -1 |
| AW_MAX_POLYGON_NUM | Maximum number of vertices of the polygon | 32 |
| AW_MAX_TRAJECT_LEN | Maximum number of track points | 40 |
| AW_MAX_TGT_NUM | Maximum number of target arrays | 64 |
| AW_MAX_EVT_NUM | Maximum number of event arrays | 128 |
| AW_MAX_BLOB_NUM | Maximum number of connected body arrays | 128 |

| | | |
|---|---|---|
| AW_MAX_ROI_NUM | Maximum number of regions of interest | 8 |

【Remarks】

## 3.3 Key data structure

| Member | Description |
|---|---|
| AW_AI_EVE_CTRL_S | EVE kernel variable setting structure |
| AW_IMAGE_S | Single frame image structure |
| AW_TARGET_SET_S | Target structure |
| AW_EVENT_SET_S | Target trigger event structure |
| AW_AI_EVE_RESULT_S | EVE event result structure |
| AW_AI_EVE_FACEDET_PARAM_S | Face detection parameter structure |
| AW_AI_EVE_KERNEL_CTRL_S | EVE kernel control structure |
| AW_AI_EVE_KERNEL_PARAM_S | EVE kernel parameter structure |

### AW_POINT_S

【Description】

Coordinate point information structure

【Definition】

```
typedef struct _AW_POINT_S
{
    AW_S16   s16X;
    AW_S16   s16Y;
}AW_POINT_S, *lpAW_POINT_S;
```

【Variable】

| Variable | Description | Type | Ranges |
|---|---|---|---|
| s16X | X-axis | short | -32767~32767 |
| s16Y | Y-axis | short | -32767~32767 |

【Remarks】

Refer to the aw_ai_common_type.h header file.

# AW_RECT_S

【Description】

Rectangular information structure

【Definition】

```
typedef struct _AW_RECT_S
{
    AW_S16 s16Left;          //!< letf x
    AW_S16 s16Top;           //!< top y
    AW_S16 s16Right;         //!< right x
    AW_S16 s16Bottom;        //!< bottom y
}AW_RECT_S, *lpAW_RECT_S;
```

【Variable】

| Variable | Description | Type | Ranges |
|---|---|---|---|
| s16Left | The X coordinate of the upper left corner of the rectangle | Signed short | 0~32767 |
| s16Top | The Y coordinate of the upper left corner of the rectangle | Signed short | 0~32767 |
| s16Right | The X coordinate of the lower right corner of the rectangle | Signed short | 0~32767 |
| s16Bottom | The Y coordinate of the lower right corner of the rectangle | Signed short | 0~32767 |

【Reference header file】

aw_ai_common_type.h

【Remarks】

# AW_BLOB_S

【Description】

Single connected body information

【Definition】

```
typedef struct _AW_BLOB_S
{
    AW_S16 s16X;
    AW_S16 s16Y;
    AW_S16 s16Width;
    AW_S16 s16Height;
}AW_BLOB_S, *lpAW_BLOB_S;
```

【Variable】

| Variable | Description | Type | Ranges |
|---|---|---|---|
| s16X | The X coordinate of Connected body minimum circumscribed rectangle upper left corner | Signed short | 0~32767 |
| s16Y | The Y coordinate of Connected body minimum circumscribed rectangle upper left corner | Signed short | 0~32767 |
| s16Width | Minimum external rectangle width of the connected body | Signed short | 0~32767 |
| s16Height | Minimum external rectangle height of the connected body | Signed short | 0~32767 |

【Reference header file】

aw_ai_common_type.h

【Remarks】

## AW_BLOB_SET_S

【Description】

Connected body array

【Definition】

```
typedef struct _AW_BLOB_SET_S
{
    AW_S32 s32Num;
    AW_BLOB_S blob[AW_MAX_BLOB_NUM];
}AW_BLOB_SET_S, *lpAW_BLOB_SET_S;
```

【Variable】

| Variable | Description | Type | Ranges |
|---|---|---|---|
| s32Num | Number of connected bodies | int | -(0xffffffff+1)~ 0xffffffff |
| blob | Connected body array | AW_BLOB_S | |

【Reference header file】

aw_ai_common_type.h

【Remarks】

Maximum number of connected bodies

## AW_SIZE_S

【Description】

Size Information

【Definition】

```
typedef struct _AW_SIZE_S
{
    AW_S16 s16Width;
    AW_S16 s16Height;
}AW_SIZE_S, *lpAW_SIZE_S;
```

【Variable】

| Variable | Description | Type | Ranges |
|---|---|---|---|

| | | | |
|---|---|---|---|
| s16Width | Width, in pixels | short | |
| s16Height | Height, in pixels | short | |

【Reference header file】

aw_ai_common_type.h

【Remarks】

## AW_LINE_S

【Description】

Line

【Definition】

```
typedef struct _AW_LINE_S
{
    AW_POINT_S    stPStart;
    AW_POINT_S    stPEnd;
}AW_LINE_S, *lpAW_LINE_S;
```

【Variable】

| Variable | Description | Type | Ranges |
|---|---|---|---|
| stPStart | Starting point of the line | AW_POINT_S | |
| stPEnd | Endting point of the line | AW_POINT_S | |

【Reference header file】

aw_ai_common_type.h

【Remarks】

## AW_POLYGON_S

【Description】



Polygon Description

【Definition】

```
typedef struct _AW_POLYGON_S
{
    AW_S32 s32Num;                  //!< polygon point num
    AW_POINT_S astPoints[AW_MAX_POLYGON_NUM]; //!< polygon point array
}AW_POLYGON_S, *lpAW_POLYGON_S;
```

【Variable】

| Variable | Description | Type | Ranges |
|---|---|---|---|
| s32Num | Number of polygons | int | |
| astPoints | Polygon point array | AW_POINT_S | |

【Reference header file】

aw_ai_common_type.h

【Remarks】

## AW_BOOL

【Description】

Bool Description

【Definition】

```
typedef enum {
    AW_FALSE    = 0,
    AW_TRUE     = 1,
} AW_BOOL;
```

【Variable】

| Variable | Description | Ranges |
|---|---|---|
| AW_FALSE | Logical false | 0 |
| AW_TRUE | Logical true | 1 |

【Reference header file】

aw_ai_common_type.h

【Remarks】

## AW_STATUS_E

【Description】

Status Description

【Definition】

```
typedef enum
{
    AW_STATUS_ERROR = 0,        //!< Error
    AW_STATUS_OK    = 1,        //!< Ok
    AW_STATUS_SKIP  = 2,        //!< Skip
}AW_STATUS_E;
```

【Variable】

| Variable | Description | Ranges |
|---|---|---|
| AW_STATUS_ERROR | Error | 0 |
| AW_STATUS_OK | OK | 1 |
| AW_STATUS_SKIP | SKIP | 2 |

【Reference header file】

aw_ai_common_type.h

【Remarks】

## AW_AI_EVE_ADDR_INPUT_TYPE

【Description】

Video source address

【Definition】

```
enum AW_AI_EVE_ADDR_INPUT_TYPE{
    AW_AI_EVE_ADDR_INPUT_PHY,
    AW_AI_EVE_ADDR_INPUT_VIR
};
```

【Variable】

| Variable | Description | Ranges |
|---|---|---|
| AW_AI_EVE_ADDR_INPUT_PHY | Video source physical address | Physical address pointer |
| AW_AI_EVE_ADDR_INPUT_VIR | Video source virtual address | Virtual address pointer |

【Reference header file】

aw_ai_eve_type.h

【Remarks】

## AW_AI_EVE_DMA_EXECUTE_TYPE

【Description】

DMA video source and EVE operator execution mode

【Definition】

```
enum AW_AI_EVE_DMA_EXECUTE_TYPE
{
    AW_AI_EVE_DMA_EXECUTE_ASYNC, //asynchronous
    AW_AI_EVE_DMA_EXECUTE_SYNC    //Synchronize
};
```

【Variable】

| Variable | Description | Ranges |
|---|---|---|
| AW_AI_EVE_DMA_EXECUTE_ASYNC | Asynchronous execution mode | 0 |
| AW_AI_EVE_DMA_EXECUTE_SYNC | Synchronous execution mode | 1 |

【Reference header file】

aw_ai_eve_type.h

【Remarks】

In the synchronous mode, the DMA transfer and the operator are sequentially executed, and the DMA transfer time is calculated into the EVE running time;

In the asynchronous mode, the DMA data transfer adopts the PingPong buffer form, and the DMA and the EVE operator execute asynchronously.

## AW_AI_EVE_CTRL_S

【Description】

EVE middleware initialization related control structure

【Definition】

```
typedef struct _AW_AI_EVE_CTRL_S
{
    AW_U8 addrInputType;
    AW_U8 classifierNum;
    AW_U8 scale_factor;
    AW_U8 yStep;
    AW_U8 xStep0;
    AW_U8 xStep1;
    AW_U8 mScanStageNo;
    AW_U8 rltType;
    AW_U8 mMidRltStageNo;
    AW_U32 mMidRltNum;
    AW_U32 mRltNum;

    // Size
    AW_SIZE_S mDmaOut;
    AW_SIZE_S mPyramidLowestLayel;
    AW_SIZE_S dmaSrcSize;
    AW_BLOB_S dmaRoi;
    AW_SIZE_S dmaDesSize;

    AW_CLSPATH_S classifierPath[AW_AI_EVE_MAX_CLASSIFIER_NUM];
```



```
    void(*dmaCallBackFunc)(void* pUsr);
    void* dma_pUsr;
}AW_AI_EVE_CTRL_S;
```

【Variable】

| Variable | Description | Type | Ranges |
|---|---|---|---|
| addrInputType | Input video source address pointer | Physical/virtual address | Reference AW_AI_EVE_ADDR_INPUT_TYPE |
| classifierNum | Number of classifiers | unsigned char | [1,16] |
| scale_factor | Image pyramid minimum scaling factor | unsigned char | [0,3], corresponding to 1.1, 1.2, 1.3, and 1.4 scale |
| yStep | Y-direction scan step size | unsigned char | [1,4]，Defaults 2 |
| xStep0 | X-direction scanning Minimum step size | unsigned char | [1,4]，Defaults 1 |
| xStep1 | X-direction scanning Maximum step size | unsigned char | [1,4],And greater thanxStep0，Defaults 4 |
| mScanStageNo | - | unsigned char | Defaults 7 |
| rltType | Open intermediate result tag | unsigned char | The default is 0 (no output) |
| mMidRltStageNo | Open intermediate result classifier hierarchy | unsigned char | The default is 0. If rltType is 1, mMidRltStageNo must set the range [0, the maximum number of classifiers] |
| mMidRltNum | Maximum number of intermediate results | unsigned int | [0, 2048] |
| mRltNum | Maximum number of features for a single | unsigned int | <=2048 |

| | | | |
|---|---|---|---|
| | classifier | | |
| mDmaOut | DMA maximum output size | AW_SIZE_S | s16Width: [0,1920]; Step: 32<br>s16Height: [0,1920] |
| mPyramidLowestLayel | Maximum size of the bottom layer of the image pyramid | AW_SIZE_S | s16Width: [64,640]; Step: 32<br>s16Height: [64,640] |
| dmaSrcSize | DMA source size | AW_SIZE_S | s16Width: [0,4096]; Step: 32<br>s16Height: [0,4096] |
| dmaRoi | DMA module ROI | AW_BLOB_S | s16X: [0,dmaSrcSize.s16Width];Step: 32<br>s16Y: [0,dmaSrcSize.s16Height]<br>s16Width: [64, dmaSrcSize.s16Width] ;Step: 32<br>s16Height: [64,dmaSrcSize.s16Height]<br>s16X+ s16Width: [64, dmaSrcSize.s16Width]<br>s16Y+ s16Height: [64,dmaSrcSize. |

| | | | |
|---|---|---|---|
| | | | s16H<br>eight] |
| classifierPath | Each classifier path and key structure | AW_CLSPATH_S | Path: classifier path<br>Key: the encryption key corresponding to the classifier |
| dmaCallBackFunc | Dma callback function | Void pointer | |
| dma_pUsr | User-defined pointer | Void pointer | Used with dmaCallBackFunc |

【Reference header file】

aw_ai_eve_type.h

【Remarks】

(1) The red font is an advanced parameter.，In order to input the parameter value of the user, the value can be set according to the actual application scenario.。

(2)scale_factor，users should consider the actual situation to set, and if the setting is too small, it may affect the performance; if the setting is over the assembly error, if the setting is too small, it may affect the performance.

## PIXEL_FORMAT_E

【Description】

Two-dimensional image structure

【Definition】

```
typedef enum PIXEL_FORMAT_E
{
    PIXEL_FORMAT_RGB_1BPP = 0,
    PIXEL_FORMAT_RGB_2BPP,
    PIXEL_FORMAT_RGB_4BPP,
    PIXEL_FORMAT_RGB_8BPP,
    PIXEL_FORMAT_RGB_444,
```



```
PIXEL_FORMAT_RGB_4444,
PIXEL_FORMAT_RGB_555,
PIXEL_FORMAT_RGB_565,
PIXEL_FORMAT_RGB_1555,

/*    9 reserved */
PIXEL_FORMAT_RGB_888,
PIXEL_FORMAT_RGB_8888,

PIXEL_FORMAT_RGB_PLANAR_888,
PIXEL_FORMAT_RGB_BAYER_8BPP,
PIXEL_FORMAT_RGB_BAYER_10BPP,
PIXEL_FORMAT_RGB_BAYER_12BPP,
PIXEL_FORMAT_RGB_BAYER_14BPP,

PIXEL_FORMAT_RGB_BAYER,              /* 16 bpp */

PIXEL_FORMAT_YUV_A422,
PIXEL_FORMAT_YUV_A444,

PIXEL_FORMAT_YUV_PLANAR_422,
PIXEL_FORMAT_YUV_PLANAR_420,      //YU12

PIXEL_FORMAT_YUV_PLANAR_444,

PIXEL_FORMAT_YUV_SEMIPLANAR_422,      //NV16
PIXEL_FORMAT_YUV_SEMIPLANAR_420,      //NV12
PIXEL_FORMAT_YUV_SEMIPLANAR_444,

PIXEL_FORMAT_UYVY_PACKAGE_422,
PIXEL_FORMAT_YUYV_PACKAGE_422,
PIXEL_FORMAT_VYUY_PACKAGE_422,
PIXEL_FORMAT_YCbCr_PLANAR,
```

```
    PIXEL_FORMAT_SINGLE,

    PIXEL_FORMAT_YVU_PLANAR_420,      //YV12
    PIXEL_FORMAT_YVU_SEMIPLANAR_422,      //NV61
    PIXEL_FORMAT_YVU_SEMIPLANAR_420,      //NV21

    PIXEL_FORMAT_BUTT
} PIXEL_FORMAT_E;
```

【Variable】

| Variable | Description | Ranges |
|---|---|---|
| PIXEL_FORMAT_RGB_1BPP | | |
| PIXEL_FORMAT_RGB_2BPP | | |
| PIXEL_FORMAT_RGB_4BPP | | |
| PIXEL_FORMAT_RGB_8BPP | | |
| PIXEL_FORMAT_RGB_444 | | |
| PIXEL_FORMAT_RGB_4444 | | |
| PIXEL_FORMAT_RGB_555 | | |
| PIXEL_FORMAT_RGB_565 | | |
| PIXEL_FORMAT_RGB_1555 | | |
| PIXEL_FORMAT_RGB_888, | | |
| PIXEL_FORMAT_RGB_8888 | | |
| PIXEL_FORMAT_RGB_PLANAR_888 | | |
| PIXEL_FORMAT_RGB_BAYER_8BPP | | |
| PIXEL_FORMAT_RGB_BAYER_10BPP | | |
| PIXEL_FORMAT_RGB_BAYER_12BPP | | |
| PIXEL_FORMAT_RGB_BAYER_14BPP | | |
| PIXEL_FORMAT_RGB_BAYER | | |
| PIXEL_FORMAT_YUV_A422 | | |

| | | |
|---|---|---|
| PIXEL_FORMAT_YUV_A444 | | |
| PIXEL_FORMAT_YUV_PLANAR_422 | | |
| PIXEL_FORMAT_YUV_PLANAR_420 | | |
| PIXEL_FORMAT_YUV_PLANAR_444 | | |
| PIXEL_FORMAT_YUV_SEMIPLANAR_422 | | |
| PIXEL_FORMAT_YUV_SEMIPLANAR_420 | | |
| PIXEL_FORMAT_YUV_SEMIPLANAR_444 | | |
| PIXEL_FORMAT_UYVY_PACKAGE_422 | | |
| PIXEL_FORMAT_YUYV_PACKAGE_422 | | |
| PIXEL_FORMAT_VYUY_PACKAGE_422 | | |
| PIXEL_FORMAT_YCbCr_PLANAR | | |
| PIXEL_FORMAT_SINGLE | | |
| PIXEL_FORMAT_YVU_PLANAR_420 | | |
| PIXEL_FORMAT_YVU_SEMIPLANAR_422 | | |
| PIXEL_FORMAT_YVU_SEMIPLANAR_420 | | |
| PIXEL_FORMAT_BUTT | | |

【Reference header file】

aw_ai_common_type.h

【Remarks】



# AW_IMAGE_S

【Description】

Two-dimensional image structure

【Definition】

```
typedef struct AW_IMAGE_S
{
    PIXEL_FORMAT_E  mPixelFormat;   //image pixel format, ref PIXEL_FORMAT_E
    AW_U32      mPhyAddr[3];    //physical address
    AW_PVOID    mpVirAddr[3]; //virtual   address
    AW_U32  mWidth;             //image wdith
    AW_U32  mHeight;            //image height
    AW_U32  mStride[3];         //image stride
}AW_IMAGE_S, *lpAW_IMAGE_S;
```

【Variable】

| Variable | Description | Type | Ranges |
|---|---|---|---|
| mPixelFormat | Pixel format | PIXEL_FORMAT_E | - |
| mPhyAddr[3] | Image data physical address | unsigned int | 0~0xFFFFFFFF |
| mpVirAddr[3] | Image data virtual address | VOID * pointer type | 0~0xFFFFFFFF |
| mWidth | Image width | unsigned int | 0~0xFFFFFFFF |
| mHeight | Image height | unsigned int | 0~0xFFFFFFFF |
| mStride[3] | Image span | unsigned int | 0~0xFFFFFFFF |

【Reference header file】

aw_ai_common_type.h

【Remarks】

PIXEL_FORMAT_E

## AW_TGT_TYPE_E

【Description】

Target Type Consortium

【Definition】

```
typedef enum
{
    AW_TGT_TYPE_UNKNOWN     = 0,    //!< unknown
    AW_TGT_TYPE_HUMAN       = 1,    //!< Human
    AW_TGT_TYPE_VEHICLE     = 2     //!< Vehicle
}AW_TGT_TYPE_E;
```

【Variable】

| Variable | Description | Ranges |
|---|---|---|
| AW_TGT_TYPE_UNKNOWN | Undefined target Type | 0 |
| AW_TGT_TYPE_HUMAN | Human | 1 |
| AW_TGT_TYPE_VEHICLE | Vehicle | 2 |

【Reference header file】

aw_ai_common_type.h

【Remarks】

## AW_TGT_TRAJECT_S

【Description】

Target track information

【Definition】

```
typedef struct
{
    AW_S32 length;                              //!< Track length
    AW_POINT_S      points[AW_MAX_TRAJECT_LEN];   //!< Track point array
} AW_TGT_TRAJECT_S;
```

【Variable】

| Variable | Description | Type | Ranges |
|---|---|---|---|
| length | Track length | int | 0~0xFFFFFFFF |
| points | Track point coordinate sequence | AW_POINT_S | |

【Reference header file】

aw_ai_common_type.h

【Remarks】

AW_MAX_TRAJECT_LEN:Maximum number of track points

## AW_ROI_SET_S

【Description】

Region of interest collection

【Definition】

```
typedef struct _AW_ROI_SET_S
{
    AW_S32      s32RoiNum;
    AW_S32      sID[AW_MAX_ROI_NUM];    //roi id
    AW_RECT_S   sRoi[AW_MAX_ROI_NUM]; //roi, support maximize 8 rois.
}AW_ROI_SET_S, *lpAW_ROI_SET_S;
```

【Variable】

| Variable | Description | Type | Ranges |
|---|---|---|---|
| s32RoiNum | Number of regions of interest | int | [1,8] |
| sID | Unique identification number | int | 0~0xFFFFFFFF |

| | | | |
|---|---|---|---|
| | set for the region of interest | | |
| sRoi | Region of interest rectangular box collection | AW_RECT_S | |

【Reference header file】

aw_ai_common_type.h

【Remarks】

## AW_TARGET_S

【Description】

Single target information

【Definition】

```
typedef struct _AW_TARGET_S
{
    AW_U32 u32ID;                   //!< Target ID
    AW_S32 u32Type;                 //!< Target type，Ref AW_TGT_TYPE
    AW_POINT_S stPoint;             //!< Target location
    AW_RECT_S stRect;               //!< Target rectangle
    AW_S32 s32AreaPix;              //!< Target area (unit: pixel)
    AW_S32 s32Area;                 //!< Target area (unit: square centimeter)
    AW_S32 s32Direct;               //!< Direction of movement (unit: angle)
    AW_FLOAT fSpeed;                //!< Movement speed (unit: km / hour)
    AW_TGT_TRAJECT_S stTraject;     //!< Target track
} AW_TARGET_S, *lpAW_TARGET_S;
```

【Variable】

| Variable | Description | Type | Ranges |
|---|---|---|---|
| u32ID | Target unique ID | unsigned int | - |
| u32Type | Target type，reference AW_TGT_TYPE | int | - |

| | | | |
|---|---|---|---|
| stPoint | Target center point coordinates | AW_POINT_S | 0~0xFFFFFFFF |
| stRect | Target rectangle | AW_RECT_S | 0~0xFFFFFFFF |
| s32AreaPix | Target area (unit: pixel) | int | 0~0xFFFFFFFF |
| s32Area | Target area (unit: square centimeter) | int | 0~0xFFFFFFFF |
| s32Direct | Direction of movement (unit: angle) | int | |
| fSpeed | Movement speed (unit: km / hour) | float | |
| stTraject | Target track | AW_TGT_TRAJECT_S | |

【Reference header file】

aw_ai_common_type.h

【Remarks】

## AW_TARET_SET_S

【Description】

Target array

【Definition】

```
typedef struct _AW_TARGET_SET_S
{
    AW_S32 s32TargetNum;                    //!< Number of effective targets
    AW_TARGET_S astTargets[AW_MAX_TGT_NUM]; //!< Target array
}AW_TARGET_SET_S, *lpAW_TARGET_SET_S;
```

【Variable】

| Variable | Description | Type | Ranges |
|---|---|---|---|
| s32TargetNum | Number of targets | int | |

| | | | |
|---|---|---|---|
| astTargets | Target information array | AW_TARGET_S | |

【Reference header file】

aw_ai_common_type.h

【Remarks】

## AW_EVENT_S

【Description】

Single event Description

【Definition】

```
typedef struct _AW_EVENT_S
{
    AW_U32 u32Type;          //!< Event Type
    AW_U32 u32ID;            //!< Event identifier
    AW_U32 u32Status;        //!< Event Status
    AW_U32 u32Zone;          //!< Event occurrence area
    AW_U32 u32TgtID;         //!< Target identifier
    AW_U8  u8Data[16];       //!< Event private data
}AW_EVENT_S, *lpAW_EVENT_S;
```

【Variable】

| Variable | Description | Type | Ranges |
|---|---|---|---|
| u32Type | Event Type,reference AW_AI_EVE_EVENT_TYPE_E | unsigned int | |
| u32ID | Event identifier | unsigned int | |
| u32Status | Event Status, reference AW_EVT_STATUS | unsigned int | |
| u32Zone | Event occurrence area | unsigned int | |
| u32TgtID | Target identifier | unsigned int | |
| u8Data | Event private data | unsigned char | |

【Reference header file】

aw_ai_common_type.h

【Remarks】

## AW_EVENT_SET_S

【Description】

Event array Description

【Definition】

```
typedef struct _AW_EVENT_SET_S
{
    AW_S32 s32EventNum;                          //!< Number of valid events
    AW_EVENT_S astEvents[AW_MAX_EVT_NUM];        //!< Event array
} AW_EVENT_SET_S, *lpAW_EVENT_SET_S;
```

【Variable】

| Variable | Description | Type | Ranges |
|---|---|---|---|
| s32EventNum | Number of events | int | |
| astEvents | Event array | AW_EVENT_S | |

【Reference header file】

aw_ai_common_type.h

【Remarks】

最大支持事件个数 AW_MAX_EVT_NUM

## AW_AI_EVE_RESULT_S

【Description】

Maximum number of supported events

【Definition】

typedef struct _AW_AI_EVE_RESULT_S

```
{
    AW_TARGET_SET_S    sTarget;
    AW_EVENT_SET_S     sEvent;
}AW_AI_EVE_RESULT_S, *lpAW_AI_EVE_RESULT_S;
```

【Variable】

| Variable | Description | Type | Ranges |
|---|---|---|---|
| sTarget | Target array | AW_TARGET_SET_S | |
| sEvent | Trigger event array | AW_EVENT_SET_S | |

【Reference header file】

aw_ai_common_type.h

【Remarks】

## AW_AI_EVE_EVENT_TYPE_E

【Description】

EVE Event Type Union

【Definition】

```
typedef enum
{
    AW_AI_EVE_EVENT_FACEDETECT = 3001,    //face detect
    AW_AI_EVE_EVENT_HEADDETECT = 3002,    //reserved
}AW_AI_EVE_EVENT_TYPE_E;
```

【Variable】

| Variable | Description | Ranges |
|---|---|---|
| AW_AI_EVE_EVENT_FACEDETECT | Face detection event | 3001 |

| | | |
|---|---|---|
| AW_AI_EVE_EVENT_HEAD DETECT | Head detection event (not supported at this time) | 3002 |

【Reference header file】

aw_ai_eve_event_interface.h

【Remarks】

## AW_AI_EVE_FACEDET_PARAM_S

【Description】

Face detection parameter information

【Definition】

```
typedef struct _AW_AI_EVE_EVENT_FACEDET_PARAM_S
{
    AW_ROI_SET_S sRoiSet;              //facedetect roi, support maximize 8 rois
    AW_S32       s32OverLapCoeff;      //overlap coeff, [1, 100]%, default: [20]%
    AW_S32       s32MinFaceSize;       //minimize face   size,[20]pixel
    AW_S32       s32MergeThreshold;    //merger threshold, [2, 5], default: 3
    AW_S32       s32MaxFaceNum;        //maximize face num, [1, 128], default: 128
    AW_S32       s32ClassifyFlag;      //the second Classify flag, 0-off, 1-on, reseverd, default: 0
    AW_S8        *s8Cfgfile;           //face classifier cfg file, reserved, default[NULL]
    AW_S8        *s8Weightfile;        //face classifier weight file，reserved, default[NULL]
}AW_AI_EVE_EVENT_FACEDET_PARAM_S, *lpAW_AI_EVE_EVENT_FACEDET_PARAM_S;
```

【Variable】

| Variable | Description | Type | Ranges |
|---|---|---|---|
| sRoiSet | facedetect roi | AW_ROI_SET_S | Supports up to 8 ROIs |
| s32OverLapCoeff | overlap coeff | int | [1,100], default 20 |
| s32MinFaceSize | Minimum face detection size in pixels | int | Minimum support (20,20) pixels |
| s32MaxFaceNu | Maximum number | int | [1,128], default |

| | | | |
|---|---|---|---|
| m | of face detections for a single image | | 128 |
| s32MergeThreshold | Classifier detection result merge threshold | int | [2, 5], the default is 3, the larger the value, the higher the credibility of the output face, but there may be a decrease in the face detected. |
| s32ClassifyFlag | Secondary classifier switch [reserved] | int | 0-off,1-on, default off 0 |
| s8Cfgfile | Secondary classifier configuration parameter file full path [reserved] | *char | The default is NULL |
| s8Weightfile | Secondary classifier weight parameter file full path [reserved] | *char | The default is NULL |

【Reference header file】

aw_ai_eve_event_interface.h

【Remarks】

## AW_AI_EVE_KERNEL_CTRL_S

【Description】

EVE kernel control parameter structure

【Definition】

```
typedef struct _AW_AI_EVE_KERNEL_PARAMETER_S{
    AW_U8                       classifierId;
    AW_U32                      cmd;
    AW_AI_EVE_KERNEL_PARAM_S    sValue;
}AW_AI_EVE_KERNEL_CTRL_S;
```

【Variable】

| Variable | Description | Type | Ranges |
|---|---|---|---|
| classifierId | Classifier corresponding identification number | unsigned char | [0,classifierNum-1] |
| cmd | Control instruction | unsigned int | See AW_AI_EVE_KERNEL_CMD |
| sValue | Control the value of Variable | AW_AI_EVE_KERNEL_PARAM_S | See AW_AI_EVE_KERNEL_PARAM_S |

【Reference header file】

aw_ai_eve_type.h

【Remarks】

classifierNum is the number of classifiers set by the user in the structure AW_AI_EVE_CTRL_SVariable.

## AW_SYSTEMTIME_S

【Description】

System time information Description

【Definition】

```
typedef struct _AW_SYSTEMTIME_S
{
    AW_U16 u16Year;                 //!< year
    AW_U16 u16Month;                //!< month
    AW_U16 u16DayOfWeek;            //!< week
    AW_U16 u16Day;                  //!< day
    AW_U16 u16Hour;                 //!< hour
    AW_U16 u16Minute;               //!< minute
    AW_U16 u16Second;               //!< second
    AW_U16 u16Milliseconds;         //!< milliseconds
}AW_SYSTEMTIME_S, *lpAW_SYSTEMTIME_S;
```

【Variable】

| Variable | Description | Type | Ranges |
|---|---|---|---|
| u16Year | Year | unsigned short | |
| u16Month | Month | unsigned short | |
| u16DayOfWeek | Week | unsigned short | |
| u16Day | Day | unsigned short | |
| u16Hour | Hour | unsigned short | |
| u16Minute | Minute | unsigned short | |
| u16Second | Second | unsigned short | |
| u16Milliseconds | Milliseconds | unsigned short | |

【Reference header file】

aw_ai_common_type.h

【Remarks】

# 4 ERROR code

【Description】

ERROR code returned by program crash

【Definition】

| Macro definition | Description | Value |
|---|---|---|
| AW_AI_ERROR_MEMALLOC_FAILURE | Memory application failed | 1001 |
| AW_AI_ERROR_INVALID_HANDLE | Illegal handle | 1002 |
| AW_AI_ERROR_INVALID_POINTER | Illegal pointer | 1003 |
| AW_AI_ERROR_UNSUPPORT_ABLITY | Unsupported behavioral analysis capability set | 2001 |

【Reference header file】

aw_ai_errorcode.h

【Description】

AW_AI_EVE_Event_GetLastErrorReturn value

# 5 Example

Refer to the sample_vi2EVE routine in the IPCLinuxPlatform\middleware\sample\sample_AILib folder.，The specific calling process is as follows：

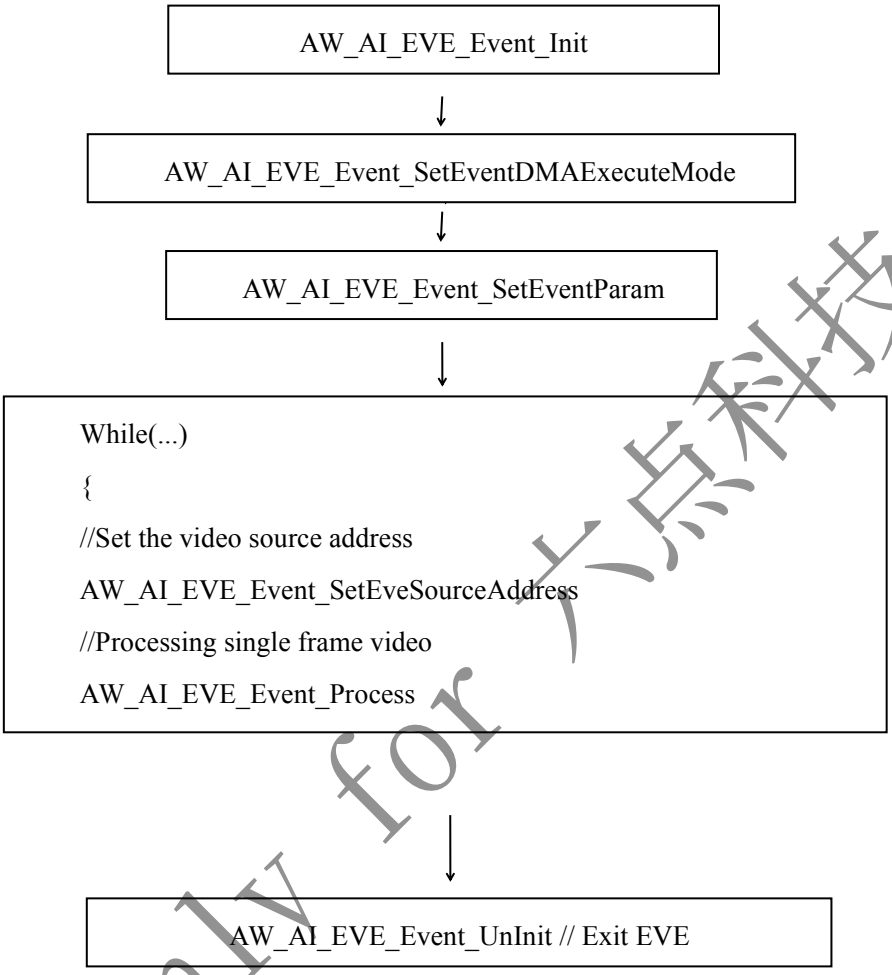



图 4-1 EVE event processing flow