MI SED API

Version 2.0.4

SigmaStar Technology makes no representations or warranties including, for example but not limited to, warranties of merchantability, fitness for a particular purpose, non-infringement of any intellectual property right or the accuracy or completeness of this document, and reserves the right to make changes without further notice to any products herein to improve reliability, function or design. No responsibility is assumed by SigmaStar Technology arising out of the application or use of any product or circuit described herein; neither does it convey any license under its patent rights, nor the rights of others.

REVISION HISTORY

Revision No.	Description	Date
2.04	• Initial release	01/20/2020

TABLE OF CONTENTS

REVISION HISTORY ... i
TABLE OF CONTENTS ... ii
1. API REFERENCE ... 1
1.1. Overview ... 1
1.1.1 image detection algorithms supported by the current intelligent coding module ... 2
1.1.2 the role of sed ... 2
1.1.3 data sources for SED ... 2
1.1.4 the flow chart of Sed detection is as follows ... 2
1.1.5 platform limitations supported by Sed's human-non-vehicle detection function ... 3
1.2. 1.2. Description of key terms ... 3
1.2.1 VDF (Video detection function) ... 3
1.2.2 IVE (Intelligent video coding) ... 3
1.2.3 IPU (AI Process Unit) ... 3
1.2.4 ROI(Region Of Interest) ... 3
1.2.5 Motion information ... 4
1.2.6 QP (Quantization Parameter) ... 4
1.2.7 MD ... 4
1.2.8 OD ... 4
1.2.9 VG ... 5
1.2.10 SAD ... 5
1.2.11 CBR (Constant Bit Rate) ... 5
1.2.12 VBR (Variable Bit Rate) ... 5
1.2.13 CVBR (Constrained VariableBit Rate) ... 5
1.2.14 ABR (Average Bit Rate) ... 6
1.2.15 IOU (Intersection Over Union) ... 6
2 API List ... 8
2.1 MI_SED_CreateChn ... 9
2.2 MI_SED_DestroyChn ... 9
2.3 MI_SED_StartDetector ... 10
2.4 MI_SED_StopDetector ... 11
2.5 MI_SED_AttachToChn ... 11
2.6 MI_SED_DetachFromChn ... 12
2.7 MI_SED_SetDbgLevel ... 13
2.8 MI_SED_GetRect ... 13
3 SED DATA TYPE ... 15
MI_SED_AlgoAttr_t ... 15
3.1 SED_MAX_CHN_NUM ... 16
3.2 SED_MAX_ROI_NUM_PER_CHN ... 16
3.3 SED_MAX_TARGET_CHN_NUM_PER_CHN ... 16
3.4 SED_MAX_CUS_DEF_ALGOPARAM_NUM ... 17
3.5 MI_SED_CHN ... 17
3.6 MI_SED_TARGET_CHN ... 17
3.7 MI_SED_AlgoType_e ... 18

3.8 MI_SED_MdMbMode_e 18
3.9 MI_SED_InputAttr_t....................19
3.10 MI_SED_Default_AlgoParam_t....................19
3.11 MI_SED_CusDef_AlgoParam_t....................20
3.12 MI_SED_AlgoAttr_t 21
3.13 MI_SED_TargetAttr_t....................21
3.14 MI_SED_DetectorAttr_t....................22
4 ERROR CODE.................... 23
5 How Build and Run Demo....................24
5.1 How to Build Sed module and demo code:....................24
5.2 How to Build Sed module and demo code: 24
5.3 Sed Demo code 示例 24

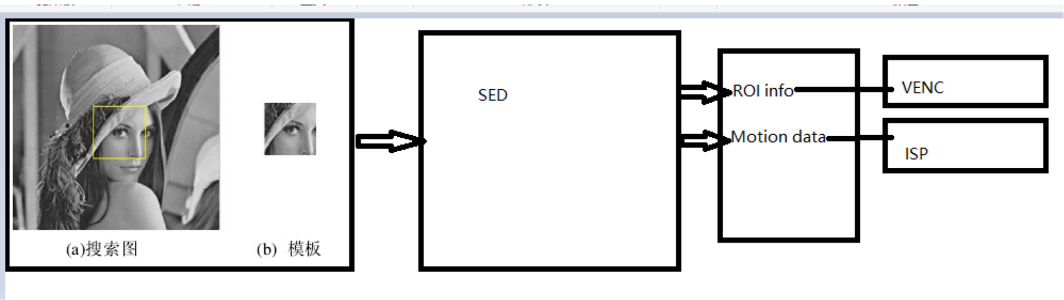
1. API REFERENCE

1.1. Overview

SED (smart coding)
The intelligent coding module mainly provides the functions of creating and destroying the intelligent coding channel, opening and stopping the detection source image, and calculating the result and associating it with the specified coding channel.

The core function of intelligent coding is first to do image recognition (recognizing images of objects in movement, or recognizing objects in the image), and then to set the coding parameters to the VENC module (e.g. when an object in motion is recognized, it can control the VENC module to reduce the QP value and improve the clarity (sharpness), and when the image recognized is in stationary state, it can control the VENC module to increase the QP value and reduce transmission bandwidth, etc.).

Generally, the detection algorithm flow will compare the difference between two consecutive frames to decide whether the screen has moved, extract the ROI and motion information of the screen, and set the QP of ROI specific area for the encoder to code. There are many kinds of detection algorithms. Those used in our platform will be introduced below.



Through SED's intelligent identification of ROI, the dynamic setting of QP value of ROI at the designated location of the image can effectively reduce bandwidth and ensure the quality of the image.

1.1.1 Image Detection Algorithms Supported by Current Intelligent Coding Module

1: E_MI_IVEOBJDETECT_ALGOPARAM: Use VDF/MD mode to detect moving objects. ROI for image tracking is supported.

2: E_MI_CNNOBJDETECT_ALGOPARAM: Objects that can be detected by this algorithm include bicycles, buses, cars, motorbikes, and human beings. ROI for image tracking is supported.

3: E_MI_MOTIONDETECT_ALGOPARAM: Motion detection based on IVE. Only supports AVBR motion detection.

The above three detection algorithms can be used simultaneously by creating different detection channel instances.

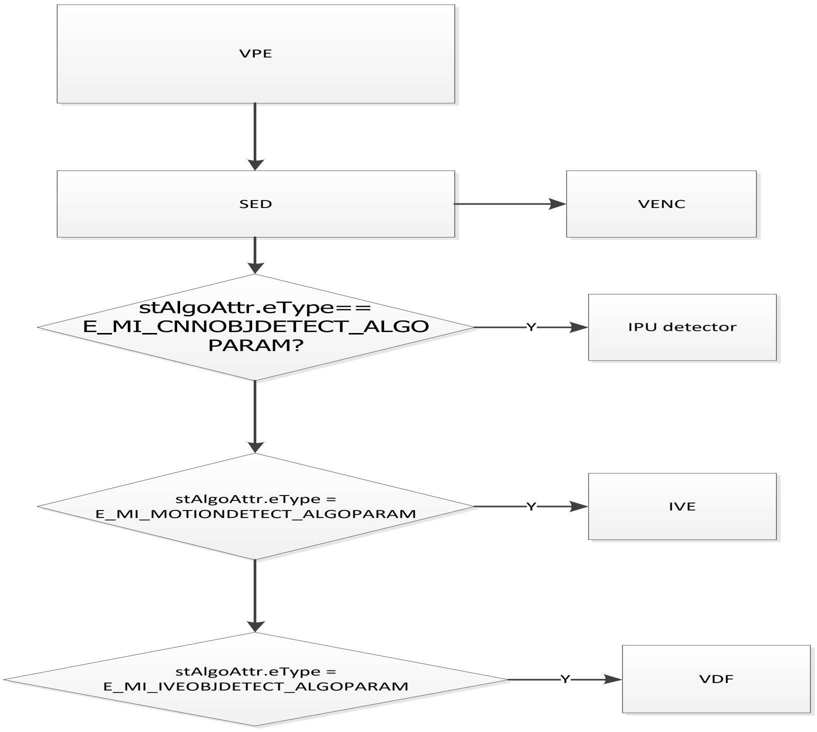
1.1.2 Role of SED

The SED is responsible for identifying ROI and motion information using Sigmastar or third-party intelligent algorithms. ROI information is set to VENC, and motion information is used to automatically adjust ISP sharpness, and so on.

1.1.3 Data Sources for SED

The data sources for SED is yuv data. The yuv data is obtained through VPE for intelligent detection (the resolution of yuv input is 352*288).

1.1.4 SED Detection Flowchart



1.1.5 Limitations of Platforms Supported by SED's Human-Non-Vehicle Detection Function

Human non-vehicle detection is based on the detection of AI training model through IPU. The accuracy of specific detection depends on the accuracy of AI model training. Currently, only the Pudding and Macaron platforms support human-non-vehicle detection, all other low-end platforms do not.

1.2. Keyword Description

1.2.1 VDF (Video Detection Function)

MI_VDF realizes initialization of MD, OD and VG video channels, channel management, management of video detection results, channel destruction, and other functions.

1.2.2 IVE (Intelligent Video Encoding)

The IVE module is responsible for processing and calculation of image data. The SAD value of two images can be obtained through MI_IVE_Sad, and the histogram statistical task data can be executed through MI_IVE_Hist to perform dynamic and static detection of the image.

1.2.3 IPU (AI Process Unit)

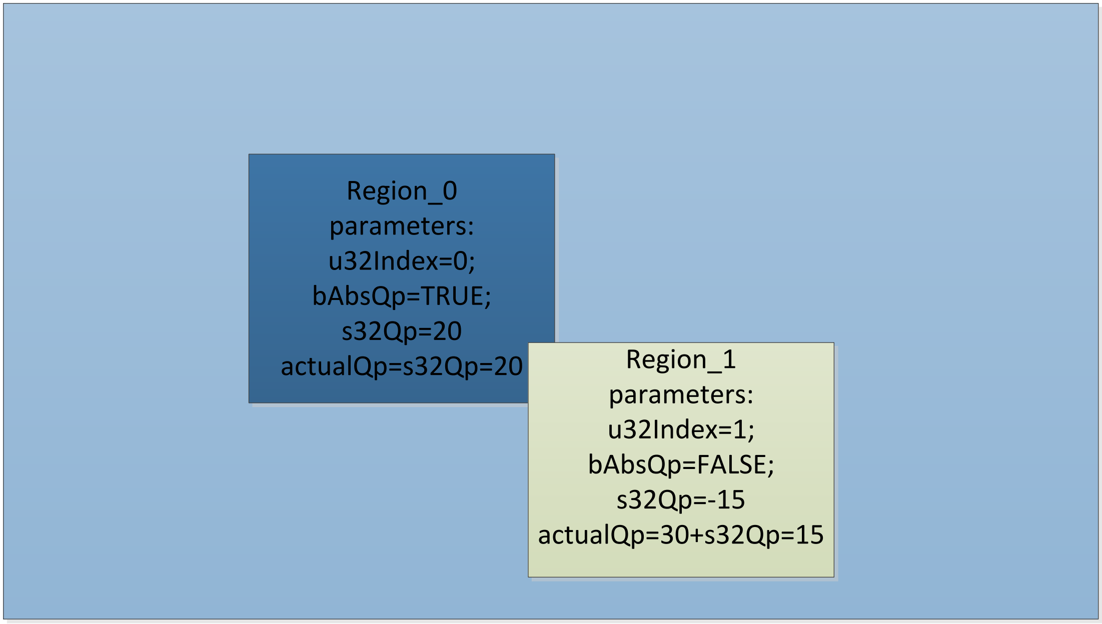
Acting as the AI model processing and computing unit, MI IPU module implements the rapid reasoning function of the network model, independently configuring the network model for each channel, and managing the acquisition and release of input and output data.

1.2.4 ROI (Region Of Interest)

With the interested-region ROI coding function enabled, important or mobile area will be in high quality lossless coding. For those which do not move, not the selected area to reduce the bit rate and image quality, the standard definition video compression, even not video transmission part of this area, so as to ultimately save network bandwidth and video storage space, users can configure their ROI area, to limit the area of image Qp, so as to realize the image of the region Qp differentiation with other image area. The system supports both H264 and H265 encoding ROI Settings and provides 16 ROI areas for simultaneous use by users.

16 ROI regions can be overlapped with each other, and the priority of the overlapped regions is raised in sequence according to the index number of 0-15, that is to say, the QP of the overlapped regions is finally determined to be processed only according to the highest priority regions. The ROI area can be configured with absolute QP and relative QP.

Absolute QP: QP in ROI area is the QP value set by the user Relative QP: the QP in ROI area is the sum of the QP generated by rate control and the QP offset value set by the user In the following example, the encoded image adopts the fixqp mode, setting the image QP to 30, that is, the QP value of all macroblocks of the image is 30. ROI region 0 is set to absolute QP mode, QP value is 20, index is 0; ROI region 1 is set to relative QP mode, QP is - 15, index is 1. Because the index of ROI region 0 is smaller than the index of ROI region 1, the QP of ROI region 1 with high priority is set in the overlapped image region. The QP value of region 1 is 30-15 = 15, except that the QP value of region 0 is 20.



1.2.5 Motion information

SAD information that contains the image.

1.2.6 QP (Quantization Parameter)

QP value corresponds to the sequence number of quantized step length. The smaller the value, the smaller the quantized step length, the higher the quantization accuracy, the better the picture quality, and the larger the size encoded.

1.2.7 MD

Motion detect is used to detect the movement of objects in films, and it is actually applied to security monitoring.

1.2.8 OD

The function of Occlusion detection is used to detect whether the received movie is blocked and output the blocked detection result.

1.2.9 VG

The virtual line segment is used to detect whether an object has crossed the set alarm line.
A zone intrusion is used to detect if an object is passing through a set alarm area.

1.2.10 SAD

The Sum of Absolute Differences. This algorithm is often used for image block matching, the sum of the absolute value of the difference between the corresponding values of each pixel is used to evaluate the similarity of two image blocks. It can be seen that this algorithm is fast, but not accurate, and is usually used for preliminary screening of multistage processing.

The value range of SAD is [0, 255]. All SAD values are divided into 16 ranges:

[0, 10), [10, 15, 15, 20), [20, 25), [25, 30), 30, 40), [40, 50), 50, 60), [; seven), 60, 70), [80 living), (90100), (100120), (120140), (140160), [160, 255].

Here is the percentage of all MB(8*8) in a frame that falls into the above range.

The larger the value, the more static the picture.

1.2.11 CBR (Constant Bit Rate)

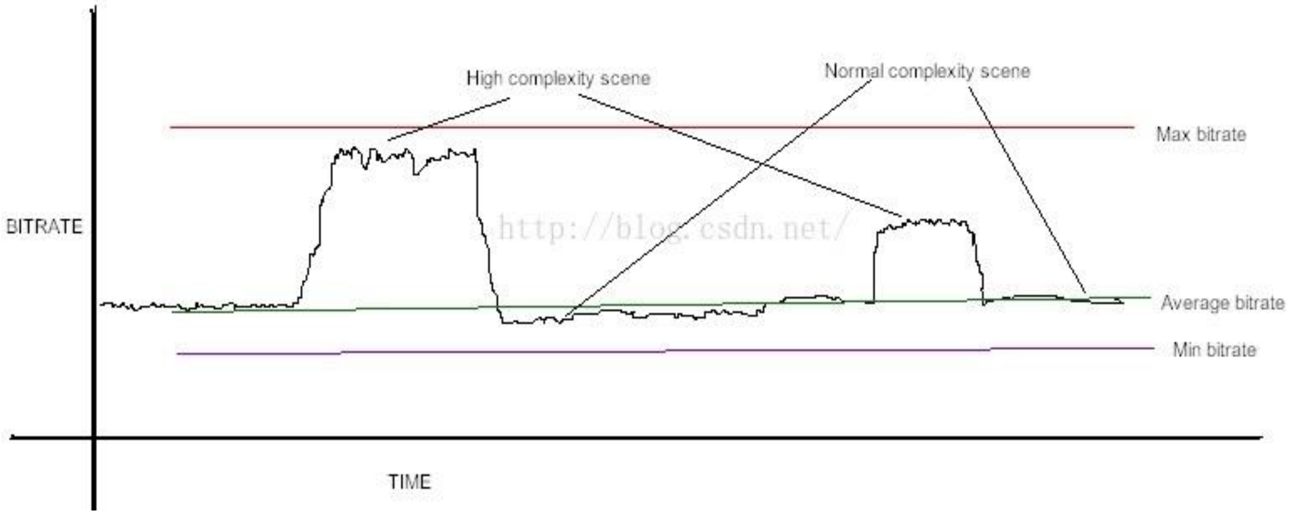
In case of Motion, since the code rate is constant, the size of the code word can only be reduced by increasing QP due to the constant code rate, so the image quality becomes worse. When the scene is still, the image quality becomes better, so the image quality is unstable. This algorithm prioritizes bitrate (bandwidth).

1.2.12 VBR (Variable Bit Rate)

The bit rate of dynamic bit rate can vary with the complexity of the image, so its coding efficiency is relatively high. When Motion occurs, Mosaic is rare. The bitrate control algorithm determines the bit rate to be used according to the image content. If the image content is simple, less bitrate will be allocated (it seems that the codeword is more suitable), while if the image content is complex, more codeword will be allocated, which not only guarantees the quality, but also takes into account the bandwidth limit. This algorithm gives priority to image quality.

1.2.13 CVBR (Constrained VariableBit Rate)

It is an improvement on VBR. But where is Constrained? The Maximum bitRate or Average bitRate corresponding to this algorithm is constant. This method gives consideration to the advantages of the above two methods: when the image content is still, the bandwidth is saved; when there is Motion, the bandwidth saved in the early stage is used to improve the image quality as much as possible, so as to achieve the purpose of giving consideration to both the bandwidth and the image quality. This method usually allows the user to input the maximum and minimum bit rate, which is stable at the minimum bit rate at rest, and greater than the minimum bit rate at motion, but not more than the maximum bit rate. The ideal model is as follows:

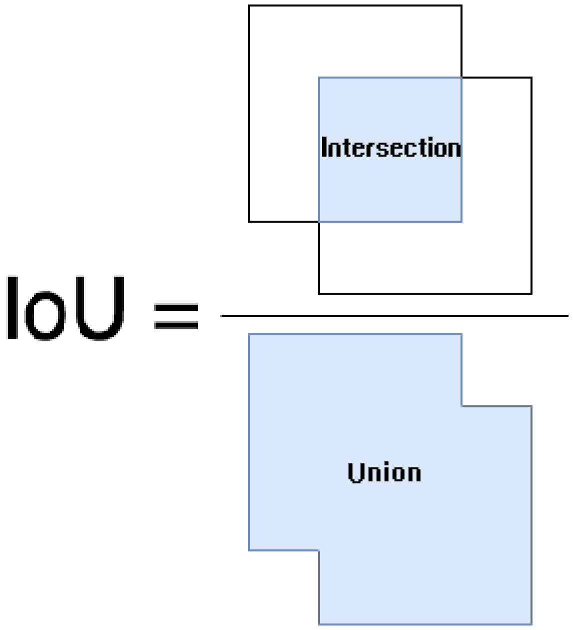


1.2.14 ABR (Average Bit Rate)

In a certain time range to reach the set code rate, but the local peak code rate can exceed the set code rate, the average code rate is constant.

1.2.15 IOU (Intersection Over Union)

Intersection over Union (IOU), a concept used in target detection, is the overlapping rate of the generated candidate box and the original marker box, that is, the ratio of their intersection and union. (for more details, please refer to: http://172.19.30.188:8090/pages/viewpage.action? Pageid = 1216493).



$$IOU = \frac{A \cap B}{A \cup B}$$

The best case is complete overlap, i.e. the ratio is 1.

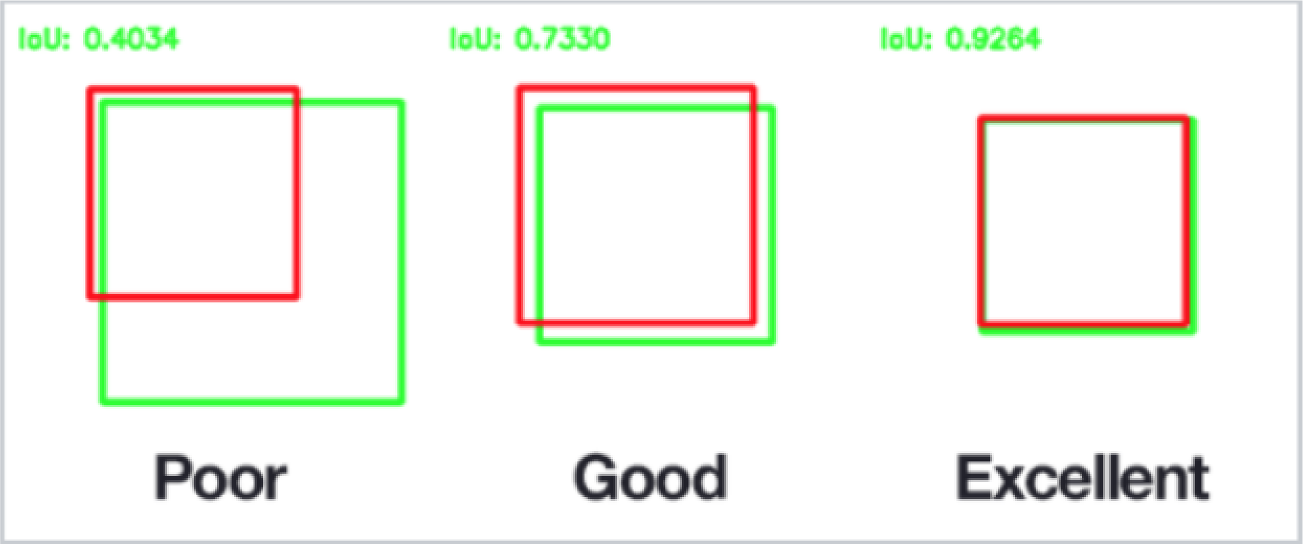


Diagram explaining IoU (from Wikipedia)

2. API LIST

This module provides the following APIs:

API Name	Function
MI_SED_CreateChn	Create an SED channel
MI_SED_DestroyChn	Destroy an SED channel
MI_SED_StartDetector	Start detection of designated channel
MI_SED_StopDetector	Stop detection of designated channel
MI_SED_AttachToChn	Attach module to designated channel
MI_SED_DetachFromChn	Detach module from designated channel

2.1. MI_SED_CreateChn

- Function

  Create an SED channel

- Syntax

  MI_S32 MI_SED_CreateChn(MI_SED_CHN SedChn, MI_SED_DetectorAttr_t* pstAttr);

- Parameter

Parameter Name	Description	Input/Output
SedChn	SED channel number. Range: [0, SED_MAX_CHN_NUM)	Input
pstAttr	SED detection attribute pointer	Input

- Return Value
  - Zero: Successful
  - Non-zero: Failed, see error code for details

- Dependency
  - Header: mi_common.h, mi_sed.h
  - Library: libmi_sed.a

- Note

  N/A.

- Example

  N/A.

- Related Function

  N/A.

2.2. MI_SED_DestroyChn

- Function

  Destroy an SED channel.

- Syntax

  MI_S32 MI_SED_DestroyChn(MI_SED_CHN SedChn);

- Parameter

Parameter Name	Description	Input/Output
SedChn	SED channel number. Range：[0, SED_MAX_CHN_NUM)	Input

- Return Value
  - Zero: Successful
  - Non-zero: Failed, see error code for details

- Dependency
  - Header: mi_common.h, mi_sed.h
  - Library: libmi_sed.a

- Note

  N/A.

- Example

  N/A.

- Related Function

  N/A.

2.3. MI_SED_StartDetector

- Function

  Start detection of a designated channel.

- Syntax

  MI_S32 MI_SED_StartDetector(MI_SED_CHN SedChn);

- Parameter

Parameter Name	Description	Input/Output
SedChn	SED channel number. Range：[0, SED_MAX_CHN_NUM)	Input

- Return Value
  - Zero: Successful
  - Non-zero: Failed, see error code for details

- Dependency
  - Header: mi_common.h, mi_sed.h
  - Library: libmi_sed.a

- Note

  N/A.

- Example

  N/A.

- Related Function

  N/A.

2.4. MI_SED_StopDetector

- Function

  Stop detection of a designated channel.

- Syntax

  MI_S32 MI_SED_StopDetector(MI_SED_CHN SedChn);

- Parameter

Parameter Name	Description	Input/Output
SedChn	SED channel number. Range：[0, SED_MAX_CHN_NUM)	Input

- Return Value
  - Zero: Successful
  - Non-zero: Failed, see error code for details

- Dependency
  - Header: mi_common.h, mi_sed.h
  - Library: libmi_sed.a

- Note

  N/A.

- Example

  N/A.

- Related Function

  N/A.

2.5. MI_SED_AttachToChn

- Description

  Attach module to designated channel.

- Syntax

  MI_S32 MI_SED_AttachToChn(MI_SED_CHN SedChn, MI_SED_TARGET_CHN TargetChn);

- Parameter

Parameter Name	Description	Input/Output
SedChn	SED channel number. Range：[0, SED_MAX_CHN_NUM)	Input
TargetChn	Target encoder channel Range：[0,VENC_MAX_CHN_NUM)	Input

- Return Value
  - Zero: Successful
  - Non-zero: Failed, see error code for details

- Dependency
  - Header: mi_common.h, mi_sed.h
  - Library: libmi_sed.a

- Note

  N/A.

- Example

  N/A.

- Related Function

  N/A.

2.6. MI_SED_DetachFromChn

- Description

  Detach module from designated channel.

- Syntax

  MI_S32 MI_SED_DetachFromChn(MI_SED_CHN SedChn, MI_SED_TARGET_CHN TargetChn);

- Parameter

Parameter Name	Description	Input/Output
SedChn	SED channel number. Range：[0, SED_MAX_CHN_NUM)	Input
TargetChn	Target encoder channel. Range：[0,VENC_MAX_CHN_NUM)	Input

- Return Value
  - Zero: Successful
  - Non-zero: Failed, see error code for details

- Dependency
  - DependencyHeader: mi_common.h, mi_sed.h
  - Library: libmi_sed.a

- Note

  N/A.

- Example

  N/A.

- Related Function

  N/A.

2.7. MI_SED_SetDbgLevel

- Description

  Set SED module debug level,control log output

- Syntax

  MI_S32 MI_SED_SetDbgLevel(MI_DBG_LEVEL_e eLevel)

- Parameter

Parameter Name	Description	Input/Output
eLevel	MI debug level enum. Range：[0, MI_DBG_ALL]	Input

- Return Value
  - Zero: Successful
  - Non-zero: Failed, see error code for details

- Dependency
  - DependencyHeader: mi_common.h, mi_sed.h
  - Library: libmi_sed.a

- Note

  N/A.

- Example

  N/A.

- Related Function

  N/A.

2.8. MI_SED_GetRect

- Description

  Get the rect info data detected by sed

- Syntax

  MI_S32 MI_SED_GetRect(MI_SED_CHN SedChn, MI_SED_RectInfo_t *pstRectInfo)

- Parameter

Parameter Name	Description	Input/Output
SedChn	SED channel number. Range：[0, SED_MAX_CHN_NUM)	input
pstRectInfo	Rect info	output

- Return Value
  - Zero: Successful
  - Non-zero: Failed, see error code for details

- Dependency
  - DependencyHeader: mi_common.h, mi_sed.h
  - Library: libmi_sed.a

- Note

N/A.

- Example

N/A.

- Related Function

N/A.

3. SED DATA TYPE

The relevant data types and data structures are defined as follows:

SED_MAX_CHN_NUM	Define maximum SED channel number
SED_MAX_ROI_NUM_PER_CHN	Define maximum ROI number per channel
SED_MAX_TARGET_CHN_NUM_PER_CHN	Define maximum target VENC channel number per SED channel
SED_MAX_CUS_DEF_ALGOPARAM_NUM	Define maximum number of supported customer-defined algorithm
MI_SED_CHN	Define SED channel number
MI_SED_TARGET_CHN	Define VENC encoder channel number corresponding to the SED channel number
MI_SED_AlgoType_e	Define SED algorithm type
MI_SED_MdMbMode_e	Define MD macro type
MI_SED_InputAttr_t	Define SED input attribute
MI_SED_Default_AlgoParam_t	Define SDK default algorithm parameter
MI_SED_CusDef_AlgoParam_t	Define customer-defined algorithm parameter
MI_SED_AlgoAttr_t	Define SED algorithm attribute
MI_SED_TargetAttr_t	Define final encoded target attribute
MI_SED_DetectorAttr_t	Define SED detector attribute

3.1. SED_MAX_CHN_NUM

- Description

  Define maximum SED channel number.

- Definition

  ```
  #define SED_MAX_CHN_NUM      (4)
  ```

- Note

  N/A.

- Related Data Type and Interface

  N/A.

3.2. SED_MAX_ROI_NUM_PER_CHN

- Description

  Define maximum ROI number per channel.

- Definition

  ```
  #define SED_MAX_ROI_NUM_PER_CHN      (16)
  ```

- Note

  N/A.

- Related Data Type and Interface

  N/A.

3.3. SED_MAX_TARGET_CHN_NUM_PER_CHN

- Description

  Define maximum target VENC channel number per SED channel.

- Definition

  ```
  #define SED_MAX_TARGET_CHN_NUM_PER_CHN      (8)
  ```

- Note

  N/A.

- Related Data Type and Interface

  N/A.

3.4. SED_MAX_CUS_DEF_ALGOPARAM_NUM

- Description

  Define maximum number of supported customer-defined algorithm.

- Definition

  ```
  #define SED_MAX_CUS_DEF_ALGOPARAM_NUM      (10)
  ```

- Note

  N/A.

- Related Data Type and Interface

  N/A.

3.5. MI_SED_CHN

- Description

  Define SED channel number.

- Definition

  ```
  typedef MI_S32 MI_SED_CHN;
  ```

- Note

  N/A.

- Related Data Type and Interface

  N/A.

3.6. MI_SED_TARGET_CHN

- Description

  Define VENC encoder channel number corresponding to the SED channel number.

- Definition

  ```
  typedef MI_S32 MI_SED_TARGET_CHN;
  ```

- Note

  N/A.

- Related Data Type and Interface

  N/A.

3.7. MI_SED_AlgoType_e

- Description

Define SED algorithm type.

- Definition

```
typedef enum
{
    E_MI_DEFAULT_ALGOPARAM = 0x0,
    E_MI_CUSDEF_ALGOPARAM = 0x1,
    E_MI_DEFAULT_ALGOPARAM_MAX
} MI_SED_AlgoType_e;
```

- Member

Member Name	Description
E_MI_DEFAULT_ALGOPARAM	Use default SDK algorithm
E_MI_CUSDEF_ALGOPARAM	Use customer-defined algorithm.

- Note

N/A.

- Related Data Type and Interface

N/A.

3.8. MI_SED_MdMbMode_e

- Description

Define MD macro type.

- Definition

```
typedef enum
{
    E_MI_MDMB_MODE_MB_4x4       = 0x0,
    E_MI_MDMB_MODE_MB_8x8       = 0x1,
    E_MI_MDMB_MODE_MB_16x16     = 0x2,
    E_MI_MDMB_MODE_MAX
} MI_SED_MdMbMode_e;
```

- Member

Member Name	Description
E_MI_MDMB_MODE_MB_4x4	Use 4×4 macroblock
E_MI_MDMB_MODE_MB_8x8	Use 8×8 macroblock
E_MI_MDMB_MODE_MB_16x16	Use 16×16 macroblock

- Note

  N/A.

- Related Data Type and Interface

  N/A.

3.9. MI_SED_InputAttr_t

- Description

  Define SED input attribute.

- Definition

```
typedef struct MI_SED_InputAttr_s
{
    MI_U32 u32Width;
    MI_U32 u32Height;
    MI_U32 u32FrameRateNum;
    MI_U32 u32FrameRateDen;
    MI_SYS_ChnPort_t stInputPort;
} MI_SED_InputAttr_t;
```

- Member

Member Name	Description
u32Width	Input YUV width
u32Height	Input YUV height
u32FrameRateNum	Input YUV framerate numerator
u32FrameRateDen	Input YUV framerate denominator
stInputPort	Input YUV channel attribute

- Note

  N/A.

- Related Data Type and Interface

  MI_SED_CreateChn

  MI_SED_DetectorAttr_t

3.10. MI_SED_Default_AlgoParam_t

- Description

  Define default SDK algorithm parameter.

- Definition

```
typedef struct MI_SED_Default_AlgoParam_s
{
     MI_U32              u32VdfChn;
     MI_U8               u8Sensitivity;
     MI_SED_MdMbMode_e eMdMbMode;
} MI_SED_Default_AlgoParam_t;
```

- Member

Member Name	Description
u32VdfChn	SED internal VDF channel number
u8Sensitivity	SED algorithm sensitivity
eMdMbMode	MD macroblock type

- Note

N/A.

- Related Data Type and Interface

MI_SED_CreateChn
MI_SED_AlgoAttr_t

3.11. MI_SED_CusDef_AlgoParam_t

- Description

Define customer-defined algorithm parameter.

- Definition

```
typedef struct MI_SED_CusDef_AlgoParam_s
{
    MI_U32 u32ParamNum;
    MI_U32 u32CusDefAlgoParam[SED_MAX_CUS_DEF_ALGOPARAM_NUM];
} MI_SED_CusDef_AlgoParam_t;
```

- Member

Member Name	Description
u32ParamNum	Effective parameter number
u32CusDefAlgoParam	Parameter array

- Note

N/A.

- Related Data Type and Interface

  MI_SED_CreateChn

  MI_SED_AlgoAttr_t

3.12. MI_SED_AlgoAttr_t

- Description

  Define SED algorithm attribute.

- Definition

```
typedef struct MI_SED_CusAlgoAttr_s
{
    MI_SED_AlgoType_e eType;
     union
    {
          MI_SED_Default_AlgoParam_t     stDefaultAlgoParam;
          MI_SED_CusDef_AlgoParam_t      stCusDefAlgoParam;

    };
} MI_SED_AlgoAttr_t;
```

- Member

Member Name	Description
eType	Type of algorithm used by SED
stDefaultAlgoParam	Default algorithm parameter
stCusDefAlgoParam	Customer-defined algorithm parameter

- Note

  Customer-defined algorithm is currently not supported.

- Related Data Type and Interface

  MI_SED_CreateChn

  MI_SED_DetectorAttr_t

3.13. MI_SED_TargetAttr_t

- Description

  Define final encoded target attribute.

- Definition

```
typedef struct MI_SED_TargetAttr_s
{
    MI_S32 s32RltQp;
} MI_SED_TargetAttr_t;
```

- Member

Member Name	Description
s32RltQp	ROI corresponding Qp value

- Note

N/A.

- Related Data Type and Interface

MI_SED_CreateChn
MI_SED_DetectorAttr_t

3.14. MI_SED_DetectorAttr_t

- Description

Define SED detector attribute.

- Definition

```
typedef struct MI_SED_DetectorAttr_s
{
    MI_SED_InputAttr_t stInputAttr;
    MI_SED_AlgoAttr_t stAlgoAttr;
    MI_SED_TargetAttr_t stTargetAttr;
} MI_SED_DetectorAttr_t;
```

- Member

Member Name	Description
stInputAttr	SED input attribute
stAlgoAttr	SED algorithm related attribute
stTargetAttr	SED final target control attribute

- Note

N/A.

- Related Data Type and Interface

MI_SED_CreateChn

4. ERROR CODE

The SED API error codes are shown in the table below:

Table 1: SED API Error Code

Error Code	Macro Definition	Description
0x00000000	MI_SUCCESS	Success
0xA01E2002	MI_ERR_SED_INVALID_CHNID	Invalid channel ID
0xA01E2003	MI_ERR_SED_ILLEGAL_PARAM	At lease one parameter is illegal
0xA01E2004	MI_ERR_SED_EXIST	Channel already exists
0xA01E2005	MI_ERR_SED_UNEXIST	Channel does not exist
0xA01E2006	MI_ERR_SED_NULL_PTR	Using a NULL point
0xA01E200c	MI_ERR_SED_NOMEM	Failure caused by malloc memory
0xA01E2013	MI_ERR_SED_CHN_NOT_STARTED	Channel not started
0xA01E2014	MI_ERR_SED_CHN_NOT_STOPPED	Channel not stopped
0xA01E2017	MI_ERR_SED_NOT_ENABLE	Channel not enabled

5. HOW BUILD AND RUN DEMO

5.1. How to Build Sed module and demo code

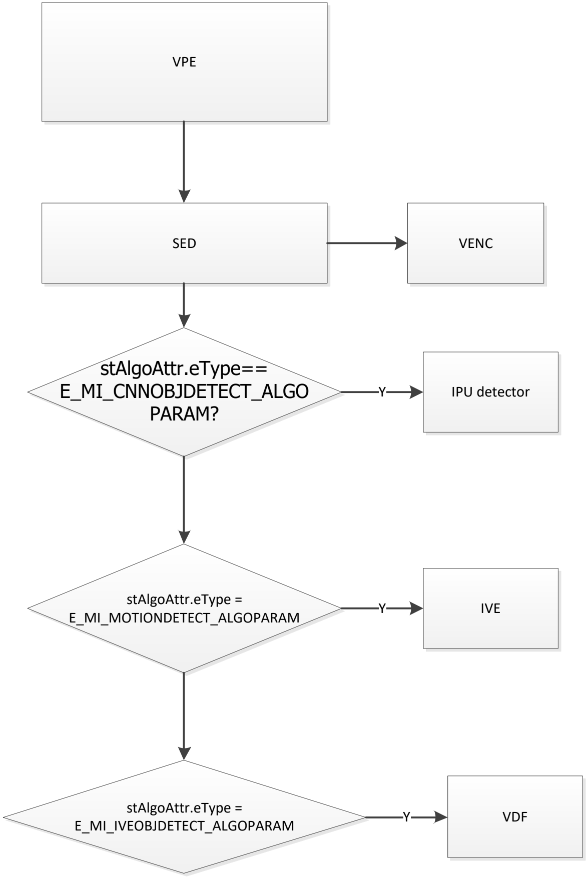
Build Demo code:
cd sdk/verify/mi_demo/alderaan$
make

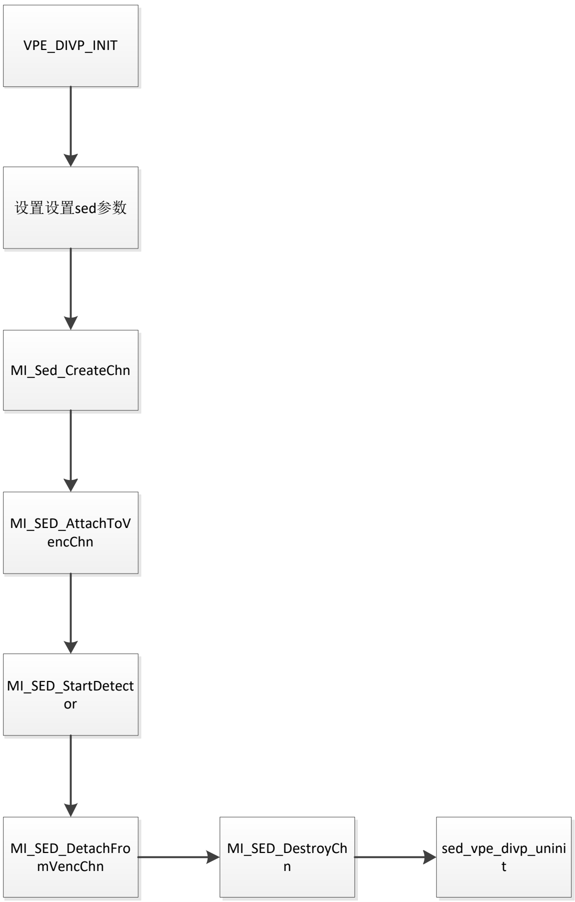
5.2. How to Build Sed module and demo code

Build Demo code:
cd sdk/verify/mi_demo/alderaan$
make

5.3. 5.3 Sed Demo code

http://hcgit04-master:9080/#/c/mstar/alkaid/sdk/+/68242/


VPE
SED
VENC
stAlgoAttr.eType==
E_MI_CNNOBJDETECT_ALGO
PARAM?
Y
IPU detector
stAlgoAttr.eType =
E_MI_MOTIONDETECT_ALGOPARAM
Y
IVE
stAlgoAttr.eType =
E_MI_IVEOBJDETECT_ALGOPARAM
Y
VDF


VPE_DIVP_INIT
设置设置sed参数
MI_Sed_CreateChn
MI_SED_AttachToVencChn
MI_SED_StartDetector
MI_SED_DetachFromVencChn
MI_SED_DestroyChn
sed_vpe_divp_uninit