MI VDF API

Version 2.04

REVISION HISTORY

Revision N/A.	Description	Date
2.03	• Initial release	11/12/2018
2.04	• Refine docs	4/12/2019

TABLE OF CONTENTS

1. API REFERENCE

1.1. Description

MI_VDF implements MD, OD, VG video channel initialization, channel management, video detection result management and channel destruction

1.2. VDF API List

API Name	Function
MI_VDF_Init	Initialize MI_VDF module
MI_VDF_Uninit	Un-initialize MI_VDF Module
MI_VDF_CreateChn	Create channel for MD/OD/VG
MI_VDF_DestroyChn	Destroy created channel MD/OD/VG
MI_VDF_SetChnAttr	Set channel（MD/OD/VG）attribute
MI_VDF_GetChnAttr	Get attribute from channel（MD/OD/VG）
MI_VDF_EnableSubWindow	Enable sub windows for （MD/OD/VG）channel
MI_VDF_Run	Start detection for channel（MD/OD/VG）
MI_VDF_Stop	Stop detection for channel（MD/OD/VG）
MI_VDF_GetResult	Get detection result from channel（MD/OD/VG）
MI_VDF_PutResult	Release detection result from channel（MD/OD/VG）
MI_VDF_GetLibVersion	Get version of VDF
MI_VDF_GetDebugInfo	Get debug info “Debuginfo”(only VG support this feature)

1.3. VDF API Description

1.3.1 MI_VDF_Init

- Functions

  Create VDF channel，initialize the VDF module.

- Syntax

  MI_S32 MI_VDF_Init(void);

- Parameters

  N/A.

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

- Dependency
  - Header File：mi_sys.h, mi_md.h, mi_od.h, mi_vg.h, mi_vdf.h
  - Library File：libOD_LINUX.a, libMD_LINUX.a, libVG_LINUX.a, libmi_vdf.a

- Note
  - MI_VDF_Init needs to be called after calling MI_SYS_Init.
  - It is not possible to call MI_VDF_Init repeatedly.

- Example

  No

- Related APIs

  MI_VDF_Uninit

1.3.2 MI_VDF_Uninit

- Functions

  Destructing the MI_VDF system, before calling MI_VDF_Uninit, you need to ensure that the created VDF channels have been disabled and the video detection results of all channels are released

- Syntax

  MI_S32 MI_VDF_Uninit (void);

- Return Value

  No

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

- Dependency
  - Header File : mi_sys.h, mi_md.h, mi_od.h, mi_vg.h, mi_vdf.h
  - Library File : libOD_LINUX.a, libMD_LINUX.a, libVG_LINUX.a, libmi_vdf.a

- Note
  - Before the MI_VDF_Uninit call, you need to ensure that all created VDF channels are in the disable state.
  - Before the MI_VDF_Uninit call, you need to ensure that the results of all created VDF channels are released.

- Example

  See MI_VDF_Init For example.

- Related APIs

  No

1.3.3 MI_VDF_CreateChn

- Functions

  Create a video detection (MD/OD/VG) channel.

- Syntax

  MI_S32 MI_VDF_CreateChn(MI_VDF_CHANNEL VdfChn, const MI_VDF_ChnAttr_t* pstAttr);

- Return Value

Parameter Name	Description	Input/Output
VdfChn	Specify the video detection channel number.	Input
pstAttr	Set the video channel property value.	Input

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

- Dependency
  - Header File : mi_sys.h, mi_md.h, mi_od.h, mi_vg.h, mi_vdf.h
  - Library File : libOD_LINUX.a, libMD_LINUX.a, libVG_LINUX.a, libmi_vdf.a

- Note
  - VdfChn cannot be specified repeatedly.
  - VdfChn valid range: 0<= VdfCh <MI_VDF_CHANNEL_MAX. MI_VDF_CHANNEL_MAX defined in header file: mi_vdf_datatype.h.

- Example

  N/A.

- Related APIs

  MI_VDF_DestroyChn

1.3.4 MI_VDF_DestroyChn

- Functions

  Destroy the created video detection channel.

- Syntax

  MI_S32 MI_VDF_DestroyChn(MI_VDF_CHANNEL VdfChn);

- Return Value

Parameter Name	Description	Input/Output
VdfChn	The video detection channel number that has been created.	Input

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

- Dependency
  - Header File : mi_sys.h, mi_md.h, mi_od.h, mi_vg.h, mi_vdf.h
  - Library File : libOD_LINUX.a, libMD_LINUX.a, libVG_LINUX.a, libmi_vdf.a

- Note
  - VdfChn cannot be specified repeatedly.
  - VdfChn valid range: 0<= VdfCh <MI_VDF_CHANNEL_MAX. MI_VDF_CHANNEL_MAX defined in header file: mi_vdf_datatype.h.

- Example

  N/A.

- Related APIs

  N/A.

1.3.5 MI_VDF_SetChnAttr

- Functions

  Set the video detection channel attributes.

- Syntax

  MI_S32 MI_VDF_SetChnAttr(MI_VDF_CHANNEL VdfChn, const MI_VDF_ChnAttr_t* pstAttr);

- Return Value

Parameter Name	Description	Input/Output
VdfChn	The video detection channel number that has been created.	Input
pstAttr	Source port configuration information data structure pointer.	Input

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

- Dependency
  - Header File : mi_sys.h, mi_md.h, mi_od.h, mi_vg.h, mi_vdf.h
  - Library File : libOD_LINUX.a, libMD_LINUX.a, libVG_LINUX.a, libmi_vdf.a

- Note
  - VdfChn must be a video detection channel that has been created.
  - This interface specifies the value of the dynamic attribute of the video detection channel. (stMdDynamicParamsIn /stOdDynamicParamsIn/ stVgAttr in MI_VDF_ChnAttr_t.).

- Example

  No

- Related APIs

  No

1.3.6 MI_VDF_GetChnAttr

- Functions

  Get the video detection channel attributes.

- Syntax

  MI_S32 MI_VDF_GetChnAttr(MI_VDF_CHANNEL VdfChn, MI_VDF_ChnAttr_t* pstAttr);

- Return Value

Parameter Name	Description	Input/Output
VdfChn	The video detection channel number that has been created.	Input
pstAttr	Used to save the returned video detection channel attribute value	Output

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

- Dependency
  - Header File : mi_sys.h, mi_md.h, mi_od.h, mi_vg.h, mi_vdf.h
  - Library File : libOD_LINUX.a, libMD_LINUX.a, libVG_LINUX.a, libmi_vdf.a

- Note

  No

- Example

  No

- Related APIs

  No

1.3.7 MI_VDF_EnableSubWindow

- Functions

  Enable/disable the specified video detection channel sub-window.

- Syntax

  MI_S32 MI_VDF_EnableSubWindow(MI_VDF_CHANNEL VdfChn, MI_U8 u8Col, MI_U8 u8Row, MI_U8 u8Enable);

- Return Value

Parameter Name	Description	Input/Output
VdfChn	The video detection channel number that has been	Input

	created.	
u8Col	The row address of the sub window **(resverd, no use)**	Input
u8Row	Sub-window column address **(resverd, no use)**	Input
u8Enable	Sub-window enable control flag	Input

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

- Dependency
  - Header File：Mi_sys.h, mi_md.h, mi_od.h, mi_vg.h, mi_vdf.h
  - Library File：libOD_LINUX.a, libMD_LINUX.a, libVG_LINUX.a, libmi_vdf.a

- Note
  - u8Col, u8Row are reserved for upward compatibility of API interface, which has no practical effect and you can using 0 directly.
  - The MI_VDF_Run/MI_VDF_Stop function act on all video detection channels in the same working mode (MD/OD/VG), and MI_VDF_EnableSubWindow act on a single video detection channel. For running a single VdfChn, please call MI_VDF_Run function and call MI_VDF_EnableSubWindow (u8Enable = True). But when you call MI_VDF_Stop function or MI_VDF_EnableSubWindow(u8Enable = False), it will stop.

- Example

  N/A.

- Related APIs

  N/A.

1.3.8 MI_VDF_Run

- Functions

  Start running video detection working mode.

- Syntax

  MI_S32 MI_VDF_Run(MI_VDF_WorkMode_e enWorkMode);

- Return Value

Parameter Name	Description	Input/Output
enWorkMode	Video detection (MD/OD/VG) working mode.	Input

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

- Dependency
  - Header File : mi_sys.h, mi_md.h, mi_od.h, mi_vg.h, mi_vdf.h
  - Library File : libOD_LINUX.a, libMD_LINUX.a, libVG_LINUX.a, libmi_vdf.a

- Note
  - Work mode only supports: MD/OD/VG.
  - MI_VDF_Run/MI_VDF_Stop function act on all video detection channels in the same working mode (MD/OD/VG).

- Example

  N/A.

- Related APIs

  N/A.

1.3.9 MI_VDF_Stop

- Functions

  Stop running video detection (MD/OD/VG) working mode.

- Syntax

  MI_S32 MI_VDF_Stop(MI_VDF_WorkMode_e enWorkMode);

- Return Value

Parameter Name	Description	Input/Output
enWorkMode	Video detection (MD/OD/VG) working mode.	Input

- Return Value
  - Zero: Successful

- Non-zero: Failed, see error code for details

➢ Dependency

- Header File : mi_sys.h, mi_md.h, mi_od.h, mi_vg.h, mi_vdf.h
- Library File : libOD_LINUX.a, libMD_LINUX.a, libVG_LINUX.a, libmi_vdf.a

➢ Note

- Work mode only supports: MD/OD/VG.
- MI_VDF_Run/MI_VDF_Stop function act on all video detection channels in the same working mode (MD/OD/VG).

➢ Example

N/A.

➢ Related APIs

N/A.

1.3.10 MI_VDF_GetResult

➢ Functions

Get the detection result of the specified video detection channel.

➢ Syntax

MI_S32 MI_VDF_GetResult(MI_VDF_CHANNEL VdfChn, MI_VDF_Result_t* pstVdfResult, MI_S32 s32MilliSec);

➢ Return Value

Parameter Name	Description	Input/Output
VdfChn	The video detection channel number that has been created.	Input
s32MilliSec	Input time **(resverd, no use)**	Input
pstVdfResult	save the returned video detection results	Output

➢ Return Value

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

➢ Dependency

- Header File : mi_sys.h, mi_md.h, mi_od.h, mi_vg.h, mi_vdf.h
- Library File : libOD_LINUX.a, libMD_LINUX.a, libVG_LINUX.a, libmi_vdf.a

➢ Note

- VdfChn must be a video detection channel that has been created.
- s32MilliSec is reserved for upward compatibility of API interface, which has no practical

effect and you can using 0 directly.

- The pointer used to save the returned result cannot be Null.

➢ Example

No

➢ Related APIs

N/A.

1.3.11 MI_VDF_PutResult

➢ Functions

Release the detection result of the specified video detection channel.

➢ Syntax

MI_S32 MI_VDF_PutResult(MI_VDF_CHANNEL VdfChn, MI_VDF_Result_t* pstVdfResult);

➢ Return Value

Parameter Name	Description	Input/Output
VdfChn	The video detection channel number that has been created.	Input
pstVdfResult	Specify the result parameters that need to release the video channel	Input

➢ Return Value

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

➢ Dependency

- Header File : mi_sys.h, mi_md.h, mi_od.h, mi_vg.h, mi_vdf.h
- Library File : libOD_LINUX.a, libMD_LINUX.a, libVG_LINUX.a, libmi_vdf.a

➢ Note

VdfChn must be a video detection channel that has been created.

➢ Example

N/A.

➢ Related APIs

N/A.

1.3.12 MI_VDF_GetLibVersion

➢ Functions

Obtain the MD/OD/VG library version number.

- Syntax

  MI_S32 MI_VDF_GetLibVersion(MI_VDF_CHANNEL VdfChn, MI_U32* u32VDFVersion);

- Return Value

Parameter Name	Description	Input/Output
VdfChn	The video detection channel number that has been created.	Input
u32VDFVersion	Pointer for saving the version number (cannot be Null).	Output

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

- Dependency
  - Header File : mi_sys.h, mi_md.h, mi_od.h, mi_vg.h, mi_vdf.h
  - Library File : libOD_LINUX.a, libMD_LINUX.a, libVG_LINUX.a, libmi_vdf.a

- Note

  N/A.

- Example

  N/A.

- Related APIs

  N/A.

1.3.13 MI_VDF_GetDebugInfo

- Functions

  Obtain the MD/OD/VG debug info.

- Syntax

  MI_S32 MI_VDF_GetDebugInfo(MI_VDF_CHANNEL VdfChn, MI_VDF_DebugInfo_t *pstDebugInfo);

- Return Value

Parameter Name	Description	Input/Output
VdfChn	The video detection channel number that has been created.	Input
u32VDFVersion	Pointer for saving the VDF debug info (cannot be Null)	Output

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

- Dependency
  - Header File : mi_sys.h, mi_md.h, mi_od.h, mi_vg.h, mi_vdf.h
  - Library File : libOD_LINUX.a, libMD_LINUX.a, libVG_LINUX.a, libmi_vdf.a

- Note

  Only supports VG mode.

- Example

  N/A.

- Related APIs

  N/A.

2. DATA TYPES

2.1. VDF Structure List

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

MI_VDF_WorkMode_e	An enumerated type that defines the working mode of VDF
MI_VDF_Color_e	Define the enumerated type of the video detection channel input source
MI_VDF_ODWindow_e	Enumerated type of the number of child windows of the screen when defining OD
MI_MD_Result_t	Structure that defines the MD result
MI_OD_Result_t	Structure that defines the OD result
MI_VG_Result_t	Structure that defines the VG result
MI_VDF_Result_t	A structure that defines the corresponding result of the VDF working mode
MI_VDF_MdAttr_t	Structure that defines MD channel attributes
MI_VDF_OdAttr_t	Structure that defines OD channel attributes
MI_VDF_VgAttr_t	Structure that defines VG channel attributes
MI_VDF_ChnAttr_t	Structure that defines the channel's working mode properties
MDRST_STATUS_t	A structure defining a motion detection result of a detection sub-window region

2.1.1 MI_VDF_WorkMode_e

- Description

  Define VDF work mode enumeration type.

- Definition

```
typedef enum
{
    E_MI_VDF_WORK_MODE_MD = 0,
    E_MI_VDF_WORK_MODE_OD,
    E_MI_VDF_WORK_MODE_VG,
    E_MI_VDF_WORK_MODE_MAX
}MI_VDF_WorkMode_e;
```

- Members

Member Name	Description
E_MI_VDF_WORK_MODE_MD	MD working mode
E_MI_VDF_WORK_MODE_OD	OD working mode
E_MI_VDF_WORK_MODE_VG	VG working mode
E_MI_VDF_WORK_MODE_MAX	Error code identification of working mode

- Note

  N/A.

- Related API and Types

  N/A.

2.1.2 MI_VDF_Color_e

- Description

  Define the enumerated type of the video detection channel input source

- Definition

```
typedef enum
{
    E_MI_VDF_COLOR_Y = 1,
    E_MI_VDF_COLOR_MAX
} MI_VDF_Color_e;
```

- Members

Member Name	Description
E_MI_VDF_COLOR_Y	Correct identification of the input source type of the video detection channel
E_MI_VDF_COLOR_MAX	Video detection channel input source type error code identification

- Note

  N/A.

- Related API and Types

  N/A.

2.1.3 MI_VDF_ODWindow_e

- Description

  Enumerated type of the number of child windows of the screen when defining OD

- Definition

```
typedef enum
{
    E_MI_VDF_ODWINDOW_1X1 = 0,
    E_MI_VDF_ODWINDOW_2X2,
    E_MI_VDF_ODWINDOW_3X3,
    E_MI_VDF_ODWINDOW_MAX
} MI_VDF_ODWindow_e;
```

- Members

Member Name	Description
E_MI_VDF_ODWINDOW_1X1	The OD screen is divided into 1 sub-window
E_MI_VDF_ODWINDOW_2X2	The OD screen is divided into 2x2 sub-windows
E_MI_VDF_ODWINDOW_3X3	The OD screen is divided into 3x3 sub-windows
E_MI_VDF_ODWINDOW_MAX	Error code identification of the number of OD screen sub-windows

- Note

  N/A.

- Related API and Data Types

  N/A.

2.1.4 MI_MD_Result_t

- Description

  A structure that defines the MD result.

- Definition

```
typedef struct MI_MD_Result_s
{
    MI_U64 u64Pts;                  //The PTS of Image
    MI_U8   u8Enable;                // =1 indicates that the result value is valid
    MI_MD_ResultSize_t stSubResultSize;
    MDRST_STATUS_t* pstMdResultStatus; //The MD result of Status
    MDSAD_DATA_t* pstMdResultSad; //The MD result of SAD
    MDOBJ_DATA_t* pstMdResultObj; //The MD result of Obj
}MI_MD_Result_t;
```

- Members

Member Name	Description
u64Pts	Image display timestamp
u8Enable	Indicates whether the channel is enabled
u8Reading	Indicates that the result is being read by the application layer
stSubResultSize	Describe the Size of the Sad , Obj , andReasultStauts substructures returned by the MD
pstMdResultStatus	Describe whether motion is detected in each area of the MD
pstMdResultSad	Describe the Sad value of the MD
pstMdResultObj	Describe the CCL value of the MD

- Note

N/A.

- Related API and Data Types

N/A.

2.1.5 MI_OD_Result_t

- Description

The structure that defines the OD result.

- Definition

```
typedef struct MI_OD_Result_s
{
    MI_U8   u8Enable;
    MI_U8    u8WideDiv;              //The number of divisions of window in horizontal direction
    MI_U8   u8HightDiv;             //The number of divisions of window in vertical direction
    MI_U8   u8DataLen;               //OD detect result readable size
    MI_U64 u64Pts;                  //The PTS of Image
    MI_S8   u8RgnAlarm[3][3];       //The OD result of the sub-window
}MI_OD_Result_t;
```

- Members

Member name	description
u8Enable	Indicates whether the channel is enabled
u8WideDiv	Get the number of windows in the horizontal direction of the OD
u8HightD iv	Get the number of windows in the vertical direction of the OD
u8DataLen	Set the readable size of the OD test results
u64Pts	Image display time
u8RgnAlarm[3][3]	OD sub window information result

- Note

  N/A.

- Related API and Data Types

  N/A.

2.1.6 MI_VG_Result_t

- Description

  The structure that defines the VG result.

- Definition

  typedef MI_VgResult_t MI_VG_Result_t;

- Members

Member Name	Description
alarm[4]	Describe the test results of VG

- Note

  N/A.

- Related API and Data Types

  N/A.

2.1.7 MI_VDF_Result_t

- Description

  A structure that defines the result of the VDF working mode.

- Definition

```
typedef struct MI_VDF_Result_s
{
    MI_VDF_WorkMode_e enWorkMode;
    VDF_RESULT_HANDLE handle;
    union
    {
        MI_MD_Result_t stMdResult;
        MI_OD_Result_t stOdResult;
        MI_VG_Result_t stVgResult;
    };
}MI_VDF_Result_t;
```

- Members

Member name	description
enWorkMode	VDF working mode (MD/OD/VG)
Handle	Save result handle
stMdResult	Structure describing the MD result
stOdResult	Structure describing the OD result
stVgResult	Structure describing the VG result

- Note

N/A.

- Related API and Data Types

N/A.

2.1.8 MI_VDF_MdAttr_t

- Description

Structure that defines MD channel properties

- Definition

```
typedef struct MI_VDF_MdAttr_s
{
    MI_U8   u8Enable;
    MI_U8   u8MdBufCnt;
    MI_U8   u8VDFIntvl;
    MI_U16 u16RstBufSize;
    MI_MD_ResultSize_t stSubResultSize;
    MDCCL_ctrl_t ccl_ctrl;
    MI_MD_static_param_t   stMdStaticParamsIn;
    MI_MD_param_t stMdDynamicParamsIn;
}MI_VDF_MdAttr_t;
```

- Members

Member name	description
u8Enable	Indicates whether the channel is enabled
u8MdBufCnt	Set the number of MD results that can be cached MD result cache number range: [1, 8] static attribute
u8VDFIntvl	Detection interval value range: [0, 29] , in frames, dynamic attribute
u16RstBufSize	The total size of the MD return results
stSubResultSize	MD returns the result of each substructure (Sad value, CCL value, ResultStatus) size
Ccl_ctrl	c cl _ctrl property setting
stMdStaticParamsIn	MD static attribute parameter setting
stMdDynamicParamsIn	MD dynamic attribute parameter setting

- Note

  N/A.

- Related API and Data Types

  NO.

2.1.9 MI_VDF_OdAttr_t

- Description

  Structure that defines OD channel attributes

- Definition

```
typedef struct MI_VDF_OdAttr_s
{
    MI_U8 u8Enable;
    MI_U8 u8OdBufCnt;
    MI_U8 u8VDFIntvl;
    MI_U16 u16RstBufSize;
    MI_OD_static_param_t stOdStaticParamsIn;
    MI_OD_param_t stOdDynamicParamsIn;
}MI_VDF_OdAttr_t;
```

- Members

Member name	description
u8Enable	Indicates whether the channel is enabled
u8OdBufCnt	OD result cached number range: [1, 16] static attribute
u8VDFIntvl	Detection interval value range: [0, 29] , in frames, dynamic attribute
u16RstBufSize	The total size of the OD return results
stOdDynamicParamsIn	O D dynamic attribute parameter setting
stOdStaticParamsIn	O D static attribute parameter setting

- Note

  N/A.

- Related API and Data Types

  N/A.

2.1.10 MI_VDF_VgAttr_t

- Description

  Structure that defines VG channel properties

- Definition

```
typedef struct MI_VDF_VgAttr_s
{
    MI_U8 u8Enable;
    MI_U8 u8VgBufCnt;
    MI_U8 u8VDFIntvl;
    MI_U16 u16RstBufSize;

    MI_U16 width;
    MI_U16 height;
    MI_U16 stride;

    float object_size_thd;
    uint8_t indoor;
    uint8_t function_state;
    uint16_t line_number;
    MI_VgLine_t line[4];
    MI_VgRegion_t vg_region;

    MI_VgSet_t stVgParamsIn;
} MI_VDF_VgAttr_t;
```

- Members

Member name	description
u8Enable	Indicates whether the channel is enabled
u8VgBufCnt	VG result buffered number range: [1, 8] static attribute
u8VDFIntvl	Detection interval value range: [0, 29] , in frames, dynamic attribute
u16RstBufSize	The total size of the result returned by the VG
Width	Image width
Height	Image height
Stride	Image of stride
Object_size_thd	Decided to filter out the percentage of objects in the area of interest (If object_size_thd = 1 indicates that the object area is less than one percent of the region of interest in the image frame, it will be ignored.)

Member name	description
Indoor	Indoor or outdoor, 1- indoor, 0- outdoor
Function_state	Set a detection mode for virtual line segments and regional intrusions (VG_VIRTUAL_GATE , indicating that the mode is a virtual line segment VG_REGION_INVASION , indicating that the pattern is a zone intrusion)
Line_number	Set the number of virtual segments, range: [1-4]
Line[4]	Show the structure of the virtual line, a maximum of four may be provided
Vg_region	Related parameters indicating regional invasion
stVgParamsIn	Vg attribute parameter structure, no need to set ,returned by API

- Note

  N/A.

- Related API and Data Types

  N/A.

2.1.11 MI_VDF_ChnAttr_t

- Description

  A structure that defines the channel's working mode properties.

- Definition

```
typedef struct MI_VDF_ChnAttr_s
{
    MI_VDF_WorkMode_e enWorkMode;
    union
    {
        MI_VDF_MdAttr_t stMdAttr;
        MI_VDF_OdAttr_t stOdAttr;
        MI_VDF_VgAttr_t stVgAttr;
    };
}MI_VDF_ChnAttr_t;
```

- Members

Member name	description
enWorkMode	Working mode (motion detection , occlusion detection , electronic fence) static properties
stMdAttr	Motion detection attribute
stOdAttr	Occlusion detection attribute
stVgAttr	Electronic fence properties

- Note

  N/A.

- Related API and Data Types

  N/A.

2.1.12 MDRST_STATUS_t

- Description

  A structure defining a motion detection result of a detection sub-window region

- Definition

```
typedef struct MDRST_STATUS_s
{
    MI_U8   *paddr;
} MDRST_STATUS_t;
```

- Members

Member Name	Description
paddr	Directed motion detection state buf, each area occupies 1byte 0- block does not detect motion, 255- block detects motion

- Note

  N/A.

- Related API and Data Types

  N/A.

2.2. MD Structure List

enumerate	
MDMB_MODE_e	Macro block size of enumeration value
MDSAD_OUT_CTRL_e	Enumeration value of the SAD output format
MDALG_MODE_e	The operation mode enumeration value of the CCL connected region can be used for CCL operation according to the foreground result or the SAD result.
structure	
MDC C L_ctrl_t	Parameter structure that controls CCL operation
MDPoint_t	Coordinate structure
MDROI_t	MD detection area structure
MDSAD_DATA_t	The structure that API MI_MD_ComputeImageSAD output.
MDOBJ_t	Define the information of the connected area: the area and the coordinate position of the smallest bounding rectangle
MDOBJ_DATA_t	Structure of CCL output
MI_MD_IMG_t	The image source structure of motion detection is divided into physical and virtual memory address pointers.
MI_MD_static_param_t	MD static parameter setting structure
MI_MD_param_t	MD dynamic parameter setting structure

2.2.1 MDMB_MODE_e

- Description

  The enumerated size of Micro-block.

- Definition

```
typedef enum MDMB_MODE_E
{
    MDMB_MODE_MB_4x4      = 0x0,
    MDMB_MODE_MB_8x8      = 0x1,
    MDMB_MODE_MB_16x16    = 0x2,
    MDMB_MODE_BUTT
} MDMB_MODE_e;
```

- Members

Member Name	Description
MDMB_MODE_MB_4x4	Use 4x4 Micro-block
MDMB_MODE_MB_8x8	Use 8x8 Micro-block
MDMB_MODE_MB_16x16	Use 16x16 Micro-block

2.2.2 MDSAD_OUT_CTRL_e

- Description

  The enumeration value of the SAD output format.

- Definition

```
typedef enum MDSAD_OUT_CTRL_E
{
    MDSAD_OUT_CTRL_16BIT_SAD    = 0x0,
    MDSAD_OUT_CTRL_8BIT_SAD     = 0x1,
    MDSAD_OUT_CTRL_BUTT
} MDSAD_OUT_CTRL_e;
```

- Members

Member Name	Description
MDSAD_OUT_CTRL_16BIT_SAD	16 bit Output
MDSAD_OUT_CTRL_8BIT_SAD	8 bit Output

2.2.3 MDALG_MODE_e

- Description

  The operation mode enumeration value of the CCL connected
  region can be CCL calculated according to the foreground result or the SAD result

- Definition

```
typedef enum MDALG_MODE_E
{
    MDALG_MODE_FG       = 0x0,
    MDALG_MODE_SAD      = 0x1,
    MDALG_MODE_BUTT
} MDALG_MODE_e;
```

- Members

Member Name	Description
MDALG_MODE_FG	Foreground mode
MDALG_MODE_SAD	SAD Mode

2.2.4 MDCCL_ctrl_t

- Description

  Controls the parameter structure of the CCL execution.

- Definition

```
typedef struct MDCCL_ctrl_s
{
    uint16_t u16InitAreaThr;
    uint16_t u16Step;
} MDCCL_ctrl_t;
```

- Members

Member Name	Description
u16InitAreaThr	Area threshold
u16Step	Increased value of each threshold

2.2.5 MDPoint_t

- Description

Coordinate structure.

- Definition

```
typedef struct MDPoint_s
{
    uint16_t x;
    uint16_t y;
} MDPoint_t;
```

- Members

Member Name	Description
x	X Coordinate
y	Y Coordinate

2.2.6 MDROI_t

- Description

MD detection area structure.

- Definition

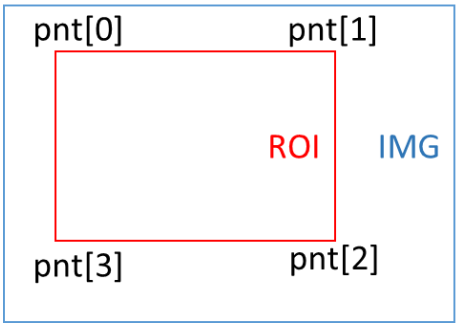
```
typedef struct MDROI_s
{
    uint8_t num;
    MDPoint_t pnt[8];
} MDROI_t;
```

- Members

Member Name	Description
num	MD detection area points, currently only supported to set to 4 points
pnt[8]	Four-point coordinates (must be set to rectangle)

- Note

The requirement is set to a rectangle, num=4 , and the coordinates of the upper left corner are set clockwise in sequence, as shown in the figure.



2.2.7 MDSAD_DATA_t

- Description

The structure of the MI_MD_ComputeImageSAD function output.

- Definition

```
typedef struct MDSAD_DATA_s
{
    void *paddr;
    uint32_t stride;
    MDSAD_OUT_CTRL_e enOutCtrl;
} MDSAD_DATA_t;
```

- Members

Member Name	Description
paddr	Memory address pointer for storing SAD results
stride	Image stride
enOutCtrl	Enumeration value of the SAD output format

2.2.8 MDOBJ_t

- Description

  Define the information of the connected area: the area and the coordinate position of the smallest bounding rectangle.

- Definition

```
typedef struct MDOBJ_s
{
    uint32_t u32Area;
    uint16_t u16Left;
    uint16_t u16Right;
    uint16_t u16Top;
    uint16_t u16Bottom;
} MDOBJ_t;
```

- Members

Member name	description
u32Area	Total number of pixels in a single connected area
u16Left	The top left x coordinate of the smallest rectangle
u16Right	The lower right x coordinate of the smallest rectangle
u16Top	The upper left y coordinate of the smallest rectangle
u16Bottom	The upper left y coordinate of the smallest rectangle

2.2.9 MDOBJ_DATA_t

- Description

  The structure of the MI_MD_CCL output.

- Definition

```
typedef struct MDOBJ_DATA_s
{
    uint8_t u8RegionNum;
    MDOBJ_t *astRegion;
    uint8_t indexofmaxobj;
    uint32_t areaofmaxobj;
    uint32_t areaoftotalobj;

} MDOBJ_DATA_t;
```

- Members

Member name	description
u8RegionNum	Number of connected areas
astRegion	Information about the connected area: the area and the coordinate position of the smallest bounding rectangle The maximum allowable number is 255
Indexofmaxobj	Maximum area connected area index value
Areaofmaxobj	Maximum area of connected area
Areaoftotalobj	The area sum-up of all connected areas

2.2.10 MI_MD_IMG_t

- Description

The image source structure of motion detection is divided into physical and virtual memory address pointers.

- Definition

```
typedef struct MI_MD_IMG_s
{
    void *pu32PhyAddr;
    uint8_t *pu8VirAddr;
} MI_MD_IMG_t;
```

- Members

Member Name	Description
pu32PhyAddr	Entity memory address pointer
pu8VirAddr	Virtual memory address pointer

2.2.11 MI_MD_static_param_t

- Description

MD static parameter setting structure.

- Definition

```
typedef struct MI_MD_static_param_s
{
    uint16_t width;
    uint16_t height;
    uint8_t color;
    uint32_t stride;
    MDMB_MODE_e mb_size;
    MDSAD_OUT_CTRL_e sad_out_ctrl;
```

```
    MDROI_t roi_md;
    MDALG_MODE_e md_alg_mode;
} MI_MD_static_param_t;
```

- Members

Member name	description
Width	Input image width
Height	Input image is high
Stride	Input image of stride
Color	MD input image type
Mb_size	Macro block size enumeration value
Sad_out_ctrl	Enumeration value of the SAD output format
Roi_md	MD detection area structure
Md_alg_mode	Operation mode enumeration value of CCL connected area

2.2.12 MI_MD_param_t

- Description

MD dynamic parameter setting structure.

- Definition

```
typedef struct MI_MD_param_s
{
    uint8_t sensitivity;
    uint16_t learn_rate;
    uint32_t md_thr;
    uint32_t obj_num_max;
} MI_MD_param_t;
```

- Members

Member name	description
Sensitivity	Algorithm sensitivity, range [10, 20, 30, ... 100] , the larger the value, the more sensitive, the sensitivity of the input is not A multiple of 10 , when the feedback is calculated , there may be a value that is not originally entered , there will be +-1 deviation
Learn_rate	In milliseconds, the range [1000, 30000] is used to control how long the front-end object stops moving, as a background image.
Md_thr	Determine the threshold value of the movement, with different setting standards depending on the mode
Obj_num_max	CCL connected area number limit value

2.3. OD Structure List

enumerate	
MI_OD_WIN_STATE	OD detection window results
ODColor_e	Type of OD data source input
ODWindow_e	Type of OD detection window
structure	
ODPoint_t	Coordinate structure
ODROI_t	OD detection area structure
MI_OD_IMG_t	The source structure of the occlusion detection image is divided into physical and virtual memory address pointers.
MI_OD_static_param_t	OD static parameter setting structure
MI_OD_param_t	OD dynamic parameter setting structure

2.3.1 MI_OD_WIN_STATE

- Description

  The result of the OD detection window.

- Definition

```
typedef enum _MI_OD_WIN_STATE
{
    MI_OD_WIN_STATE_TAMPER          = 0,
    MI_OD_WIN_STATE_NON_TAMPER      = 1,
    MI_OD_WIN_STATE_NO_FEATURE      = 2,
    MI_OD_WIN_STATE_FAIL            = -1,
} MI_OD_WIN_STATE;
```

- Members

Member name	description
MI_OD_WIN_STATE_TAMPER	Window is occluded
MI_OD_WIN_STATE_NON_TAMPER	Window is not obscured
MI_OD_WIN_STATE_NO_FEATURE	Insufficient window features
MI_OD_WIN_STATE_FAIL	failure

2.3.2 ODColor_e

- Description

  The type of OD data source input.

- Definition

```
typedef enum
    {
        OD_Y = 1,
        OD_COLOR_MAX
    } ODColor_e;
```

- Members

Member Name	Description
OD_Y	y component in the YUV data source
OD_COLOR_MAX	Enter the maximum value of the image type

2.3.3 ODWindow_e

- Description

  The type of OD detection window, the recommended value is OD_WINDOW_3X3 for testing.

- Definition

```
typedef enum
    {
     OD_WINDOW_1X1 = 0,
        OD_WINDOW_2X2,
     OD_WINDOW_3X3,
        OD_WINDOW_MAX
} ODWindow_e;
```

- Members

Member Name	Description
OD_WINDOW_1X1	1 window
OD_WINDOW_2X2	4 windows
OD_WINDOW_3X3	9 windows
OD_WINDOW_MAX	Maximum window type

2.3.4 ODPoint_t

- Description

  Coordinate structure.

- Definition

```
typedef struct ODPoint_s
{
    uint16_t x;
    uint16_t y;
} ODPoint_t;
```

- Members

Member Name	Description
x	X coordinate
y	Y coordinate

2.3.5 ODROI_t

- Description

OD detection area structure.

- Definition

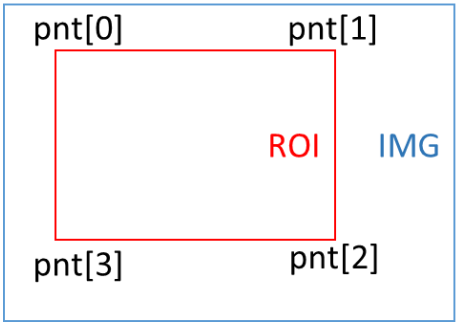
```
typedef struct ODROI_s
{
    uint8_t num;
    ODPoint_t pnt[8];
} ODROI_t;
```

- Members

Member Name	Description
num	OD detection area points, currently only supported to set to 4 points
pnt[8]	Four-point coordinates (must be set to rectangle)

- Note

The requirement is set to a rectangle, num=4 , and the coordinates of the upper left corner are set clockwise in sequence, as shown in the figure.



2.3.6 MI_OD_IMG_t

- Description

  The source structure of the occlusion detection image is divided into physical and virtual memory address pointers.

- Definition

```
typedef struct MI_OD_IMG_s
{
    void *pu32PhyAddr;
    uint8_t *pu8VirAddr;
} MI_OD_IMG_t;
```

- Members

Member Name	Description
pu32PhyAddr	Entity memory address pointer
pu8VirAddr	Virtual memory address pointer

2.3.7 MI_OD_static_param_t

- Description

  OD static parameter setting structure.

- Definition

```
typedef struct MI_OD_static_param_s
{
    uint16_t inImgW;
    uint16_t inImgH;
    uint32_t inImgStride;
    ODColor_e nClrType;
    ODWindow_e div;
    ODROI_t roi_od;
    int32_t alpha;
    int32_t M;
    int32_t MotionSensitivity;
} MI_OD_static_param_t;
```

- Members

Member name	description
inImgW	Input image width
inImgH	Input image is high
inImgStride	Input image stride
nClrType	OD input image type

Member name	description
Div	Type of OD detection window
Roi_od	OD detection area structure
Alpha	Controlling the learning rate of the reference image
M	How many images are updated once for reference images
MotionSensitivity	Mobile sensitivity setting

➢ Note

- Setting range Alpha : 0~10 , it is recommended to set 2 , it is not recommended to change.
- Setting range MotionSensitivity: 0~5 , set 5 to be sensitive to slight sway, easy to report; set 0 means better tolerance for slight sway, will not report, this side refers to the slight sway is wind swaying The recommended initial setting is 5 .
- M is recommended to set 120 , it is not recommended to change

2.3.8 MI_OD_param_t

➢ Description

OD dynamic parameter setting structure.

➢ Definition

```
typedef struct MI_OD_param_s
{
    int32_t thd_tamper;
    int32_t tamper_blk_thd;
    int32_t min_duration;
} MI_OD_param_t;
```

➢ Members

Member name	description
Thd_tamper	Image difference proportional threshold
Tamper_blk_thd	Image occlusion area number threshold value
Min_duration	Image difference duration threshold

➢ Note

- Set the range thd_tamper : 0~10 . If thd_tamper=3, it means that more than 70% of the picture is occluded.
- Setting range tamper_blk_thd: MI_OD_Init type parameter corresponding to the window, if it is OD_WINDOW_3X3, the tamper_blk_thd The maximum number cannot exceed 9, which is 1~9.
- MI_OD_Init parameters such as window type is OD_WINDOW_3X3 (9 subareas) tamper_blk_thd value 4, when the number of sub-areas is blocked up to 4 MI_OD_Runtriggered only returns a value of 1.

- The larger the min_duration value, the longer it takes to detect occlusion.
- The sensitivity of MI_OD_Run can be adjusted by
  setting tamper_blk_thd and min_duration. The recommended values for high, medium and low are as follows:

parameter name	high	in	low
Tamper_blk_thd	2	4	8
Min_duration	5	15	30

2.4. VG Structure List

enumerate	
VgFunction	Detect mode enumeration value
VgRegion_Dir	Enumeration value of the direction of the zone intrusion
structure	
MI_VG_Point_t	Coordinate point corresponding structure
MI_VgLine_t	Structure describing virtual segments and directions
MI_VgRegion_t	Describe the structure of setting up zone intrusion
MI_VgSet_t	Vg corresponds to the structure of the parameter settings
MI_VgResult_t	Corresponding structure of Vg detection result

2.4.1 VgFunction

➢ Description

The enumeration value of the detection mode.

➢ Definition

```
typedef enum _VgFunction
{
    VG_VIRTUAL_GATE        = 2,
    VG_REGION_INVASION     = 3
} VgFunction;
```

➢ Members

Member name	description
VG_VIRTUAL_GATE	Indicates that the mode is a virtual line segment
VG_REGION_INVASION	Indicates that the pattern is a regional invasion

2.4.2 VgRegion_Dir

- Description

  The enumeration value of the direction of the zone intrusion.

- Definition

```
typedef enum _VgRegion_Dir
{
    VG_REGION_ENTER      = 0,
    VG_REGION_LEAVING    = 1,
    VG_REGION_CROSS      = 2
} VgRegion_Dir;
```

- Members

Member name	description
VG_REGION_ENTER	Indicates that an alarm is triggered before entering the alarm zone
VG_REGION_LEAVING	Indicates that you want to leave the alert area to trigger an alert
VG_REGION_CROSS	Indicates that an alarm is triggered as soon as it crosses the alarm zone

2.4.3 MI_VG_Point_t

- Description

  Describe the structure of the virtual line segment and direction.

- Definition

```
typedef struct _VG_Point_t
{
    int32_t x;
    int32_t y;
} MI_VG_Point_t;
```

- Members

Member Name	Description
x	X Coordinate
y	Y Coordinate

2.4.4 MI_VgLine_t

- Description

  Describe the structure of the virtual line segment and direction.

- Definition

```
typedef struct _VG_Line_t
{
    MI_VG_Point_t px;    //point x
    MI_VG_Point_t py;    //point y
    MI_VG_Point_t pdx;   //point direction x
    MI_VG_Point_t pdy;   //point direction y
} MI_VgLine_t;
```

- Members

Member name	description
Px	First line point
Py	Second line point
Pdx	First direction point
Pdy	Second direction point

2.4.5 MI_VgRegion_t

- Description

Describe the structure of setting up zone intrusion.

- Definition

```
typedef struct _VG_Region_t
{
    MI_VG_Point_t p_one;      //point one
    MI_VG_Point_t p_two;      //point two
    MI_VG_Point_t p_three;    //point three
    MI_VG_Point_t p_four;     //point four

    int region_dir;           //Region direction;
} MI_VgRegion_t;
```

- Members

Member name	description
P_one	Describe the first point of the area
P_two	Describe the second point of the area
P_three	Describe the third point of the area
P_four	The fourth point of the description area
Region_dir	Set the direction of the area intrusion

2.4.6 MI_VgSet_t

- Description

  Vg corresponds to the structure of the parameter settings.

- Definition

```
typedef struct _MI_VgSet_t
{
    //Common Information
    float object_size_thd;
    uint16_t line_number;
    uint8_t indoor;

    //Line info
    MI_VG_Point_t fp[4];     //First point
    MI_VG_Point_t sp[4];     //Second point
    MI_VG_Point_t fdp[4];    //First direction point
    MI_VG_Point_t sdp[4];    //Second direction point

    //Function
    uint8_t function_state;

    //Region info
    MI_VG_Point_t first_p;     //First point
    MI_VG_Point_t second_p;    //Second point
    MI_VG_Point_t third_p;     //Third point
    MI_VG_Point_t fourth_p;    //Fourth point

    //Region direction
    uint8_t region_direction;

    //Magic_number
    int32_t magic_number;
} MI_VgSet_t;
```

- Members

Member name	description
Object_size_thd	Decided to filter out the percentage of objects in the area of interest (If object_size_thd = 1 indicates that the object area is less than one percent of the region of interest in the image frame, it will be ignored.)
Line_number	Set the number of virtual segments, range: [1-4]
Indoor	Indoor or outdoor, 1- indoor, 0- outdoor
Fp[4]	The first point of the line array

Member name	description
Sp[4]	Second line point array
Fdp[4]	Third line segment point array
Sdp[4]	Fourth line segment point array
Function_state	Set virtual line segment and area intrusion detection mode
First_p	The first point in the invading area
Second_p	Second point in the invading area
Third_p	The third point of the invading area
Fourth_p	Invasion area is four points lower
Region_direction	Related parameters indicating regional invasion
Magic_number	Magic Number

2.4.7 MI_VgResult_t

- Description

  The structure corresponding to the Vg detection result.

- Definition

```
typedef struct _MI_VgResult_t
{
        int32_t alarm[4];
} MI_VgResult_t;
```

- Members

Member Name	Description
alarm[4]	Vg alarm result

3. ERROR CODE

The area management API error code is shown in the following table.

Table 1: Area Management API Error Codes

Error Code	Macro Definition	Description
0xA0038001	MI_ERR_REG_INVALID_DEVID	Device ID out of range
0xA0038002	MI_ERR_REG_INVALID_CHNID	Channel group number error or invalid area handle
0xA0038003	MI_ERR_REG_ILLEGAL_PARAM	Parameter out of range
0xA0038004	MI_ERR_REG_EXIST	Repeat to create a device, channel or an existing resource
0xA0038005	MI_ERR_REG_UNEXIST	use or destruction test set does not exist Backup, channel or resource
0xA0038006	MI_ERR_REG_NULL_PTR	Parameter function in a null pointer
0xA0038007	MI_ERR_REG_NOT_CONFIG	Module is not configured
0xA0038008	MI_ERR_REG_NOT_SUPPORT	It does not support the parameter or function
0xA0038009	MI_ERR_REG_NOT_PERM	This operation is not allowed, as attempts to modify the static configuration parameters
0xA003800C	MI_ERR_REG_NOMEM	Memory allocation failure, such as lack of system memory
0xA003800D	MI_ERR_REG_NOBUF	Cache allocation fails, if the application data buffer is too big
0xA003800E	MI_ERR_REG_BUF_EMPTY	No data in buffer
0xA003800F	MI_ERR_REG_BUF_FULL	The data buffer is full
0xA0038010	MI_ERR_REG_NOTREADY	The system is not initialized or not to load the appropriate module
0xA0038011	MI_ERR_REG_BADADDR	Address illegal
0xA0038012	MI_ERR_REG_BUSY	System is busy
0xA0038013	E_MI_ERR_CHN_NOT_STARTED	channel not start
0xA0038014	E_MI_ERR_CHN_NOT_STOPED	channel not stop
0xA0038015	E_MI_ERR_NOT_INIT	module not initialized before use it
0xA0038016	E_MI_ERR_NOT_ENABLE	device or channel not enable
0xA0038017	E_MI_ERR_FAILED	unexpected error

Error Code	Macro Definition	Description
0x00000000	MI_MD_RET_SUCCESS	is successful
0x10000401	MI_MD_RET_INIT_ERROR	Failed
0x10000402	MI_MD_RET_IC_CHECK_ERROR	IC model check error
0x10000403	MI_MD_RET_INVALID_HANDLE	OD handle is invalid.
0x10000404	MI_MD_RET_INVALID_PARAMETER	Incorrect parameter setting
0x10000405	MI_MD_RET_MALLOC_ERROR	Memory configuration error
Error Code	**Macro Definition**	**Description**
0x00000000	MI_RET_SUCCESS	is successful
0x10000501	MI_OD_RET_INIT_ERROR	Initialization is failed
0x10000502	MI_OD_RET_IC_CHECK_ERROR	IC model check error
0x10000503	MI_OD_RET_INVALID_HANDLE	OD handle is null.
0x10000504	MI_OD_RET_INVALID_PARAMETER	Incorrect parameter setting
0x10000505	MI_OD_RET_INVALID_WINDOW	Wrong window setting
0x10000506	MI_OD_RET_INVALID_COLOR_TYPE	Color setting error
Error Code	**Macro Definition**	**Description**
0x00000000	MI_VG_RET_SUCCESS	VG Success
0x10000301	MI_VG_RET_INIT_ERROR	VG initialized error
0x10000302	MI_VG_RET_IC_CHECK_ERROR	VG platform check error
0x10000303	MI_VG_RET_INVALID_USER_INFO_POINTER	Invalid user information pointer
0x10000304	MI_VG_RET_INVALID_FUNCTION_STATE	Invalid function state
0x10000305	MI_VG_RET_INVALID_THRESHOLD	Invalid object threshold
0x10000306	MI_VG_RET_INVALID_THRESHOLD_POINTER	Invalid threshold pointer
0x10000307	MI_VG_RET_INVALID_ENVIRONMENT_STATE	Invalid environment state
0x10000308	MI_VG_RET_INVALID_ENVIRONMENT_POINTER	Invalid environment pointer
0x10000309	MI_VG_RET_INVALID_LINE_NUMBER	Invalid line number
0x1000030A	MI_VG_RET_INVALID_LINE_POINTER	Invalid line pointer
0x1000030B	MI_VG_RET_INVALID_LINE_COORDINATE	Invalid line coordinate
0x1000030C	MI_VG_RET_INVALID_LINE_COORDINATE_POINTER	Invalid line coordinate pointer

Error Code	Macro Definition	Description
0x1000030D	MI_VG_RET_INVALID_LINE_MAGIC_NUMBER	Invalid line magic number
0x1000030E	MI_VG_RET_INVALID_REGION_COORDINATE_POINTER	Invalid region coordinate pointer
0x1000030F	MI_VG_RET_INVALID_REGION_MAGIC_NUMBER	Invalid region magic number
0x10000310	MI_VG_RET_INVALID_REGION_COORDINATE	Invalid region coordinate
0x10000311	MI_VG_RET_INVALID_HANDLE	Invalid VG handle
0x10000312	MI_VG_RET_INVALID_HANDLE_MAGIC_NUMBER	Invalid handle magic number
0x10000313	MI_VG_RET_INVALID_INPUT_POINTER	Invalid input pointer
0x10000314	MI_VG_RET_OPERATE_ERROR	VG operate error
0x10000315	MI_VG_RET_INVALID_ALARM_POINTER	Invalid alarm pointer
0x10000316	MI_VG_RET_INVALID_DEBUG_POINTER	Invalid debug pointer