MI VDF API

Version 2.04

REVISION HISTORY

Revision No.	Description	Date
2.03	• Initial release	11/12/2018
2.04	• Refine docs	4/12//2019

TABLE OF CONTENTS

1. API 参考

1.1. 概述

MI_VDF 实现 MD，OD，VG 视频通道的初始化，通道管理，视频检测结果的管理和通道销毁等功能。

1.2. VDF API 列表

API 名	功能
MI_VDF_Init	初始化 MI_VDF 模块
MI_VDF_Uninit	析构 MI_VDF 模块
MI_VDF_CreateChn	创建视频侦测(MD/OD/VG)通道
MI_VDF_DestroyChn	销毁已创建的视频侦测（MD/OD/VG）通道
MI_VDF_SetChnAttr	设置视频侦测（MD/OD/VG）通道属性
MI_VDF_GetChnAttr	获取视频侦测（MD/OD/VG）通道属性
MI_VDF_EnableSubWindow	使能 VDF 视频侦测（MD/OD/VG）通道子窗口
MI_VDF_Run	开始运行视频侦测（MD/OD/VG）
MI_VDF_Stop	停止运行视频侦测（MD/OD/VG）
MI_VDF_GetResult	获取视频侦测（MD/OD/VG）结果
MI_VDF_PutResult	释放视频侦测（MD/OD/VG）结果
MI_VDF_GetLibVersion	获取 VDF 指定通道的版本号
MI_VDF_GetDebugInfo	获取 VDF 指定通道的 Debuginfo(only VG 支持)

1.3. VDF API 说明

1.3.1 MI_VDF_Init

➢ 功能

创建 VDF 通道，初始化 VDF 模块。

➢ 语法

MI_S32 MI_VDF_Init(void);

➢ 形参

无

➢ 返回值

返回值
- 0 成功。
- 非 0 失败，参照错误码。

➢ 依赖

- 头文件：mi_sys.h、mi_md.h、mi_od.h、mi_vg.h、mi_vdf.h
- 库文件：libOD_LINUX.a、libMD_LINUX.a、libVG_LINUX.a、libmi_vdf.a

※ 注意

- MI_VDF_Init 需要在调用 MI_SYS_Init 之后调用。
- 不可以重复调用 MI_VDF_Init。

➢ 举例

无

➢ 相关主题

MI_VDF_Uninit

1.3.2 MI_VDF_Uninit

➢ 功能

析构 MI_VDF 系统， 调用 MI_VDF_Uninit 之前，需要确保已创建的 VDF 通道都已经被 disable，所有通道的视频检测结果都被释放。

➢ 语法

MI_S32 MI_VDF_Uninit (void);

➢ 形参

无

➢ 返回值

返回值
- 0 成功。
- 非 0 失败，参照错误码。

➢ 依赖
- 头文件：mi_sys.h、mi_md.h、mi_od.h、mi_vg.h、mi_vdf.h
- 库文件：libOD_LINUX.a、libMD_LINUX.a、libVG_LINUX.a、libmi_vdf.a

※ 注意
- MI_VDF_Uninit 调用前需要确保所有创建的 VDF 通道都处于 disable 状态。
- MI_VDF_Uninit 调用前需要确保所有创建的 VDF 通道的结果都释放。

➢ 举例

参见 MI_VDF_Init 举例。

➢ 相关主题

无

1.3.3 MI_VDF_CreateChn

➢ 功能

创建视频侦测(MD/OD/VG)通道。

➢ 语法

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

➢ 形参

参数名称	参数含义	输入/输出
VdfChn	指定视频侦测通道号。	输入
pstAttr	设置视频通道属性值。	输入

➢ 返回值

返回值
- 0 成功。
- 非 0 失败，参照错误码。

➢ 依赖
- 头文件：mi_sys.h、mi_md.h、mi_od.h、mi_vg.h、mi_vdf.h
- 库文件：libOD_LINUX.a、libMD_LINUX.a、libVG_LINUX.a、libmi_vdf.a

※ 注意
- VdfChn 不能重复指定。
- VdfChn 的有有效值范围：0<= VdfCh <MI_VDF_CHANNEL_MAX，MI_VDF_CHANNEL_MAX 定义在 mi_vdf_datatype.h。

➢ 举例

无。

➢ 相关主题

MI_VDF_DestroyChn

1.3.4 MI_VDF_DestroyChn

➢ 功能

销毁已创建的视频侦测通道。

➢ 语法

MI_S32 MI_VDF_DestroyChn(MI_VDF_CHANNEL VdfChn);

➢ 形参

参数名称	描述	输入/输出
VdfChn	已经创建的视频侦测通道号。	输入

➢ 返回值

返回值
- 0 成功。
- 非 0 失败，参照错误码。

➢ 依赖

- 头文件：mi_sys.h、mi_md.h、mi_od.h、mi_vg.h、mi_vdf.h
- 库文件：libOD_LINUX.a、libMD_LINUX.a、libVG_LINUX.a、libmi_vdf.a

※ 注意

- VdfChn 必须是已经创建的视频侦测通道。
- VdfChn 的有有效值范围：0<= VdfCh <MI_VDF_CHANNEL_MAX，MI_VDF_CHANNEL_MAX 定义在 mi_vdf_datatype.h。

➢ 举例

无。

➢ 相关主题

无。

1.3.5 MI_VDF_SetChnAttr

➢ 功能

设置视频侦测通道的属性。

➢ 语法

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

➢ 形参

参数名称	描述	输入/输出
VdfChn	已经创建的视频侦测通道号。	输入
pstAttr	源端口配置信息数据结构指针。	输入

➢ 返回值

返回值
- 0 成功。
- 非 0 失败，参照错误码。

➢ 依赖

- 头文件：mi_sys.h、mi_md.h、mi_od.h、mi_vg.h、mi_vdf.h
- 库文件：libOD_LINUX.a、libMD_LINUX.a、libVG_LINUX.a、libmi_vdf.a

※ 注意

- VdfChn 必须是已经创建的视频侦测通道。
- 该接口指定设置视频侦测通道的动态属性值，即 MI_VDF_ChnAttr_t 中的 stMdDynamicParamsIn /stOdDynamicParamsIn/ stVgAttr 部分。

➢ 举例

无

➢ 相关主题

无

1.3.6 MI_VDF_GetChnAttr

➢ 功能

获取视频侦测通道属性。

➢ 语法

MI_S32 MI_VDF_GetChnAttr(MI_VDF_CHANNEL VdfChn, MI_VDF_ChnAttr_t* pstAttr);

➢ 形参

参数名称	描述	输入/输出
VdfChn	已经创建的视频侦测通道号。	输入
pstAttr	用来保存返回的视频侦测通道属性值	输出

➢ 返回值

返回值
- 0 成功。
- 非 0 失败，参照错误码。

➢ 依赖

- 头文件：mi_sys.h、mi_md.h、mi_od.h、mi_vg.h、mi_vdf.h
- 库文件：libOD_LINUX.a、libMD_LINUX.a、libVG_LINUX.a、libmi_vdf.a

※ 注意

无

➢ 举例

无

➢ 相关主题

无

1.3.7 MI_VDF_EnableSubWindow

➢ 功能

使能/关闭指定的视频侦测通道子窗口。

➢ 语法

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

➢ 形参

参数名称	描述	输入/输出
VdfChn	已经创建的视频侦测通道号。	输入
u8Col	子窗口的行地址(保留参数，暂不使用)	输入
u8Row	子窗口的列地址(保留参数，暂不使用)	输入
u8Enable	子窗口的使能控制标志	输入

➢ 返回值

返回值
- 0 成功。
- 非 0 失败，参照错误码。

➢ 依赖

- 头文件：Mi_sys.h、mi_md.h、mi_od.h、mi_vg.h、mi_vdf.h
- 库文件：libOD_LINUX.a、libMD_LINUX.a、libVG_LINUX.a、libmi_vdf.a

※ 注意

- u8Col, u8Row 为 API 接口向上兼容而保留，暂无实际作用，可以直接赋值 0。
- 工作模式目前支持三种：MD，OD，VG。
- MI_VDF_Run/MI_VDF_Stop 函数作用的是整个工作模式（MD/OD/VG）下的所有视频侦测通道；而 MI_VDF_EnableSubWindow 则是作用于单个视频侦测通道。对于一个 VdfChn 来说，只有 MI_VDF_Run 和 MI_VDF_EnableSubWindow 都开启，才会开始运行。只要调用 MI_VDF_Stop 或者调用 MI_VDF_EnableSubWindow 的时候 u8Enable 设置成 flase，都会停止。

➢ 举例

无。

➢ 相关主题

无。

1.3.8 MI_VDF_Run

➢ 功能

开启指定的视频侦测工作模式（MD/OD/VG）。

➢ 语法

MI_S32 MI_VDF_Run(MI_VDF_WorkMode_e enWorkMode);

- 形参

参数名称	描述	输入/输出
enWorkMode	视频侦测（MD/OD/VG）工作模式。	输入

- 返回值

返回值
- 0　成功。
- 非 0　失败，参照错误码。

- 依赖
  - 头文件：mi_sys.h、mi_md.h、mi_od.h、mi_vg.h、mi_vdf.h
  - 库文件：libOD_LINUX.a、libMD_LINUX.a、libVG_LINUX.a、libmi_vdf.a

※ 注意
  - 工作模式目前支持三种：MD，OD，VG。
  - MI_VDF_Run/MI_VDF_Stop 函数作用于整个工作模式（MD/OD/VG）下的所有视频侦测通道。

- 举例

无。

- 相关主题

无。

1.3.9 MI_VDF_Stop

- 功能

停止视频侦测（MD/OD/VG）工作模式。

- 语法

MI_S32 MI_VDF_Stop(MI_VDF_WorkMode_e enWorkMode);

- 形参

参数名称	描述	输入/输出
enWorkMode	视频侦测（MD/OD/VG）工作模式。	输入

- 返回值

返回值
- 0　成功。
- 非 0　失败，参照错误码。

- 依赖
  - 头文件：mi_sys.h、mi_md.h、mi_od.h、mi_vg.h、mi_vdf.h
  - 库文件：libOD_LINUX.a、libMD_LINUX.a、libVG_LINUX.a、libmi_vdf.a

※ 注意
  - 工作模式目前支持三种：MD，OD，VG。

- MI_VDF_Run/MI_VDF_Stop 函数作用于整个工作模式（MD/OD/VG）下的所有视频侦测通道。

➢ 举例

无。

➢ 相关主题

无。

1.3.10 MI_VDF_GetResult

➢ 功能

获取指定视频侦测通道的检测结果。

➢ 语法

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

➢ 形参

参数名称	描述	输入/输出
VdfChn	已经创建的视频侦测通道号。	输入
s32MilliSec	输入时间参数(保留参数，暂不使用)	输入
pstVdfResult	用来保存返回的视频侦测结果	输出

➢ 返回值

返回值
- 0 成功。
- 非 0 失败，参照错误码。

➢ 依赖

- 头文件：mi_sys.h、mi_md.h、mi_od.h、mi_vg.h、mi_vdf.h
- 库文件：libOD_LINUX.a、libMD_LINUX.a、libVG_LINUX.a、libmi_vdf.a

※ 注意

- VdfChn 必须是已经创建的视频侦测通道。
- s32MilliSec 为 API 接口向上兼容而保留，暂无实际作用，可以直接赋值 0。
- 用于保存返回结果的指针不能为空。

➢ 举例

无

➢ 相关主题

无。

1.3.11 MI_VDF_ PutResult

➢ 功能

释放指定视频侦测通道的检测结果。

➢ 语法

MI_S32 MI_VDF_PutResult(MI_VDF_CHANNEL VdfChn, MI_VDF_Result_t* pstVdfResult);

- 形参

参数名称	描述	输入/输出
VdfChn	已经创建的视频侦测通道号。	输入
pstVdfResult	指定需要释放视频通道的结果参数	输入

- 返回值

返回值
- 0　成功。
- 非 0　失败，参照错误码。

- 依赖
  - 头文件：mi_sys.h、mi_md.h、mi_od.h、mi_vg.h、mi_vdf.h
  - 库文件：libOD_LINUX.a、libMD_LINUX.a、libVG_LINUX.a、libmi_vdf.a

※ 注意
  - VdfChn 必须是已经创建的视频侦测通道。

- 举例

无。

- 相关主题

无。

1.3.12 MI_VDF_GetLibVersion

- 功能

  获取 MD/OD/VG 库版本号。

- 语法

  MI_S32 MI_VDF_GetLibVersion(MI_VDF_CHANNEL VdfChn, MI_U32* u32VDFVersion);

- 形参

参数名称	描述	输入/输出
VdfChn	已经创建的视频侦测通道号。	输入
u32VDFVersion	存储版本号的整形指针（不能为空）	输出

- 返回值

  返回值：
  - 0　成功。
  - 非 0　失败，参照错误码。

- 依赖
  - 头文件：mi_sys.h、mi_md.h、mi_od.h、mi_vg.h、mi_vdf.h
  - 库文件：libOD_LINUX.a、libMD_LINUX.a、libVG_LINUX.a、libmi_vdf.a

※ 注意

无。

- 举例

无。

- 相关主题

无。

1.3.13 MI_VDF_GetDebugInfo

- 功能

  获取 MD/OD/VG 库的调试信息。

- 语法

  MI_S32 MI_VDF_GetDebugInfo(MI_VDF_CHANNEL VdfChn, MI_VDF_DebugInfo_t *pstDebugInfo);

- 形参

参数名称	描述	输入/输出
VdfChn	已经创建的视频侦测通道号。	输入
pstDebugInfo	存储 VDF 调试信息的整形指针（不能为空）	输出

- 返回值

  返回值：
  - 0　成功。

非 0　失败，参照错误码。

- 依赖
  - 头文件：mi_sys.h、mi_md.h、mi_od.h、mi_vg.h、mi_vdf.h
  - 库文件：libOD_LINUX.a、libMD_LINUX.a、libVG_LINUX.a、libmi_vdf.a

※ 注意
  - 目前只支持返回 VG 的调试信息，当 VdfChn 的工作模式是 VG 时，才有效。

- 举例

  无。

- 相关主题

  无。

2. 数据类型

2.1. VDF 结构体列表

相关数据类型、数据结构、联合体定义如下：

MI_VDF_WorkMode_e	定义 VDF 工作模式的枚举类型
MI_VDF_Color_e	定义视频侦测通道输入源的枚举类型
MI_VDF_ODWindow_e	定义 OD 时画面的子窗口数量的枚举类型
MI_MD_Result_t	定义 MD 结果的结构体
MI_OD_Result_t	定义 OD 结果的结构体
MI_VG_Result_t	定义 VG 结果的结构体
MI_VDF_Result_t	定义 VDF 工作模式对应结果的结构体
MI_VDF_MdAttr_t	定义 MD 通道属性的结构体
MI_VDF_OdAttr_t	定义 OD 通道属性的结构体
MI_VDF_VgAttr_t	定义 VG 通道属性的结构体
MI_VDF_ChnAttr_t	定义通道工作模式属性的结构体
MDRST_STATUS_t	定义侦测子窗口区域运动检测结果的结构体

2.2. VDF 结构体说明

2.2.1 MI_VDF_WorkMode_e

➢ 说明

定义 VDF 工作模式的枚举类型。

➢ 定义

```
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;
```

➢ 成员

成员名称	描述
E_MI_VDF_WORK_MODE_MD	MD 工作模式
E_MI_VDF_WORK_MODE_OD	OD 工作模式
E_MI_VDF_WORK_MODE_VG	VG 工作模式
E_MI_VDF_WORK_MODE_MAX	工作模式的错误码标识

※ 注意事项

无。

➢ 相关数据接口及类型

无。

2.2.2 MI_VDF_Color_e

➢ 说明

定义视频侦测通道输入源的枚举类型

➢ 定义

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

➢ 成员

成员名称	描述
E_MI_VDF_COLOR_Y	视频侦测通道输入源类型的正确标识
E_MI_VDF_COLOR_MAX	视频侦测通道输入源类型的错误码标识

※ 注意事项

无。

- 相关数据接口及类型
  无。

2.2.3 MI_VDF_ODWindow_e

- 说明
  定义 OD 时画面的子窗口数量的枚举类型

- 定义

```
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;
```

- 成员

成员名称	描述
E_MI_VDF_ODWINDOW_1X1	OD 画面分割为 1 个子窗口
E_MI_VDF_ODWINDOW_2X2	OD 画面分割为 2x2 个子窗口
E_MI_VDF_ODWINDOW_3X3	OD 画面分割为 3x3 个子窗口
E_MI_VDF_ODWINDOW_MAX	OD 画面子窗口数量的错误码标识

※ 注意事项
无。

- 相关数据接口及类型
  无。

2.2.4 MI_MD_Result_t

- 说明
  定义 MD 结果的结构体。

- 定义

```
typedef struct MI_MD_Result_s
{
    MI_U64 u64Pts;                          //The PTS of Image
    MI_U8   u8Enable;                        //=1 表明该结果值有效
    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;
```

➢ 成员

成员名称	描述
u64Pts	图像显示时间
u8Enable	表明该通道是否使能
u8Reading	表明该结果正在被应用层读取
stSubResultSize	描述 MD 返回的 Sad、Obj、ReasultStauts 子结构的 Size
pstMdResultStatus	描述 MD 各区域是否检测到运动
pstMdResultSad	描述 MD 的 Sad 值
pstMdResultObj	描述 MD 的 CCL 值

※ 注意事项

无。

➢ 相关数据类型及接口

无。

2.2.5 MI_OD_Result_t

➢ 说明

定义 OD 结果的结构体。

➢ 定义

```
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;
```

➢ 成员

成员名称	描述
u8Enable	表明该通道是否使能
u8WideDiv	获取 OD 在水平方向的窗口数量
u8HightDiv	获取 OD 在垂直方向的窗口数量
u8DataLen	设置 OD 测试结果的可读大小
u64Pts	图像显示时间
u8RgnAlarm[3][3]	OD 子窗口信息的结果

※ 注意事项

无。

➢ 相关数据类型及接口

无。

2.2.6 MI_VG_Result_t

➢ 说明

定义 VG 结果的结构体。

➢ 定义

```
typedef MI_VgResult_t MI_VG_Result_t;
```

➢ 成员

成员名称	描述
alarm[4]	描述 VG 的检测结果

※ 注意事项

无。

➢ 相关数据类型及接口

无。

2.2.7 MI_VDF_Result_t

➢ 说明

定义 VDF 工作模式对应结果的结构体。

➢ 定义

```
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;
```

➢ 成员

成员名称	描述
enWorkMode	VDF 工作模式（MD/OD/VG）
handle	保存结果的 handle
stMdResult	描述 MD 结果的结构体
stOdResult	描述 OD 结果的结构体
stVgResult	描述 VG 结果的结构体

※ 注意事项

无。

➢ 相关数据类型及接口

无。

2.2.8 MI_VDF_MdAttr_t

➢ 说明

定义 MD 通道属性的结构体

➢ 定义

```
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;
```

➢ 成员

成员名称	描述
u8Enable	表明该通道是否使能
u8MdBufCnt	设置可以缓存的 MD 结果数量 MD 结果缓存个数取值范围：[1, 8] 静态属性
u8VDFIntvl	侦测间隔取值范围：[0, 29]，以帧为单位　动态属性
u16RstBufSize	MD 返回结果的总大小
stSubResultSize	MD 返回结果各子结构体 （Sad 值、CCL 值、ResultStatus）的大小
ccl_ctrl	ccl_ctrl 属性设置
stMdStaticParamsIn	MD 静态属性参数设置
stMdDynamicParamsIn	MD 动态属性参数设置

※ 注意事项

无。

➢ 相关数据类型及接口

无。

2.2.9 MI_VDF_OdAttr_t

➢ 说明

定义 OD 通道属性的结构体

➢ 定义

```
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;
```

➢ 成员

成员名称	描述
u8Enable	表明该通道是否使能
u8OdBufCnt	OD 结果缓存个数取值范围：[1, 16] 静态属性
u8VDFIntvl	侦测间隔取值范围：[0, 29]，以帧为单位　动态属性
u16RstBufSize	OD 返回结果的总大小
stOdDynamicParamsIn	OD 动态属性参数设置
stOdStaticParamsIn	OD 静态属性参数设置

※ 注意事项

无。

➢ 相关数据类型及接口

无。

2.2.10 MI_VDF_VgAttr_t

➢ 说明

定义 VG 通道属性的结构体

➢ 定义

```
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;
```

➢ 成员

成员名称	描述
u8Enable	表明该通道是否使能
u8VgBufCnt	VG 结果缓存个数取值范围：[1, 8] 静态属性
u8VDFIntvl	侦测间隔取值范围：[0, 29]，以帧为单位　动态属性
u16RstBufSize	VG 返回结果的总大小
width	图像的宽度
height	图像的高度
stride	图像的 stride
object_size_thd	决定滤除物体占感兴趣区域的百分比大小 （若 object_size_thd = 1，表示有物体面积小于图像画面中感兴趣区域的百分之一则会忽略不计算。）
indoor	室内或者室外，1-室内，0-室外
function_state	设定虚拟线段与区域入侵有种侦测模式 （VG_VIRTUAL_GATE，表示模式为虚拟线段 VG_REGION_INVASION，表示模式为区域入侵）
line_number	设定虚拟线段的数目，范围：[1-4]
line[4]	表明虚拟线段的结构体，最多可设置 4 条
vg_region	表明区域入侵的相关参数
stVgParamsIn	Vg 属性参数结构体，无需设置，由 API 返回

※ 注意事项

无。

➢ 相关数据类型及接口

无。

2.2.11 MI_VDF_ChnAttr_t

➢ 说明

定义通道工作模式属性的结构体。

➢ 定义

```
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;
```

➢ 成员

成员名称	描述
enWorkMode	工作模式(移动侦测,遮挡侦测，电子围栏)　静态属性
stMdAttr	移动侦测属性
stOdAttr	遮挡侦测属性
stVgAttr	电子围栏属性

※ 注意事项

无。

➢ 相关数据类型及接口

无。

2.2.12 MDRST_STATUS_t

➢ 说明

定义侦测子窗口区域运动检测结果的结构体

➢ 定义

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

➢ 成员

成员名称	描述
paddr	指向运动检测状态的 buf，每个区域占用 1 字节 0-区块未检测运动，255-区块检测到运动

※ 注意事项

无。

➢ 相关数据类型及接口

无。

2.3. MD 结构体列表

枚举	
MDMB_MODE_e	宏块的大小枚举值
MDSAD_OUT_CTRL_e	SAD 输出格式的枚举值
MDALG_MODE_e	CCL 连通区域的运算模式枚举值，可依前景结果或者 SAD 结果做 CCL 运算
结构	
MDCCL_ctrl_t	控制 CCL 运行的参数结构
MDPoint_t	坐标结构
MDROI_t	MD 侦测区域结构
MDSAD_DATA_t	MI_MD_ComputeImageSAD 函式输出的结构
MDOBJ_t	定义连通区域的信息：面积及最小包围矩形的坐标位置
MDOBJ_DATA_t	CCL 输出的结构
MI_MD_IMG_t	移动侦测的图像来源结构，分为实体与虚拟的内存地址指针。
MI_MD_static_param_t	MD 静态参数设置结构
MI_MD_param_t	MD 动态参数设置结构

2.4. MD 结构体说明

2.4.1 MDMB_MODE_e

- 说明

  宏块的大小枚举值。

- 定义

```
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;
```

- 成员

成员名称	描述
MDMB_MODE_MB_4x4	使用 4x4 宏块
MDMB_MODE_MB_8x8	使用 8x8 宏块
MDMB_MODE_MB_16x16	使用 16x16 宏块

2.4.2 MDSAD_OUT_CTRL_e

- 说明
  SAD 输出格式的枚举值。

- 定义

```
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;
```

- 成员

成员名称	描述
MDSAD_OUT_CTRL_16BIT_SAD	16 bit 输出
MDSAD_OUT_CTRL_8BIT_SAD	8 bit 输出

2.4.3 MDALG_MODE_e

- 说明
  CCL 连通区域的运算模式枚举值，可依前景结果或者 SAD 结果做 CCL 运算。

- 定义

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

- 成员

成员名称	描述
MDALG_MODE_FG	前景模式
MDALG_MODE_SAD	SAD 模式

2.4.4 MDCCL_ctrl_t

- 说明
  控制 CCL 运行的参数结构。

- 定义

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

- 成员

成员名称	描述
u16InitAreaThr	区域面积的门坎值
u16Step	每提高一次门坎值的提升值

2.4.5 MDPoint_t

- 说明

  坐标结构。

- 定义

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

- 成员

成员名称	描述
x	X 坐标
y	Y 坐标

2.4.6 MDROI_t

- 说明

  MD 侦测区域结构。

- 定义

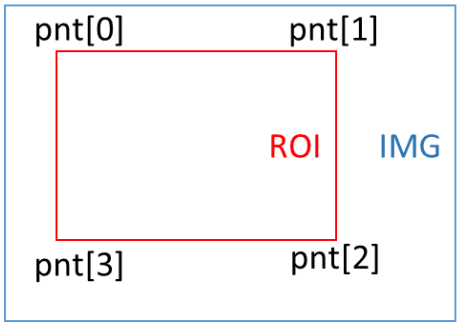
```
typedef struct MDROI_s
{
    uint8_t num;
    MDPoint_t pnt[8];
} MDROI_t;
```

➢ 成员

成员名称	描述
num	MD 侦测区域点数，目前仅支持设为 4 点
pnt[8]	四点坐标(须设为矩形)

※ 注意事项

要求设置为矩形，num=4，左上角坐标顺时针依序设置四个点，如图示。



2.4.7 MDSAD_DATA_t

➢ 说明

MI_MD_ComputeImageSAD 函式输出的结构。

➢ 定义

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

➢ 成员

成员名称	描述
paddr	存放 SAD 结果的内存地址指针
stride	Image stride
enOutCtrl	SAD 输出格式的枚举值

2.4.8 MDOBJ_t

➢ 说明

定义连通区域的信息：面积及最小包围矩形的坐标位置。

- 定义

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

- 成员

成员名称	描述
u32Area	单一连通区域的像素总数
u16Left	最小矩形的左上 x 坐标
u16Right	最小矩形的右下 x 坐标
u16Top	最小矩形的左上 y 坐标
u16Bottom	最小矩形的左上 y 坐标

2.4.9 MDOBJ_DATA_t

- 说明

MI_MD_CCL 输出的结构。

- 定义

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

} MDOBJ_DATA_t;
```

- 成员

成员名称	描述
u8RegionNum	连通区域数量
astRegion	连通区域的信息：面积及最小包围矩形的坐标位置 最大容许数量为 255
indexofmaxobj	最大面积的连通区域索引值

成员名称	描述
areaofmaxobj	最大面积的连通区域面积值
areaoftotalobj	所有连通区域的面积和

2.4.10 MI_MD_IMG_t

- 说明

  移动侦测的图像来源结构，分为实体与虚拟的内存地址指针。

- 定义

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

- 成员

成员名称	描述
pu32PhyAddr	实体的内存地址指针
pu8VirAddr	虚拟的内存地址指针

2.4.11 MI_MD_static_param_t

- 说明

  MD 静态参数设置结构。

- 定义

```
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;
```

➢ 成员

成员名称	描述
width	输入图像宽
height	输入图像高
stride	输入图像的 stride
color	MD 输入图像的类型
mb_size	宏块的大小枚举值
sad_out_ctrl	SAD 输出格式的枚举值
roi_md	MD 侦测区域结构
md_alg_mode	CCL 连通区域的运算模式枚举值

2.4.12 MI_MD_param_t

➢ 说明

MD 动态参数设置结构。

➢ 定义

```
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;
```

➢ 成员

成员名称	描述
sensitivity	算法灵敏度，范围[10,20,30,.....100]，值越大越灵敏，输入的灵敏度如非 10 的倍数，当运算后反馈，有可能不为当初输入的数值，会有 +-1 之偏差
learn_rate	单位毫秒，范围[1000,30000]，用于控制前端物体停止运动多久时，才作为背景画面
md_thr	判断移动的门坎值，随不同模式而有不同设定标准
obj_num_max	CCL 的连通区域数量限制值

2.5. OD 结构体列表

枚举	
MI_OD_WIN_STATE	OD 检测窗口的结果
ODColor_e	OD 数据源输入的类型
ODWindow_e	OD 检测窗口的类型
结构	
ODPoint_t	坐标结构
ODROI_t	OD 侦测区域结构
MI_OD_IMG_t	遮挡侦测的图像来源结构，分为实体与虚拟的内存地址指针。
MI_OD_static_param_t	OD 静态参数设置结构
MI_OD_param_t	OD 动态参数设置结构

2.6. OD 结构体说明

2.6.1 MI_OD_WIN_STATE

- 说明

  OD 检测窗口的结果。

- 定义

```
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;
```

- 成员

成员名称	描述
MI_OD_WIN_STATE_TAMPER	窗口被遮挡
MI_OD_WIN_STATE_NON_TAMPER	窗口没遮挡
MI_OD_WIN_STATE_NO_FEATURE	窗口特征不足
MI_OD_WIN_STATE_FAIL	失败

2.6.2 ODColor_e

- 说明

  OD 数据源输入的类型。

- 定义

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

- 成员

成员名称	描述
OD_Y	YUV 数据源中的 y 分量
OD_COLOR_MAX	输入图像类型的最大值

2.6.3 ODWindow_e

- 说明

  OD 检测窗口的类型，推荐值为 OD_WINDOW_3X3，用于测试。

- 定义

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

- 成员

成员名称	描述
OD_WINDOW_1X1	1 个窗口
OD_WINDOW_2X2	4 个窗口
OD_WINDOW_3X3	9 个窗口
OD_WINDOW_MAX	窗口类型的最大值

2.6.4 ODPoint_t

- 说明

  坐标结构。

- 定义

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

- 成员

成员名称	描述
x	X 坐标
y	Y 坐标

2.6.5 ODROI_t

- 说明

  OD 侦测区域结构。

- 定义

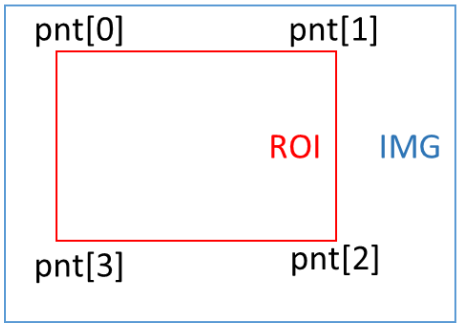
```
typedef struct ODROI_s
{
    uint8_t num;
    ODPoint_t pnt[8];
} ODROI_t;
```

- 成员

成员名称	描述
num	OD 侦测区域点数，目前仅支持设为 4 点
pnt[8]	四点坐标(须设为矩形)

- 注意

  要求设置为矩形，num=4，左上角坐标顺时针依序设置四个点，如图示。



2.6.6 MI_OD_IMG_t

- 说明

  遮挡侦测的图像来源结构，分为实体与虚拟的内存地址指针。

- 定义

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

- 成员

成员名称	描述
pu32PhyAddr	实体的内存地址指针
pu8VirAddr	虚拟的内存地址指针

2.6.7 MI_OD_static_param_t

- 说明

  OD 静态参数设置结构。

- 定义

```
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;
```

➢ 成员

成员名称	描述
inImgW	输入图像宽
inImgH	输入图像高
inImgStride	输入图像 stride
nClrType	OD 输入图像的类型
div	OD 检测窗口的类型
roi_od	OD 侦测区域结构
alpha	控制产生参考图像的学习速率
M	多少张图像更新一次参考图像
MotionSensitivity	移动敏感度设置

※ 注意事项

- 设置范围 alpha：0~10，建议设置 2，不建议更动。
- 设置范围 MotionSensitivity: 0~5，设 5 表示对轻微的晃动都很敏感，容易发报；设 0 表示对轻微晃动的宽容性比较好，不会发报，这边指的轻微晃动是风吹摇曳之类的，建议初始可设 5。
- M 建议设置 120，不建议更动。

2.6.8 MI_OD_param_t

➢ 说明

OD 动态参数设置结构。

➢ 定义

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

➢ 成员

成员名称	描述
thd_tamper	图像差异比例门坎值
tamper_blk_thd	图像被遮挡区域数量门坎值
min_duration	图像差异持续时间门坎值

※ 注意事项

- 设置范围 thd_tamper：0~10。若 thd_tamper=3，表示超过 70%画面遮挡即发报。
- 设置范围 tamper_blk_thd：对应 MI_OD_Init 的窗口类型参数，若为 OD_WINDOW_3X3，则 tamper_blk_thd 最多不可超过 9，即 1~9。
- 例如 MI_OD_Init 的窗口类型参数为 OD_WINDOW_3X3（9 个子区域）tamper_blk_thd 值为 4 时，当被遮挡的子区域的数量达到 4 个才触发 MI_OD_Run 的返回值为 1。
- min_duration 数值越大，检测到被遮挡所需的时间越长。
- MI_OD_Run 的灵敏度可以通过设置 tamper_blk_thd 和 min_duration 来调节。对应高中低的推荐值如下：

参数名	高	中	低
tamper_blk_thd	2	4	8
min_duration	5	15	30

2.7. VG 结构体列表

枚举	
VgFunction	侦测模式的枚举值
VgRegion_Dir	区域入侵的方向的枚举值
结构	
MI_VG_Point_t	坐标点对应的结构
MI_VgLine_t	描述虚拟线段和方向的结构
MI_VgRegion_t	描述设置区域入侵的结构
MI_VgSet_t	Vg 对应参数设置的结构
MI_VgResult_t	Vg 检测结果对应的结构

2.8. VG 结构体说明

2.8.1 VgFunction

- 说明

  侦测模式的枚举值。

- 定义

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

- 成员

成员名称	描述
VG_VIRTUAL_GATE	表示模式为虚拟线段
VG_REGION_INVASION	表示模式为区域入侵

2.8.2 VgRegion_Dir

- 说明

  区域入侵的方向的枚举值。

- 定义

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

- 成员

成员名称	描述
VG_REGION_ENTER	表示要进入警报区域才触发警报
VG_REGION_LEAVING	表示要离开警报区域才触发警报
VG_REGION_CROSS	表示只要穿越警报区域就触发警报

2.8.3 MI_VG_Point_t

- 说明

  坐标点对应的结构。

- 定义

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

- 成员

成员名称	描述
x	X 坐标
y	Y 坐标

2.8.4 MI_VgLine_t

- 说明
  描述虚拟线段和方向的结构。

- 定义

```
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;
```

- 成员

成员名称	描述
px	第一个线段点
py	第二个线段点
pdx	第一个方向点
pdy	第二个方向点

2.8.5 MI_VgRegion_t

- 说明
  描述设置区域入侵的结构。

- 定义

```
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;
```

- 成员

成员名称	描述
p_one	描述区域的第一个点
p_two	描述区域的第二个点
p_three	描述区域的第三个点
p_four	描述区域的第四个点
region_dir	设定区域入侵的方向

2.8.6 MI_VgSet_t

- 说明

  Vg 对应参数设置的结构。

- 定义

```
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;
```

➢ 成员

成员名称	描述
object_size_thd	决定滤除物体占感兴趣区域的百分比大小 （若 object_size_thd = 1，表示有物体面积小于图像画面中感兴趣区域的百分之一则会忽略不计算。）
line_number	设定虚拟线段的数目，范围：[1-4]
indoor	室内或者室外，1-室内，0-室外
fp[4]	第一个线段点数组
sp[4]	第二个线段点数组
fdp[4]	第三个线段点数组
sdp[4]	第四个线段点数组
function_state	设定虚拟线段、区域入侵侦测模式
first_p	入侵区域第一个点
second_p	入侵区域第二个点
third_p	入侵区域第三个点
fourth_p	入侵区域低四个点
region_direction	表明区域入侵的相关参数
magic_number	Magic Number

2.8.7 MI_VgResult_t

➢ 说明

Vg 检测结果对应的结构。

➢ 定义

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

➢ 成员

成员名称	描述
alarm[4]	Vg 的报警结果

3. 错误码

区域管理 API 错误码如表 3-1 所示。

表 3-1 区域管理 API 错误码

错误代码	宏定义	描述
0xA0038001	MI_ERR_REG_INVALID_DEVID	设备 ID 超出合法范围
0xA0038002	MI_ERR_REG_INVALID_CHNID	通道组号错误或无效区域句柄
0xA0038003	MI_ERR_REG_ILLEGAL_PARAM	参数超出合法范围
0xA0038004	MI_ERR_REG_EXIST	重复创建已存在的设备、通道或资源
0xA0038005	MI_ERR_REG_UNEXIST	试图使用或者销毁不存在的设备、通道或者资源
0xA0038006	MI_ERR_REG_NULL_PTR	函数参数中有空指针
0xA0038007	MI_ERR_REG_NOT_CONFIG	模块没有配置
0xA0038008	MI_ERR_REG_NOT_SUPPORT	不支持的参数或者功能
0xA0038009	MI_ERR_REG_NOT_PERM	该操作不允许，如试图修改静态配置参数
0xA003800C	MI_ERR_REG_NOMEM	分配内存失败，如系统内存不足
0xA003800D	MI_ERR_REG_NOBUF	分配缓存失败，如申请的数据缓冲区太大
0xA003800E	MI_ERR_REG_BUF_EMPTY	缓冲区中无数据
0xA003800F	MI_ERR_REG_BUF_FULL	缓冲区中数据满
0xA0038010	MI_ERR_REG_NOTREADY	系统没有初始化或没有加载相应模块
0xA0038011	MI_ERR_REG_BADADDR	地址非法
0xA0038012	MI_ERR_REG_BUSY	系统忙
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 init before use it
0xA0038016	E_MI_ERR_NOT_ENABLE	device or channel not enable
0xA0038017	E_MI_ERR_FAILED	unexpected error
错误代码	宏定义	描述

错误代码	宏定义	描述
0x00000000	MI_MD_RET_SUCCESS	成功
0x10000401	MI_MD_RET_INIT_ERROR	初始化失败
0x10000402	MI_MD_RET_IC_CHECK_ERROR	芯片确认错误
0x10000403	MI_MD_RET_INVALID_HANDLE	MD handle 错误
0x10000404	MI_MD_RET_INVALID_PARAMETER	参数错误
0x10000405	MI_MD_RET_MALLOC_ERROR	内存配置错误
错误代码	**宏定义**	**描述**
0x00000000	MI_RET_SUCCESS	成功
0x10000501	MI_OD_RET_INIT_ERROR	初始化失败
0x10000502	MI_OD_RET_IC_CHECK_ERROR	IC 型号检查错误
0x10000503	MI_OD_RET_INVALID_HANDLE	OD handle is null.
0x10000504	MI_OD_RET_INVALID_PARAMETER	参数设置错误
0x10000505	MI_OD_RET_INVALID_WINDOW	窗口设置错误
0x10000506	MI_OD_RET_INVALID_COLOR_TYPE	色彩设置错误
错误代码	**宏定义**	**描述**
0x00000000	MI_VG_RET_SUCCESS	VG Success
0x10000301	MI_VG_RET_INIT_ERROR	VG init 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
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

错误代码	宏定义	描述
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