



cvsCamCtrl

프로그래머 가이드

1. 개요.....	1
2. 정의 및 상수.....	1
2.1 이벤트 정의.....	1
2.1.1 Grab 이벤트.....	1
2.1.2 장치 이벤트.....	1
2.2 오류 코드 정의.....	2
2.3 장치 정보 정의.....	3
2.4 XML 버전.....	3
2.5 파라미터 가시성.....	4
3. 구조체 정의.....	4
3.1 CVS_IMAGE 구조체.....	4
3.2 CVS_BUFFER 구조체.....	4
3.3 CVS_EVENT 구조체.....	5
4. 색상 변환 (enum ConvertColor).....	5
5. 함수.....	5
5.1 콜백 함수 정의.....	5
5.1.1 Grab 이벤트 콜백.....	6
5.1.2 장치 이벤트 콜백.....	6
5.2 시스템 관리.....	6
5.2.1 ST_InitSystem.....	6
5.2.2 ST_FreeSystem.....	7
5.2.3 ST_IsInitSystem.....	7
5.3 장치 검색 및 정보 조회.....	8
5.3.1 ST_GetAvailableCameraNum.....	8
5.3.2 ST_UpdateDevice.....	8
5.3.3 ST_GetEnumDeviceID.....	9
5.3.4 ST_GetEnumDeviceInfo.....	10
5.4 장치 연결 및 제어.....	11
5.4.1 ST_IsConnectable.....	11
5.4.2 ST_OpenDevice.....	11
5.4.3 ST_IsOpenDevice.....	12
5.4.4 ST_CloseDevice.....	13
5.4.5 ST_GetInterface.....	13
5.5 이미지 획득 및 설정.....	14
5.5.1 ST_AcqStart.....	14
5.5.2 ST_AcqStop.....	15
5.5.3 ST_DoAbortGrab.....	15
5.5.4 ST_SetGrabTimeout.....	16
5.5.5 ST_GetGrabTimeout.....	16
5.5.6 ST_SetBufferCount.....	17
5.5.7 ST_GetBufferCount.....	17
5.6 이미지 처리 및 상태 확인.....	18
5.6.1 ST_GrabImage.....	18
5.6.2 ST_SingleGrabImage.....	19
5.6.3 ST_GetImageAvailable.....	20
5.6.4 ST_GetGrabCount.....	20

5.6.5 ST_GetFrameRate.....	21
5.6.6 ST_GetBandwidth.....	21
5.6.7 ST_GetLastError.....	22
5.6.8 ST_GetLastErrorDescription.....	22
5.7 버퍼 관리.....	23
5.7.1 ST_InitBuffer.....	23
5.7.2 ST_FreeBuffer.....	24
5.8 이미지 색상 변환.....	24
5.8.1 ST_CvtColor.....	24
5.9 로그 및 설정 관리.....	25
5.9.1 ST_SetDetailedLog.....	25
5.9.2 ST_GetDetailedLog.....	26
5.10 레지스터 제어.....	27
5.10.1 ST_SetIntReg.....	27
5.10.2 ST_GetIntReg.....	27
5.10.3 ST_SetFloatReg.....	28
5.10.4 ST_GetFloatReg.....	29
5.10.5 ST_SetEnumReg.....	29
5.10.6 ST_GetEnumReg.....	30
5.10.7 ST_SetStrReg.....	31
5.10.8 ST_GetStrReg.....	31
5.10.9 ST_SetCmdReg.....	32
5.11 레지스터 정보 조회.....	33
5.11.1 ST_GetIntRegRange.....	33
5.11.2 ST_GetFloatRegRange.....	34
5.11.3 ST_GetEnumEntrySize.....	34
5.11.4 ST_GetEnumEntryIntValue.....	35
5.11.5 ST_GetEnumEntryValue.....	36
5.12 파라미터 읽기 및 쓰기.....	37
5.12.1 ST_ReadParam.....	37
5.12.2 ST_WriteParam.....	37
5.13 이벤트 및 콜백 관리.....	38
5.13.1 ST_RegisterGrabCallback.....	38
5.13.2 ST_UnregisterGrabCallback.....	39
5.13.3 ST_RegisterEventCallback.....	40
5.13.4 ST_UnregisterEventCallback.....	41
5.14 XML 및 JSON 파일 관리.....	41
5.14.1 ST_ImportXML.....	41
5.14.2 ST_ExportXML.....	42
5.14.3 ST_ImportJson.....	42
5.14.4 ST_ExportJson.....	43
6. 주의사항.....	44
7. 라이선스 및 저작권.....	44

1. 개요

cvsCamCtrl는 Crevis에서 제공하는 가상 프레임그래버 라이브러리입니다. 본 매뉴얼은 해당 라이브러리에서 제공하는 함수와 오류 코드, 데이터 구조체에 대한 설명을 포함합니다.

2. 정의 및 상수

2.1 이벤트 정의

2.1.1 Grab 이벤트

Grab 이벤트는 **Grab Callback**에서 동작하며, 카메라에서 이미지를 획득하는 과정에서 발생하는 이벤트를 의미합니다. 이 이벤트들은 카메라가 새로운 이미지를 수신했을 때, 오류가 발생했을 때, 또는 **Grab** 동작이 시간 초과되었을 때 자동으로 트리거됩니다. 콜백 함수 내부에서 **Grab** 이벤트를 감지하고 적절한 처리를 수행할 수 있습니다.

이벤트 이름	값	설명
EVENT_NEW_IMAGE	0x0	새로운 이미지가 정상적으로 수신되었으며, 콜백에서 이를 처리 가능
EVENT_GRAB_ERROR	0x1	Grab 과정에서 오류가 발생함 (예: 장치 문제, 데이터 손실 등)
EVENT_GRAB_TIMEOUT	0x2	지정된 시간 내에 Grab 이 완료되지 않아 콜백이 트리거됨

2.1.2 장치 이벤트

장치 이벤트는 카메라 장치의 상태 변화를 감지하고 자동으로 트리거됩니다. 이벤트를 통해 장치의 연결 상태, 노출 종료, 데이터 전송 시작/종료, 온도 상태 등의 중요한 정보를 실시간으로 확인할 수 있습니다. 콜백 함수 내부에서 장치 이벤트를 감지하고, 적절한 처리를 수행할 수 있습니다.

Note: 일부 이벤트는 **GEV**(GigEVision) 또는 **U3V**(USB3Vision)에서만 지원될 수 있으며, 사용 중인 인터페이스에 따라 이벤트 발생 여부가 다를 수 있습니다.

이벤트 이름	값	설명
EVENT_DEVICE_DISCONNECT	0x0010	장치가 연결 해제됨 (예: 네트워크 오류, 전원 차단 등)
EVENT_EXPOSURE_END	0x0014	센서의 노출(Exposure) 과정이 완료됨
EVENT_ACQUISITION_TRANSFER_START	0x0018	이미지 데이터 전송이 시작됨
EVENT_ACQUISITION_TRANSFER_END	0x001C	이미지 데이터 전송이 완료됨
EVENT_OVER_TRIGGER	0x0020	과도한 트리거 신호가 감지됨
EVENT_WARNING_TEMPERATURE_STATE	0x0030	장치의 온도가 경고 수준에 도달함

EVENT_CRITICAL_TEMPERATURE_STATE	0x0034	장치의 온도가 심각한 수준에 도달하여 위험한 상태임
----------------------------------	--------	------------------------------

2.2 오류 코드 정의

오류 코드는 장치 및 API 사용 중 발생할 수 있는 다양한 오류 상태를 나타냅니다. 각 오류 코드는 특정 문제를 나타내며, 오류 처리를 위한 디버깅 및 예외 처리를 쉽게 수행할 수 있도록 설계되었습니다. 일반적으로 0은 정상 상태이며, 음수 값은 오류를 의미합니다.

오류 코드	값	설명
MCAM_ERR_OK	0	정상
MCAM_ERR_GENERIC_ERROR	-1001	일반 오류
MCAM_ERR_NOT_INITIALIZED	-1002	초기화되지 않음
MCAM_ERR_NOT_IMPLEMENTED	-1003	구현되지 않음
MCAM_ERR_BUSY	-1004	장치 사용 중
MCAM_ERR_STATE_ERROR	-1005	상태 오류
MCAM_ERR_NOT_CONNECTED	-1006	연결되지 않음
MCAM_ERR_NOT_FOUND	-1007	찾을 수 없음
MCAM_ERR_NO_MORE_ITEM	-1008	더 이상 항목 없음
MCAM_ERR_INVALID_PARAMETER	-1009	잘못된 매개변수
MCAM_ERR_FILE_ERROR	-1010	파일 오류
MCAM_ERR_TIMEOUT	-1011	시간 초과
MCAM_ERR_ABORTED	-1012	작업 중단됨
MCAM_ERR_BUFFER_TOO_SMALL	-1013	버퍼 크기 부족
MCAM_ERR_CANNOT_OPEN_FILE	-1014	파일 열기 실패
MCAM_ERR_THREAD_ERROR	-1015	스레드 오류
MCAM_ERR_INVALID_DATA_FORMAT	-1016	잘못된 데이터 형식
MCAM_ERR_NOT_ENOUGH_MEMORY	-1017	메모리 부족
MCAM_ERR_CANCEL	-1018	취소됨
MCAM_ERR_PENDING	-1019	대기 중
MCAM_ERR_NO_LICENSE	-1020	라이선스 없음
MCAM_ERR_CANT_READ_MANIFEST	-1021	매니페스트 읽기 실패
MCAM_ERR_NOT_SUPPORTED	-1022	지원되지 않음
MCAM_ERR_ERR_OVERFLOW	-1023	오버플로우 오류
MCAM_ERR_IMAGE_ERROR	-1024	이미지 오류
MCAM_ERR_MISSING_PACKETS	-1025	누락된 패킷
MCAM_ERR_TOO_MANY_RESENDS	-1026	재전송 횟수 초과
MCAM_ERR_RESENDS_FAILURE	-1027	재전송 실패

MCAM_ERR_TOO_MANY_CONSECUTIVE_RESENDS	-1028	연속 재전송 초과
MCAM_ERR_AUTO_ABORTED	-1029	자동 중단됨
MCAM_ERR_BAD_VERSION	-1030	잘못된 버전
MCAM_ERR_NO_MORE_ENTRY	-1031	더 이상 항목 없음
MCAM_ERR_NO_AVAILABLE_DATA	-1032	사용할 데이터 없음
MCAM_ERR_NETWORK_ERROR	-1033	네트워크 오류
MCAM_ERR_RESYNC	-1034	재동기화 필요
MCAM_ERR_CORRUPTED_DATA	-1035	손상된 데이터
MCAM_ERR_GENICAM_XML_ERROR	-1036	GenICam XML 오류
MCAM_ERR_NO_DEVICE	-1037	장치 없음
MCAM_ERR_NO_SYSTEM	-1038	시스템 없음
MCAM_ERR_NOT_OPEN_SYSTEM	-1039	시스템이 열려 있지 않음

2.3 장치 정보 정의

장치 정보 정의는 카메라의 기본적인 속성을 제공하는 매개변수들로 구성됩니다. 이 정보는 장치를 식별하고 관리하는 데 사용되며, 특정 장치에 대한 속성 값을 조회할 수 있도록 지원합니다. 장치 모델명, 시리얼 번호, 네트워크 정보(MAC 주소, IP 주소) 등 다양한 정보를 포함하며, 인터페이스에 따라 지원되는 정보가 다를 수도 있습니다.

Note: 일부 장치 정보는 특정 인터페이스(**GEV** 또는 **U3V**)에서만 제공될 수 있습니다.

정의	값	설명	비고
MCAM_DEVICEINFO_USER_ID	10000	사용자 ID	
MCAM_DEVICEINFO_MODEL_NAME	10001	모델명	
MCAM_DEVICEINFO_SERIAL_NUMBER	10002	시리얼 번호	
MCAM_DEVICEINFO_DEVICE_VERSION	10003	장치 버전	
MCAM_DEVICEINFO_MAC_ADDRESS	10004	MAC 주소	GEV Only
MCAM_DEVICEINFO_IP_ADDRESS	10005	IP 주소	GEV Only
MCAM_DEVICEINFO_CURRENT_SPEED	10006	현재 속도	U3V Only

2.4 XML 버전

XML 버전은 ST_ImportXML, ST_ExportXML 함수를 통해 장치 설정을 저장하거나 로드할 때 사용됩니다.

정의	값	설명
XML_VERSION_1_0	0	GenICam XML 1.0 version (기본 지원)

XML_VERSION_1_1	1	GenICam XML 1.1 version (항상된 기능 포함)
-----------------	---	-------------------------------------

2.5 파라미터 가시성

파라미터 가시성(**Visibility**)은 장치 설정에서 사용자가 볼 수 있는 설정의 수준을 결정합니다. 초보자부터 전문가 수준까지 다양한 단계가 제공되며, 특정 설정은 숨겨질 수도 있습니다.

정의	값	설명
VISIBILITY_BEGINNER	0	초보자 수준 - 기본적인 설정만 표시
VISIBILITY_EXPERT	1	전문가 수준 - 고급 설정 표시
VISIBILITY_GURU	2	최고 수준 - 모든 설정 표시
VISIBILITY_INVISIBLE	3	보이지 않음

3. 구조체 정의

3.1 CVS_IMAGE 구조체

CVS_IMAGE 구조체는 이미지 데이터의 속성을 저장하는 구조체입니다. 이미지의 너비, 높이, 한 줄당 바이트 수, 채널 수 및 이미지 데이터의 포인터를 포함합니다.

```
typedef struct _cvImage {
    int32_t width;      // 이미지 너비 (픽셀 단위)
    int32_t height;     // 이미지 높이 (픽셀 단위)
    int32_t step;       // 한 줄당 바이트 수 (Stride)
    int32_t channels;    // 이미지의 채널 수 (예: 3 = RGB, 1 = 그레이스케일)
    void* pImage;       // 이미지 데이터가 저장된 메모리 포인터
} CVS_IMAGE;
```

3.2 CVS_BUFFER 구조체

CVS_BUFFER 구조체는 개별 프레임 또는 이미지 버퍼에 대한 정보를 포함합니다. 블록 ID, 타임스탬프, 이미지 크기 및 실제 이미지 데이터가 저장됩니다.

```
typedef struct _cvBuffer {
    uint64_t blockID;   // 현재 이미지 블록의 고유 ID
    uint64_t timestamp; // 이미지가 수집된 타임스탬프 (마이크로초 단위)
    uint32_t size;      // 이미지 버퍼 크기 (바이트 단위)
    CVS_IMAGE image;    // 이미지 데이터 구조체
} CVS_BUFFER;
```

3.3 CVS_EVENT 구조체

CVS_EVENT 구조체는 카메라 시스템에서 발생하는 이벤트 정보를 저장합니다. 이벤트 ID, 발생 시간 및 관련 데이터가 포함됩니다.

```
typedef struct _cvsEvent {
    int id;                // 이벤트 ID (예: 장치 연결 해제, 노출 종료 등)
    uint64_t timestamp;    // 이벤트 발생 시간 (마이크로초 단위)
    void* data;            // 이벤트와 관련된 추가 데이터
    uint32_t dataLength;    // 추가 데이터 크기 (바이트 단위)
} CVS_EVENT;
```

4. 색상 변환 (enum ConvertColor)

ConvertColor 열거형(enum)은 이미지 색상 변환을 위한 다양한 변환 옵션을 정의합니다. 이 값들은 이미지 처리 과정에서 특정 색상 공간을 다른 색상 공간으로 변환하는 데 사용됩니다.

```
enum ConvertColor {
    CVP_BGR2RGB = 4,        // BGR -> RGB 변환
    CVP_RGB2BGR = 4,        // RGB -> BGR 변환 (동일한 값 사용)
    CVP_BayerBG2RGB = 48,   // BayerBG 패턴을 RGB로 변환
    CVP_BayerGB2RGB = 49,   // BayerGB 패턴을 RGB로 변환
    CVP_BayerRG2RGB = 46,   // BayerRG 패턴을 RGB로 변환
    CVP_BayerGR2RGB = 47,   // BayerGR 패턴을 RGB로 변환
    CVP_YUV2RGB_UYVY = 107, // YUV(UYVY) 형식을 RGB로 변환
    CVP_YUV2BGR_UYVY = 108, // YUV(UYVY) 형식을 BGR로 변환
    CVP_YUV2RGB_YVYU = 117, // YUV(YVYU) 형식을 RGB로 변환
    CVP_YUV2BGR_YVYU = 118, // YUV(YVYU) 형식을 BGR로 변환
    CVP_YUV2RGB_YUYV = 115, // YUV(YUYV) 형식을 RGB로 변환
    CVP_YUV2BGR_YUYV = 116, // YUV(YUYV) 형식을 BGR로 변환
};
```

5. 함수

5.1 콜백 함수 정의

API에서 Grab 및 장치 이벤트를 처리하기 위해 콜백 함수를 등록할 수 있습니다. 이 섹션에서는 이벤트 발생 시 호출되는 콜백 함수의 정의 및 매개변수를 설명합니다.

5.1.1 Grab 이벤트 콜백

Grab 이벤트가 발생하면 자동으로 호출되는 콜백 함수입니다. 콜백을 통해 새로운 이미지가 수신되었거나 오류가 발생했는지 확인할 수 있습니다.

```
typedef void(*GrabCallback)(
    int32_t eventID,
    const CVS_BUFFER* pBuffer,
    void* pUserDefine
);
```

- **eventID** (IN): 발생한 **Grab** 이벤트의 ID.
- **pBuffer** (IN): **Grab**된 이미지 데이터가 포함된 버퍼.
- **pUserDefine** (IN): 사용자 정의 데이터.

5.1.2 장치 이벤트 콜백

장치 이벤트가 발생하면 호출되는 콜백 함수입니다. 콜백을 통해 장치 상태 변화를 감지할 수 있습니다.

```
typedef void(*EventCallback)(
    const CVS_EVENT* pEvent,
    void* pUserDefine
);
```

- **pEvent** (IN): 장치 이벤트 정보가 포함된 구조체.
- **pUserDefine** (IN): 사용자 정의 데이터.

5.2 시스템 관리

API를 사용하기 위해서는 먼저 시스템을 초기화해야 하며, 사용이 끝난 후에는 시스템을 정리해야 합니다.

이 섹션에서는 시스템 초기화, 해제, 상태 확인 등의 기능을 제공합니다.

5.2.1 ST_InitSystem

시스템을 초기화하는 함수입니다. **API** 사용을 시작하기 전에 반드시 호출해야 하며, 성공적으로 호출되면 시스템이 정상적으로 동작할 준비가 됩니다.

```
CVS_IMPORT_EXPORT CVS_ERROR WINAPI ST_InitSystem();
```

- 매개변수: 없음
- 반환값:
 - **MCAM_ERR_OK** (성공)
 - 오류 코드 (실패)
- 예제 코드:

```
CVS_ERROR result = ST_InitSystem();

if (result != MCAM_ERR_OK)
{
    printf("Failed to InitSystem : %d\n", result);
}
else
{
    printf("System initialized successfully.\n");
}
```

5.2.2 ST_FreeSystem

시스템에서 할당된 리소스를 해제하는 함수입니다. API 사용이 끝난 후 반드시 호출해야 하며, 초기화(ST_InitSystem)된 시스템을 정리하는 역할을 합니다. 이 함수를 호출하면 모든 장치 연결이 해제되고, 내부적으로 사용된 메모리가 정리됩니다.

```
CVS_IMPORT_EXPORT CVS_ERROR WINAPI ST_FreeSystem();
```

- 매개변수: 없음
- 반환값:
 - MCAM_ERR_OK (성공)
 - 오류 코드 (실패)
- 예제 코드:

```
CVS_ERROR result = ST_FreeSystem();

if (result != MCAM_ERR_OK)
{
    printf("Failed to FreeSystem : %d\n", result);
}
else
{
    printf("System resources freed successfully.\n");
}
```

5.2.3 ST_IsInitSystem

시스템이 초기화되었는지 확인하는 함수입니다. ST_InitSystem()이 성공적으로 호출되었는지를 확인할 수 있습니다.

```
CVS_IMPORT_EXPORT CVS_ERROR WINAPI ST_IsInitSystem(
    bool* pFlag
);
```

- 매개변수:
 - pFlag (OUT): 초기화 여부를 저장할 변수 (true = 초기화됨, false = 초기화되지 않음)
- 반환값:

- MCAM_ERR_OK (성공)
 - 오류 코드 (실패)
- 예제 코드:

```
bool isInitialized = false;

CVS_ERROR result = ST_IsInitSystem(&isInitialized);

if (result != MCAM_ERR_OK)
{
    printf("Failed to ST_IsInitSystem : %d\n", result);
}
else
{
    printf("System is %s\n", isInitialized ? "initialized" : "not initialized");
}
```

5.3 장치 검색 및 정보 조회

시스템에 연결된 카메라 장치를 검색하고, 각 장치의 ID 및 상세 정보를 조회할 수 있습니다.

5.3.1 ST_GetAvailableCameraNum

현재 시스템에 연결된 사용 가능한 카메라 개수를 반환하는 함수입니다.

```
CVS_IMPORT_EXPORT CVS_ERROR WINAPI ST_GetAvailableCameraNum(
    uint32_t* pNum
);
```

- 매개변수:
 - pNum (OUT): 사용 가능한 카메라 개수를 저장할 변수
- 반환값:
 - MCAM_ERR_OK (성공)
 - 오류 코드 (실패)
- 예제 코드:

```
uint32_t cameraCount = 0;

CVS_ERROR result = ST_GetAvailableCameraNum(&cameraCount);

if (result != MCAM_ERR_OK)
{
    printf("Failed to GetAvailableCameraNum : %d\n", result);
}
else
{
    printf("Number of available cameras: %d\n", cameraCount);
}
```

5.3.2 ST_UpdateDevice

장치 목록을 갱신하는 함수입니다. 카메라가 새로 연결되거나 제거된 경우 장치 목록을 최신 상태로 업데이트할 수 있습니다.

```
CVS_IMPORT_EXPORT CVS_ERROR WINAPI ST_UpdateDevice(  
    uint32_t uTimeout = 500  
);
```

- 매개변수:
 - uTimeout (IN): 장치 검색을 위한 타임아웃 (기본값: 500ms, 허용 범위: 100~60000ms)
- 반환값:
 - MCAM_ERR_OK (성공)
 - 오류 코드 (실패)
- 예제 코드:

```
CVS_ERROR result = ST_UpdateDevice(1000);  
  
if (result != MCAM_ERR_OK) {  
    printf("Failed to UpdateDevice : %d\n", result);  
}  
else  
{  
    printf("Device list updated successfully.\n");  
}
```

5.3.3 ST_GetEnumDeviceID

지정된 장치의 고유 ID를 조회하는 함수입니다. 장치의 인덱스(EnumNum)를 기반으로 해당 장치의 ID를 반환합니다.

```
CVS_IMPORT_EXPORT CVS_ERROR WINAPI ST_GetEnumDeviceID(  
    uint32_t EnumNum,  
    char* pDeviceID,  
    uint32_t* pSize  
);
```

- 매개변수:
 - EnumNum (IN): 조회할 장치의 인덱스
 - pDeviceID (OUT): 장치 ID를 저장할 문자열 버퍼
 - pSize (IN/OUT): 버퍼 크기. 함수 호출 후 실제 크기로 업데이트됨
- 반환값:
 - MCAM_ERR_OK (성공)
 - 오류 코드 (실패)
- 예제 코드:

```
uint32_t size = 128;
```

```

char deviceID[128];

CVS_ERROR result = ST_GetEnumDeviceID(0, deviceID, &size);

if (result != MCAM_ERR_OK)
{
    printf("Failed to GetEnumDeviceID : %d\n", result);
}
else
{
    printf("Device ID: %s\n", deviceID);
}

```

5.3.4 ST_GetEnumDeviceInfo

지정된 장치의 정보를 조회하는 함수입니다. 장치의 모델명, 시리얼 번호, MAC 주소 등의 정보를 검색할 수 있습니다.

```

CVS_IMPORT_EXPORT CVS_ERROR WINAPI ST_GetEnumDeviceInfo(
    uint32_t EnumNum,
    int32_t DeviceInfoCmd,
    char* pDeviceInfo,
    uint32_t* pSize
);

```

- 매개변수:
 - EnumNum (IN): 조회할 장치의 인덱스
 - DeviceInfoCmd (IN): 요청할 장치 정보 명령어 (예: MCAM_DEVICEINFO_MODEL_NAME)
 - pDeviceInfo (OUT): 장치 정보를 저장할 문자열 버퍼
 - pSize (IN/OUT): 버퍼 크기. 함수 호출 후 실제 크기로 업데이트됨
- 반환값:
 - MCAM_ERR_OK (성공)
 - 오류 코드 (실패)
- 예제 코드:

```

uint32_t size = 128;

char deviceInfo[128];

CVS_ERROR result = ST_GetEnumDeviceInfo(0, MCAM_DEVICEINFO_MODEL_NAME,
deviceInfo, &size);
if (result != MCAM_ERR_OK)
{
    printf("Failed to GetEnumDeviceInfo : %d\n", result);
}
else
{
    printf("Device Model: %s\n", deviceInfo);
}

```

```
}
```

5.4 장치 연결 및 제어

카메라 장치를 연결하거나 해제하고, 현재 연결 상태를 확인하는 기능을 제공합니다.

5.4.1 ST_IsConnectable

지정된 장치가 연결 가능한 상태인지 확인하는 함수입니다.

장치가 네트워크에 연결되어 있거나 정상적으로 작동할 수 있는지 검사합니다.

```
CVS_IMPORT_EXPORT CVS_ERROR WINAPI ST_IsConnectable(  
    uint32_t EnumNum,  
    bool* pFlag  
);
```

- 매개변수:
 - EnumNum (IN): 조회할 장치의 인덱스
 - pFlag (OUT): 장치가 연결 가능한 경우 **true**, 그렇지 않으면 **false**
- 반환값:
 - MCAM_ERR_OK (성공)
 - 오류 코드 (실패)
- 예제 코드:

```
bool isConnectable = false;  
  
CVS_ERROR result = ST_IsConnectable(0, &isConnectable);  
  
if (result != MCAM_ERR_OK)  
{  
    printf("Failed to IsConnectable : %d\n", result);  
}  
else  
{  
    printf("Device is %s\n", isConnectable ? "connectable" : "not  
connectable");  
}
```

5.4.2 ST_OpenDevice

지정된 장치를 열고 사용할 준비를 하는 함수입니다. 성공적으로 호출되면 장치 핸들(hDevice)이 반환되며, 이후 해당 핸들을 사용하여 장치를 제어할 수 있습니다.

```
CVS_IMPORT_EXPORT CVS_ERROR WINAPI ST_OpenDevice(  
    uint32_t EnumNum,  
    int32_t* hDevice,
```

```
bool isDetailedLog = false);
```

- 매개변수:
 - EnumNum (IN): 연결할 장치의 인덱스
 - hDevice (OUT): 연결된 장치 핸들
 - isDetailedLog (IN, 선택): 디버깅을 위한 자세한 로그 활성화 여부 (**false**가 기본값)
- 반환값:
 - MCAM_ERR_OK (성공)
 - 오류 코드 (실패)
- 예제 코드:

```
int32_t hDevice= 0;

CVS_ERROR result = ST_OpenDevice(0, &hDevice, false);

if (result != MCAM_ERR_OK)
{
    printf("Failed to OpenDevice : %d\n", result);
}
else
{
    printf("Device opened successfully. Handle: %d\n", hDevice);
}
```

5.4.3 ST_IsOpenDevice

지정된 장치가 현재 열려 있는지 확인하는 함수입니다. ST_OpenDevice()가 성공적으로 호출된 후 장치가 정상적으로 열렸는지를 검사할 수 있습니다.

```
CVS_IMPORT_EXPORT CVS_ERROR WINAPI ST_IsOpenDevice(
    int32_t hDevice,
    bool* pFlag
);
```

- 매개변수:
 - hDevice (IN): 확인할 장치의 핸들
 - pFlag (OUT): 장치가 열려 있으면 **true**, 그렇지 않으면 **false**
- 반환값:
 - MCAM_ERR_OK (성공)
 - 오류 코드 (실패)
- 예제 코드:

```
bool isOpen = false;

CVS_ERROR result = ST_IsOpenDevice(hDevice, &isOpen);

if (result != MCAM_ERR_OK)
{
```

```

    printf("Failed to IsOpenDevice : %d\n", result);
}
else
{
    printf("Device is %s\n", isOpen ? "open" : "not open");
}

```

5.4.4 ST_CloseDevice

지정된 장치를 닫고 시스템에서 해제하는 함수입니다. 장치를 더 이상 사용하지 않을 경우 호출해야 합니다.

```

CVS_IMPORT_EXPORT CVS_ERROR WINAPI ST_CloseDevice(
    int32_t hDevice
);

```

- 매개변수:
 - hDevice (IN): 닫을 장치의 핸들
- 반환값:
 - MCAM_ERR_OK (성공)
 - 오류 코드 (실패)
- 예제 코드:

```

CVS_ERROR result = ST_CloseDevice(hDevice);

if (result != MCAM_ERR_OK)
{
    printf("Failed to CloseDevice : %d\n", result);
}
else
{
    printf("Device closed successfully.\n");
}

```

5.4.5 ST_GetInterface

지정된 장치의 인터페이스 정보를 조회하는 함수입니다. 인터페이스 정보에는 장치가 연결된 방식(예: **GEV**, **U3V** 등)이 포함됩니다.

```

CVS_IMPORT_EXPORT CVS_ERROR WINAPI ST_GetInterface(
    int32_t hDevice,
    char* pInterface,
    uint32_t* pSize
);

```

- 매개변수:
 - hDevice (IN): 조회할 장치의 핸들
 - pInterface (OUT): 인터페이스 정보를 저장할 문자열 버퍼

- pSize (IN/OUT): 버퍼 크기. 함수 호출 후 실제 크기로 업데이트됨
- 반환값:
 - MCAM_ERR_OK (성공)
 - 오류 코드 (실패)
- 예제 코드:

```
uint32_t size = 128;

char interfaceInfo[128];

CVS_ERROR result = ST_GetInterface(hDevice, interfaceInfo, &size);

if (result != MCAM_ERR_OK)
{
    printf("Failed to GetInterface : %d\n", result);
}
else
{
    printf("Device Interface: %s\n", interfaceInfo);
}
```

5.5 이미지 획득 및 설정

카메라의 이미지 수집(**Grab**)과 관련된 설정을 관리하는 함수들입니다. 이미지 획득을 시작하고 중지하며, **Grab** 타임아웃 및 강제 중단 기능을 제공합니다.

5.5.1 ST_AcqStart

지정된 장치에서 이미지 획득(**Acquisition**)을 시작하는 함수입니다. 장치가 연속적으로 이미지를 수집하기 시작합니다.

```
CVS_IMPORT_EXPORT CVS_ERROR WINAPI ST_AcqStart(
    int32_t hDevice
);
```

- 매개변수:
 - hDevice (IN): 이미지 획득을 시작할 장치의 핸들
- 반환값:
 - MCAM_ERR_OK (성공)
 - 오류 코드 (실패)
- 예제 코드:

```
CVS_ERROR result = ST_AcqStart(hDevice);

if (result != MCAM_ERR_OK)
{
    printf("Failed to AcqStart : %d\n", result);
}
else
{
    }
```

```
    printf("Acquisition started successfully.\n");
}
```

5.5.2 ST_AcqStop

지정된 장치에서 이미지 획득을 중지하는 함수입니다. **ST_AcqStart()**로 시작된 이미지 수집을 중단합니다.

```
CVS_IMPORT_EXPORT CVS_ERROR WINAPI ST_AcqStop(
    int32_t hDevice
);
```

- 매개변수:
 - **hDevice (IN)**: 이미지 획득을 중지할 장치의 핸들
- 반환값:
 - **MCAM_ERR_OK** (성공)
 - 오류 코드 (실패)
- 예제 코드:

```
CVS_ERROR result = ST_AcqStop(hDevice);

if (result != MCAM_ERR_OK)
{
    printf("Failed to AcqStop : %d\n", result);
}
else
{
    printf("Acquisition stopped successfully.\n");
}
```

5.5.3 ST_DoAbortGrab

지정된 장치에서 진행 중인 **Grab** 프로세스를 강제로 중단하는 함수입니다. 이미지 수집이 진행 중일 때 호출하면 즉시 중단됩니다.

```
CVS_IMPORT_EXPORT CVS_ERROR WINAPI ST_DoAbortGrab(
    int32_t hDevice
);
```

- 매개변수:
 - **hDevice (IN)**: **Grab** 프로세스를 중단할 장치의 핸들
- 반환값:
 - **MCAM_ERR_OK** (성공)
 - 오류 코드 (실패)
- 예제 코드:

```
CVS_ERROR result = ST_DoAbortGrab(hDevice);
```

```

if (result != MCAM_ERR_OK)
{
    printf("Failed to DoAbortGrab : %d\n", result);
}
else
{
    printf("Grab process aborted successfully.\n");
}

```

5.5.4 ST_SetGrabTimeout

지정된 장치의 **Grab** 타임아웃을 설정하는 함수입니다. **Grab**이 일정 시간 내에 완료되지 않으면 타임아웃 오류를 반환하도록 설정할 수 있습니다.

```

CVS_IMPORT_EXPORT CVS_ERROR WINAPI ST_SetGrabTimeout(
    int32_t hDevice,
    uint32_t Timeout
);

```

- 매개변수:
 - hDevice (IN): 타임아웃을 설정할 장치의 핸들
 - Timeout (IN): Grab 타임아웃 값 (밀리초 단위)
- 반환값:
 - MCAM_ERR_OK (성공)
 - 오류 코드 (실패)
- 예제 코드:

```

CVS_ERROR result = ST_SetGrabTimeout(hDevice, 3000);

if (result != MCAM_ERR_OK)
{
    printf("Failed to SetGrabTimeout : %d\n", result);
}
else
{
    printf("Grab timeout set to 3000 ms.\n");
}

```

5.5.5 ST_GetGrabTimeout

지정된 장치의 **Grab** 타임아웃 값을 조회하는 함수입니다.

```

CVS_IMPORT_EXPORT CVS_ERROR WINAPI ST_GetGrabTimeout(
    int32_t hDevice,
    uint32_t* pTimeout
);

```

- 매개변수:

- hDevice (IN): 타임아웃을 조회할 장치의 핸들
 - pTimeout (OUT): 현재 설정된 Grab 타임아웃 값 (밀리초 단위)
- 반환값:
 - MCAM_ERR_OK (성공)
 - 오류 코드 (실패)
- 예제 코드:

```
uint32_t timeout = 0;

CVS_ERROR result = ST_GetGrabTimeout(hDevice, &timeout);

if (result != MCAM_ERR_OK)
{
    printf("Failed to GetGrabTimeout : %d\n", result);
}
else
{
    printf("Current grab timeout: %d ms\n", timeout);
}
```

5.5.6 ST_SetBufferCount

장치에서 사용할 이미지 버퍼의 개수를 설정하는 함수입니다. 버퍼 개수를 증가시키면 프레임 드롭 가능성이 줄어들지만 메모리 사용량이 증가합니다.

```
CVS_IMPORT_EXPORT CVS_ERROR WINAPI ST_SetBufferCount(
    int32_t hDevice,
    uint32_t BufferCount
);
```

- 매개변수:
 - hDevice (IN) : 장치 핸들
 - BufferCount (IN) : 설정할 버퍼 개수
- 반환값:
 - MCAM_ERR_OK (성공)
 - 오류 코드 (실패)
- 예제 코드:

```
uint32_t bufferCount = 10;

CVS_ERROR result = ST_SetBufferCount(hDevice, bufferCount);

if (result != MCAM_ERR_OK)
{
    printf("Failed to SetBufferCount : %d\n", result);
}
else
{
    printf("Buffer count set to: %d\n", bufferCount);
}
```

5.5.7 ST_GetBufferCount

현재 장치에서 설정된 버퍼 개수를 조회하는 함수입니다.

```
CVS_IMPORT_EXPORT CVS_ERROR WINAPI ST_GetBufferCount(
    int32_t hDevice,
    uint32_t* pBufferCount
);
```

- 매개변수:
 - hDevice (IN) : 장치 핸들
 - pBufferCount (OUT) : 버퍼 개수를 저장할 변수
- 반환값:
 - MCAM_ERR_OK (성공)
 - 오류 코드 (실패)
- 예제 코드:

```
uint32_t bufferCount = 0;

CVS_ERROR result = ST_GetBufferCount(hDevice, &bufferCount);

if (result != MCAM_ERR_OK)
{
    printf("Failed to GetBufferCount : %d\n", result);
}
else
{
    printf("Current buffer count: %d\n", bufferCount);
}
```

5.6 이미지 처리 및 상태 확인

이미지를 획득한 후, 상태를 확인하고 관련된 정보를 조회하는 함수들입니다.

5.6.1 ST_GrabImage

장치에서 연속적으로 이미지를 획득(Grab)하는 함수입니다. **ST_AcqStart** 함수를 먼저 호출하여 이미지 획득을 시작해야 하며, **ST_AcqStop** 함수를 호출하면 이미지 획득이 중단됩니다.

ST_GrabImage 함수는

ST_AcqStart 가 활성화된 상태에서 이미지를 버퍼에 저장하며, 획득된 이미지는 **pBuffer** 에 저장됩니다.

```
CVS_IMPORT_EXPORT CVS_ERROR WINAPI ST_GrabImage(
    int32_t hDevice,
    CVS_BUFFER* pBuffer
);
```

- 매개변수:
 - hDevice (IN) : 장치 핸들

- pBuffer (OUT) : 획득한 이미지 데이터를 저장할 CVS_BUFFER 구조체 포인터
- 반환값:
 - MCAM_ERR_OK (성공)
 - 오류 코드 (실패)
- 예제 코드:

```
CVS_BUFFER buffer;

CVS_ERROR result = ST_GrabImage(hDevice, &buffer);

if (result != MCAM_ERR_OK)
{
    printf("Failed to GrabImage : %d\n", result);
}
else
{
    printf("Image grabbed successfully.\n");
}
```

5.6.2 ST_SingleGrabImage

단일 프레임 이미지를 획득(Grab)하는 함수입니다. ST_AcqStart를 호출하지 않고도 독립적으로 실행할 수 있으며, 연속적인 이미지 획득이 필요하지 않을 때 사용됩니다. 획득된 이미지는 pBuffer에 저장됩니다. 일반적으로 ST_GrabImage는 연속 촬영을 위한 함수이고, ST_SingleGrabImage는 단발성 이미지 캡처에 적합합니다.

```
CVS_IMPORT_EXPORT CVS_ERROR WINAPI ST_SingleGrabImage(
    int32_t hDevice,
    CVS_BUFFER* pBuffer
);
```

- 매개변수:
 - hDevice (IN) : 장치 핸들
 - pBuffer (OUT) : 획득한 이미지 데이터를 저장할 CVS_BUFFER 구조체 포인터
- 반환값:
 - MCAM_ERR_OK (성공)
 - 오류 코드 (실패)
- 예제 코드:

```
CVS_BUFFER buffer;

CVS_ERROR result = ST_SingleGrabImage(hDevice, &buffer);

if (result != MCAM_ERR_OK)
{
    printf("Failed to SingleGrabImage : %d\n", result);
}
else
{
    printf("Single image grabbed successfully.\n");
}
```

```
}
```

5.6.3 ST_GetImageAvailable

장치에서 이미지가 획득 가능한 상태인지 확인하는 함수입니다. 이미지 획득이 완료되었는지 확인하는 데 사용할 수 있습니다.

```
CVS_IMPORT_EXPORT CVS_ERROR WINAPI ST_GetImageAvailable(  
    int32_t hDevice,  
    uint32_t* pFlag  
);
```

- 매개변수:
 - hDevice (IN) : 장치 핸들
 - pFlag (OUT) : 이미지가 존재하면 true, 없으면 false
- 반환값:
 - MCAM_ERR_OK (성공)
 - 오류 코드 (실패)
- 예제 코드:

```
uint32_t isAvailable = 0;  
  
CVS_ERROR result = ST_GetImageAvailable(hDevice, &isAvailable);  
  
if (result != MCAM_ERR_OK)  
{  
    printf("Failed to GetImageAvailable : %d\n", result);  
}  
else  
{  
    printf("Image available: %d\n", isAvailable);  
}
```

5.6.4 ST_GetGrabCount

장치에서 현재까지 획득한 이미지 개수와 발생한 오류 개수를 조회하는 함수입니다.

```
CVS_IMPORT_EXPORT CVS_ERROR WINAPI ST_GetGrabCount(  
    int32_t hDevice,  
    uint64_t* Count,  
    uint64_t* ErrorCount  
);
```

- 매개변수:
 - hDevice (IN) : 장치 핸들
 - Count (OUT) : 획득한 이미지 개수
 - ErrorCount (OUT) : 획득 과정에서 발생한 오류 개수

- 반환값:
 - MCAM_ERR_OK (성공)
 - 오류 코드 (실패)
- 예제 코드:

```
uint64_t grabCount = 0, errorCount = 0;

CVS_ERROR result = ST_GetGrabCount(hDevice, &grabCount, &errorCount);

if (result != MCAM_ERR_OK)
{
    printf("Failed to GetGrabCount : %d\n", result);
}
else
{
    printf("Grab count: %llu, Error count: %llu\n", grabCount, errorCount);
}
```

5.6.5 ST_GetFrameRate

장치에서 현재 설정된 프레임 속도를 조회하는 함수입니다. 카메라가 초당 몇 개의 프레임을 촬영하는지 확인할 수 있습니다.

```
CVS_IMPORT_EXPORT CVS_ERROR WINAPI ST_GetFrameRate(
    int32_t hDevice,
    double* FrameRate
);
```

- 매개변수:
 - hDevice (IN) : 장치 핸들
 - FrameRate (OUT) : 현재 프레임 속도 (fps)
- 반환값:
 - MCAM_ERR_OK (성공)
 - 오류 코드 (실패)
- 예제 코드:

```
double frameRate = 0.0;

CVS_ERROR result = ST_GetFrameRate(hDevice, &frameRate);

if (result != MCAM_ERR_OK)
{
    printf("Failed to GetFrameRate : %d\n", result);
}
else
{
    printf("Current frame rate: %.2f fps\n", frameRate);
}
```


5.6.6 ST_GetBandwidth

장치에서 현재 사용 중인 네트워크 또는 인터페이스의 대역폭을 조회하는 함수입니다.

```
CVS_IMPORT_EXPORT CVS_ERROR WINAPI ST_GetBandwidth(
    int32_t hDevice,
    double* Bandwidth
);
```

- 매개변수:
 - hDevice (IN) : 장치 핸들
 - Bandwidth (OUT) : 현재 사용 중인 대역폭 (Mbps)
- 반환값:
 - MCAM_ERR_OK (성공)
 - 오류 코드 (실패)
- 예제 코드:

```
double bandwidth = 0.0;

CVS_ERROR result = ST_GetBandwidth(hDevice, &bandwidth);

if (result != MCAM_ERR_OK)
{
    printf("Failed to GetBandwidth : %d\n", result);
}
else
{
    printf("Current bandwidth usage: %.2f Mbps\n", bandwidth);
}
```

5.6.7 ST_GetLastError

장치에서 발생한 마지막 오류 코드를 조회하는 함수입니다.

```
CVS_IMPORT_EXPORT CVS_ERROR WINAPI ST_GetLastError(
    int32_t hDevice
);
```

- 매개변수:
 - hDevice (IN) : 장치 핸들
- 반환값:
 - 마지막 오류 코드 (MCAM_ERR_XXX)
- 예제 코드:

```
CVS_ERROR lastError = ST_GetLastError(hDevice);

printf("Last device error code: %d\n", lastError);
```

5.6.8 ST_GetLastErrorDescription

장치에서 발생한 마지막 오류의 설명을 문자열로 반환하는 함수입니다. ST_GetLastError 함수와 함께 사용하여 상세한 오류 정보를 얻을 수 있습니다.

```
CVS_IMPORT_EXPORT const char* WINAPI ST_GetLastErrorDescription(  
    int32_t hDevice  
);
```

- 매개변수:
 - hDevice (IN) : 장치 핸들
- 반환값:
 - 오류 설명 문자열
- 예제 코드:

```
const char* errorDescription = ST_GetLastErrorDescription(hDevice);  
  
printf("Last device error: %s\n", errorDescription);
```

5.7 버퍼 관리

이미지 데이터를 저장하는 메모리 버퍼를 초기화하고 해제하는 기능을 제공합니다.

5.7.1 ST_InitBuffer

ST_InitBuffer 함수는 CVS_BUFFER 구조체를 초기화하는 함수입니다. 영상 데이터를 저장할 버퍼를 생성하며, 채널 수를 지정할 수 있습니다. ST_GrabImage 또는 ST_SingleGrabImage 를 호출하기 전에 반드시 호출해야 합니다.

기본값(0)으로 설정하면 연결된 카메라의 설정값에 따라 자동으로 채널이 결정됩니다. 따라서, 별도로 채널을 지정하지 않으면 카메라의 출력 형식에 따라 적절한 값이 설정됩니다.

```
CVS_IMPORT_EXPORT CVS_ERROR WINAPI ST_InitBuffer(  
    int32_t hDevice,  
    CVS_BUFFER* pBuffer,  
    int32_t channels = 0  
);
```

- 매개변수:
 - hDevice (IN) : 장치 핸들
 - pBuffer (OUT) : 초기화할 CVS_BUFFER 구조체 포인터
 - channels (IN) : 버퍼의 채널 수 (기본값 0, 기본값 사용 시 카메라 설정값을 따름)
- 반환값:
 - MCAM_ERR_OK (성공)
 - 오류 코드 (실패)
- 예제 코드:

```
CVS_BUFFER buffer;
```

```
CVS_ERROR result = ST_InitBuffer(hDevice, &buffer, 0);

if (result != MCAM_ERR_OK)
{
    printf("Failed to InitBuffer : %d\n", result);
}
else
{
    printf("Buffer initialized successfully.\n");
}
```

5.7.2 ST_FreeBuffer

ST_FreeBuffer 함수는 ST_InitBuffer 로 생성한 CVS_BUFFER의 메모리를 해제하는 함수입니다. 더 이상 사용하지 않는 버퍼를 해제하여 메모리 누수를 방지합니다.

ST_FreeBuffer 호출 후 해당 버퍼를 다시 사용하려면 ST_InitBuffer 를 다시 호출해야 합니다.

```
CVS_IMPORT_EXPORT CVS_ERROR WINAPI ST_FreeBuffer(
    CVS_BUFFER* pBuffer
);
```

- 매개변수:
 - pBuffer (IN) : 해제할 CVS_BUFFER 구조체 포인터
- 반환값:
 - MCAM_ERR_OK (성공)
 - 오류 코드 (실패)
- 예제 코드:

```
CVS_ERROR result = ST_FreeBuffer(&buffer);

if (result != MCAM_ERR_OK)
{
    printf("Failed to FreeBuffer : %d\n", result);
}
else
{
    printf("Buffer freed successfully.\n");
}
```

5.8 이미지 색상 변환

이미지의 색상 공간을 변환하는 기능을 제공합니다.

5.8.1 ST_CvtColor

ST_CvtColor 함수는 이미지를 하나의 색 공간에서 다른 색 공간으로 변환하는 기능을 제공합니다. 입력 이미지를 변환 코드(code)에 따라 새로운 색 공간으로 변환하여 출력

버퍼(pDest)에 저장합니다. 이 함수는 ST_GrabImage 또는 ST_SingleGrabImage로 획득한 이미지를 후처리할 때 유용합니다.

```
CVS_IMPORT_EXPORT CVS_ERROR WINAPI ST_CvtColor(
    CVS_BUFFER pSrc,
    CVS_BUFFER* pDest,
    int32_t code
);
```

- 매개변수:
 - pSrc (IN) : 변환할 원본 이미지가 저장된 CVS_BUFFER 구조체
 - pDest (OUT) : 변환된 이미지가 저장될 CVS_BUFFER 구조체 포인터
 - code (IN) : 색 공간 변환 코드 (예: RGB → GRAY, BGR → YUV 등)
- 반환값:
 - MCAM_ERR_OK (성공)
 - 오류 코드 (실패)
- 예제 코드:

```
CVS_BUFFER destBuffer;

// destBuffer 초기화
ST_InitBuffer(hDevice, &destBuffer, 3);

// BayerRG8에서 RGB 변환
int32_t code = CVP_BayerRG2RGB;

// srcBuffer는 이전에 Grab하여 획득한 이미지 버퍼
CVS_ERROR result = ST_CvtColor(srcBuffer, &destBuffer, code);

if (result != MCAM_ERR_OK)
{
    printf("Failed to convert color : %d\n", result);
}
else
{
    printf("Color conversion successful.\n");
}

// destBuffer 해제
ST_FreeBuffer(&destBuffer);
```

5.9 로그 및 설정 관리

디버깅을 위해 상세 로그를 활성화하거나 비활성화하는 기능을 제공합니다.

5.9.1 ST_SetDetailedLog

ST_SetDetailedLog 함수는 특정 장치(hDevice)에 대한 상세 로그 기능을 활성화하거나 비활성화하는 기능을 제공합니다. 상세 로그를 활성화하면 장치의 작동 과정에서 발생하는

이벤트나 오류를 더 상세하게 기록할 수 있습니다. 이 기능은 문제 해결 및 디버깅을 위해 유용합니다.

```
CVS_IMPORT_EXPORT CVS_ERROR WINAPI ST_SetDetailedLog(
    int32_t hDevice,
    bool Flag
);
```

- 로그 파일 저장 위치:
 - Windows: C:\ProgramData\CREVIS\Logs\장치 별
 - Linux: ~/CREVIS/장치 별
- 로그 관리 정책:
 - 각 장치별 폴더 내에 500개 이상의 파일이 존재 하거나,
 - 폴더 크기가 5MB를 초과 하는 경우,
 - 가장 오래된 파일부터 자동으로 삭제하여 저장 공간을 확보합니다.
- 매개변수:
 - hDevice (IN) : 로그를 설정할 장치의 핸들
 - Flag (IN) : 상세 로그 활성화 여부 (true = 활성화, false = 비활성화)
- 반환값:
 - MCAM_ERR_OK (성공)
 - 오류 코드 (실패)
- 예제 코드:

```
CVS_ERROR result = ST_SetDetailedLog(hDevice, true);

if (result != MCAM_ERR_OK)
{
    printf("Failed to enable detailed log : %d\n", result);
}
else
{
    printf("Detailed log enabled successfully. Logs saved in:\n");
}
```

5.9.2 ST_GetDetailedLog

ST_GetDetailedLog 함수는 특정 장치(hDevice)에서 현재 상세 로그 기능이 활성화되어 있는지를 확인하는 기능을 제공합니다. 현재 설정된 상세 로그 상태를 반환받을 수 있습니다.

```
CVS_IMPORT_EXPORT CVS_ERROR WINAPI ST_GetDetailedLog(
    int32_t hDevice,
    bool* pFlag
);
```

- 매개변수:
 - hDevice (IN) : 로그 상태를 확인할 장치의 핸들
 - pFlag (OUT) : 상세 로그 상태를 저장할 변수 (true = 활성화됨, false = 비활성화됨)
- 반환값:

- MCAM_ERR_OK (성공)
 - 오류 코드 (실패)
- 예제 코드:

```
bool logStatus = false;
CVS_ERROR result = ST_GetDetailedLog(hDevice, &logStatus);

if (result != MCAM_ERR_OK)
{
    printf("Failed to get detailed log status : %d\n", result);
}
else
{
    printf("Detailed log status : %s\n", logStatus ? "Enabled" : "Disabled");
}
```

5.10 레지스터 제어

장치 내부의 특정 레지스터 값을 설정하거나 조회하는 기능을 제공합니다.

5.10.1 ST_SetIntReg

지정된 장치의 정수형(Integer) 레지스터 값을 설정합니다.

```
CVS_IMPORT_EXPORT CVS_ERROR WINAPI ST_SetIntReg(
    int32_t hDevice,
    const char* NodeName,
    int64_t val
);
```

- 매개변수:
 - hDevice (IN): 장치 핸들
 - NodeName (IN): 설정할 레지스터의 노드 이름
 - val (IN): 설정할 정수 값
- 반환값:
 - MCAM_ERR_OK (성공)
 - 오류 코드 (실패)
- 예제 코드:

```
CVS_ERROR result = ST_SetIntReg(hDevice, "Width", 1000);

if (result != MCAM_ERR_OK)
{
    printf("Failed to set integer register: %d\n", result);
}
else
{
    printf("Successfully set integer register.\n");
}
```

5.10.2 ST_GetIntReg

지정된 장치의 정수형(Integer) 레지스터 값을 조회합니다.

```
CVS_IMPORT_EXPORT CVS_ERROR WINAPI ST_GetIntReg(
    int32_t hDevice,
    const char* NodeName,
    int64_t* pVal
);
```

- 매개변수:
 - hDevice (IN): 장치 핸들
 - NodeName (IN): 조회할 레지스터의 노드 이름
 - pVal (OUT): 조회된 정수 값을 저장할 변수 포인터
- 반환값:
 - MCAM_ERR_OK (성공)
 - 오류 코드 (실패)
- 예제 코드:

```
int64_t value = 0;

CVS_ERROR result = ST_GetIntReg(hDevice, "Width", &value);

if (result != MCAM_ERR_OK)
{
    printf("Failed to get integer register: %d\n", result);
}
else
{
    printf("Width: %lld\n", value);
}
```

5.10.3 ST_SetFloatReg

지정된 장치의 실수형(Float) 레지스터 값을 설정합니다.

```
CVS_IMPORT_EXPORT CVS_ERROR WINAPI ST_SetFloatReg(
    int32_t hDevice,
    const char* NodeName,
    double fVal
);
```

- 매개변수:
 - hDevice (IN): 장치 핸들
 - NodeName (IN): 설정할 레지스터의 노드 이름
 - val (IN): 설정할 실수 값
- 반환값:
 - MCAM_ERR_OK (성공)
 - 오류 코드 (실패)

- 예제 코드:

```
CVS_ERROR result = ST_SetFloatReg(hDevice, "ExposureTime", 1000.0);

if (result != MCAM_ERR_OK)
{
    printf("Failed to set float register: %d\n", result);
}
else
{
    printf("Successfully set float register.\n");
}
```

5.10.4 ST_GetFloatReg

지정된 장치의 실수형(Float) 레지스터 값을 조회합니다.

```
CVS_IMPORT_EXPORT CVS_ERROR WINAPI ST_GetFloatReg(
    int32_t hDevice,
    const char* NodeName,
    double* pFval
);
```

- 매개변수:
 - hDevice (IN): 장치 핸들
 - NodeName (IN): 조회할 레지스터의 노드 이름
 - pFval (OUT): 조회된 실수 값을 저장할 변수 포인터
- 반환값:
 - MCAM_ERR_OK (성공)
 - 오류 코드 (실패)
- 예제 코드:

```
double value = 0.0;

CVS_ERROR result = ST_GetFloatReg(hDevice, "ExposureTime", &value);

if (result != MCAM_ERR_OK)
{
    printf("Failed to get float register: %d\n", result);
}
else
{
    printf("Exposure Time: %.2f\n", value);
}
```

5.10.5 ST_SetEnumReg

지정된 장치의 열거형(Enum) 레지스터 값을 설정합니다.


```
CVS_IMPORT_EXPORT CVS_ERROR WINAPI ST_SetEnumReg(
    int32_t hDevice,
    const char* NodeName,
    char* val
);
```

- 매개변수:
 - hDevice (IN): 장치 핸들
 - NodeName (IN): 설정할 레지스터의 노드 이름
 - val(IN): 설정할 열거형 값
- 반환값:
 - MCAM_ERR_OK (성공)
 - 오류 코드 (실패)
- 예제 코드:

```
CVS_ERROR result = ST_SetEnumReg(hDevice, "TriggerMode", "On");

if (result != MCAM_ERR_OK)
{
    printf("Failed to set enum register: %d\n", result);
}
else
{
    printf("Successfully set enum register.\n");
}
```

5.10.6 ST_GetEnumReg

지정된 장치의 열거형(Enum) 레지스터 값을 조회합니다.

```
CVS_IMPORT_EXPORT CVS_ERROR WINAPI ST_GetEnumReg(
    int32_t hDevice,
    const char* NodeName,
    char* pInfo,
    uint32_t* pSize
);
```

- 매개변수:
 - hDevice (IN): 장치 핸들
 - NodeName (IN): 조회할 레지스터의 노드 이름
 - pInfo (OUT): 조회된 열거형 값을 저장할 문자열 버퍼
 - pSize (IN/OUT): 버퍼 크기. 함수 호출 후 실제 크기로 업데이트됨
- 반환값:
 - MCAM_ERR_OK (성공)
 - 오류 코드 (실패)
- 예제 코드:

```
char triggerMode[128];
```

```

uint32_t size = sizeof(triggerMode);

CVS_ERROR result = ST_GetEnumReg(hDevice, "TriggerMode", triggerMode, &size);

if (result != MCAM_ERR_OK)
{
    printf("Failed to get enum register: %d\n", result);
}
else
{
    printf("Trigger Mode: %s\n", triggerMode);
}

```

5.10.7 ST_SetStrReg

지정된 장치의 문자열(String) 레지스터 값을 설정합니다.

```

CVS_IMPORT_EXPORT CVS_ERROR WINAPI ST_SetStrReg(
    int32_t hDevice,
    const char* NodeName,
    char* val
);

```

- 매개변수:
 - hDevice (IN): 장치 핸들
 - NodeName (IN): 설정할 레지스터의 노드 이름
 - val (IN): 설정할 문자열 값
- 반환값:
 - MCAM_ERR_OK (성공)
 - 오류 코드 (실패)
- 예제 코드:

```

CVS_ERROR result = ST_SetStrReg(hDevice, "DeviceUserID", "Camera_01");

if (result != MCAM_ERR_OK)
{
    printf("Failed to set string register: %d\n", result);
}
else
{
    printf("Successfully set string register.\n");
}

```

5.10.8 ST_GetStrReg

지정된 장치의 문자열(String) 레지스터 값을 조회합니다.

```

CVS_IMPORT_EXPORT CVS_ERROR WINAPI ST_GetStrReg(

```

```
int32_t hDevice,
const char* NodeName,
char* pInfo,
uint32_t* pSize
);
```

- 매개변수:
 - hDevice (IN): 장치 핸들
 - NodeName (IN): 조회할 레지스터의 노드 이름
 - pInfo (OUT): 조회된 문자열 값을 저장할 버퍼
 - pSize (IN/OUT): 버퍼 크기. 함수 호출 후 실제 크기로 업데이트됨
- 반환값:
 - MCAM_ERR_OK (성공)
 - 오류 코드 (실패)
- 예제 코드:

```
char deviceID[128];

uint32_t size = sizeof(deviceID);

CVS_ERROR result = ST_GetStrReg(hDevice, "DeviceUserID", deviceID, &size);

if (result != MCAM_ERR_OK)
{
    printf("Failed to get string register: %d\n", result);
}
else
{
    printf("Device User ID: %s\n", deviceID);
}
```

5.10.9 ST_SetCmdReg

지정된 장치의 명령(Command) 레지스터를 실행합니다. 예를 들어, 소프트웨어 트리거(Software Trigger)와 같은 명령을 실행하는 데 사용할 수 있습니다.

```
CVS_IMPORT_EXPORT CVS_ERROR WINAPI ST_SetCmdReg(
    int32_t hDevice,
    const char* NodeName
);
```

- 매개변수:
 - hDevice (IN): 장치 핸들
 - NodeName (IN): 실행할 명령 레지스터의 노드 이름
- 반환값:
 - MCAM_ERR_OK (성공)
 - 오류 코드 (실패)
- 예제 코드:

```

CVS_ERROR result = ST_SetCmdReg(hDevice, "TriggerSoftware");

if (result != MCAM_ERR_OK)
{
    printf("Failed to execute command register: %d\n", result);
}
else
{
    printf("Successfully executed command register.\n");
}

```

5.11 레지스터 정보 조회

레지스터의 값 범위 및 상태 정보를 조회하는 기능을 제공합니다.

5.11.1 ST_GetIntRegRange

지정된 장치의 정수형(Integer) 레지스터의 값 범위를 조회합니다. 최소값, 최대값, 증가 단위를 반환합니다.

```

CVS_IMPORT_EXPORT CVS_ERROR WINAPI ST_GetIntRegRange(
    int32_t hDevice,
    const char* NodeName,
    int64_t* pMin,
    int64_t* pMax,
    int64_t* pInc
);

```

- 매개변수:
 - hDevice (IN): 장치 핸들
 - NodeName (IN): 조회할 레지스터의 노드 이름
 - pMin (OUT): 레지스터의 최소값을 저장할 변수
 - pMax (OUT): 레지스터의 최대값을 저장할 변수
 - pInc (OUT): 레지스터의 증가 단위를 저장할 변수
- 반환값:
 - MCAM_ERR_OK (성공)
 - 오류 코드 (실패)
- 예제 코드:

```

int64_t min, max, inc;

CVS_ERROR result = ST_GetIntRegRange(hDevice, "ExposureTime", &min, &max,
&inc);

if (result != MCAM_ERR_OK)
{
    printf("Failed to get integer register range: %d\n", result);
}

```

```

else
{
    printf("ExposureTime Range: Min = %lld, Max = %lld, Increment = %lld\n", min,
max, inc);
}

```

5.11.2 ST_GetFloatRegRange

지정된 장치의 실수형(Float) 레지스터의 값 범위를 조회합니다. 최소값과 최대값을 반환합니다.

```

CVS_IMPORT_EXPORT CVS_ERROR WINAPI ST_GetFloatRegRange(
    int32_t hDevice,
    const char* NodeName,
    double* pMin,
    double* pMax
);

```

- 매개변수:
 - hDevice (IN): 장치 핸들
 - NodeName (IN): 조회할 레지스터의 노드 이름
 - pMin (OUT): 레지스터의 최소값을 저장할 변수
 - pMax (OUT): 레지스터의 최대값을 저장할 변수
- 반환값:
 - MCAM_ERR_OK (성공)
 - 오류 코드 (실패)
- 예제 코드:

```

double min, max;

CVS_ERROR result = ST_GetFloatRegRange(hDevice, "Gain", &min, &max);

if (result != MCAM_ERR_OK)
{
    printf("Failed to get float register range: %d\n", result);
}
else
{
    printf("Gain Range: Min = %.2f, Max = %.2f\n", min, max);
}

```

5.11.3 ST_GetEnumEntrySize

지정된 장치의 열거형(Enum) 레지스터에 존재하는 항목 개수를 조회합니다.

```

CVS_IMPORT_EXPORT CVS_ERROR WINAPI ST_GetEnumEntrySize(
    int32_t hDevice,
    const char* NodeName,

```

```
int32_t* pVal
);
```

- 매개변수:
 - hDevice (IN): 장치 핸들
 - NodeName (IN): 조회할 레지스터의 노드 이름
 - pVal (OUT): 열거형 항목 개수를 저장할 변수
- 반환값:
 - MCAM_ERR_OK (성공)
 - 오류 코드 (실패)
- 예제 코드:

```
int32_t enumSize;

CVS_ERROR result = ST_GetEnumEntrySize(hDevice, "PixelFormat", &enumSize);

if (result != MCAM_ERR_OK)
{
    printf("Failed to get enum entry size: %d\n", result);
}
else
{
    printf("PixelFormat Enum Size: %d\n", enumSize);
}
```

5.11.4 ST_GetEnumEntryIntValue

지정된 장치의 열거형(Enum) 레지스터에서 특정 인덱스의 항목 값을 정수(Integer)로 조회합니다.

```
CVS_IMPORT_EXPORT CVS_ERROR WINAPI ST_GetEnumEntryIntValue(
    int32_t hDevice,
    const char* NodeName,
    int32_t EntryIdx,
    int32_t* pVal
);
```

- 매개변수:
 - hDevice (IN): 장치 핸들
 - NodeName (IN): 조회할 레지스터의 노드 이름
 - EntryIdx (IN): 조회할 항목의 인덱스
 - pVal (OUT): 항목의 정수값을 저장할 변수
- 반환값:
 - MCAM_ERR_OK (성공)
 - 오류 코드 (실패)
- 예제 코드:

```
int32_t entryValue;

CVS_ERROR result = ST_GetEnumEntryIntValue(hDevice, "PixelFormat", 0,
```

```

&entryValue);

if (result != MCAM_ERR_OK)
{
    printf("Failed to get enum entry int value: %d\n", result);
}
else
{
    printf("PixelFormat[0] Enum Value: %d\n", entryValue);
}

```

5.11.5 ST_GetEnumEntryValue

지정된 장치의 열거형(Enum) 레지스터에서 특정 인덱스의 항목 값을 문자열(String)로 조회합니다.

```

CVS_IMPORT_EXPORT CVS_ERROR WINAPI ST_GetEnumEntryValue(
    int32_t hDevice,
    const char* NodeName,
    int32_t EntryIdx,
    char* pInfo,
    uint32_t* pSize
);

```

- 매개변수:
 - hDevice (IN): 장치 핸들
 - NodeName (IN): 조회할 레지스터의 노드 이름
 - EntryIdx (IN): 조회할 항목의 인덱스
 - pInfo (OUT): 조회된 항목 값을 저장할 문자열 버퍼
 - pSize (IN/OUT): 버퍼 크기. 함수 호출 후 실제 크기로 업데이트됨
- 반환값:
 - MCAM_ERR_OK (성공)
 - 오류 코드 (실패)
- 예제 코드:

```

char enumEntry[128];

uint32_t size = sizeof(enumEntry);

CVS_ERROR result = ST_GetEnumEntryValue(hDevice, "PixelFormat", 0, enumEntry,
&size);

if (result != MCAM_ERR_OK)
{
    printf("Failed to get enum entry value: %d\n", result);
}
else
{
    printf("PixelFormat[0] Enum Value: %s\n", enumEntry);
}

```

```
}
```

5.12 파라미터 읽기 및 쓰기

장치의 특정 메모리 주소에서 데이터를 읽거나 쓰는 기능을 제공합니다.

5.12.1 ST_ReadParam

지정된 장치에서 특정 주소(Address)와 길이(Length)의 파라미터 값을 읽어옵니다.

```
CVS_IMPORT_EXPORT CVS_ERROR WINAPI ST_ReadParam(
    int32_t hDevice,
    unsigned char* pBuffer,
    int64_t Address,
    int64_t Length
);
```

- 매개변수:
 - hDevice (IN): 장치 핸들
 - pBuffer (OUT): 읽은 파라미터 데이터를 저장할 버퍼
 - Address (IN): 읽을 파라미터의 시작 주소
 - Length (IN): 읽을 데이터의 길이(Byte 단위)
- 반환값:
 - MCAM_ERR_OK (성공)
 - 오류 코드 (실패)
- 예제 코드:

```
int64_t address = 0x1000; // Example address
int64_t length = 16;      // Read 16 bytes
unsigned char buffer[16];

CVS_ERROR result = ST_ReadParam(hDevice, buffer, address, length);

if (result != MCAM_ERR_OK)
{
    printf("Failed to read parameter: %d\n", result);
}
else
{
    printf("Successfully read parameter data: ");
    for (int i = 0; i < length; i++)
    {
        printf("%02X ", buffer[i]);
    }
    printf("\n");
}
```


5.12.2 ST_WriteParam

지정된 장치에 특정 주소(Address)와 길이(Length)의 파라미터 값을 기록합니다.

```
CVS_IMPORT_EXPORT CVS_ERROR WINAPI ST_WriteParam(
    int32_t hDevice,
    const unsigned char* pBuffer,
    int64_t Address,
    int64_t Length
);
```

- 매개변수:
 - hDevice (IN): 장치 핸들
 - pBuffer (IN): 기록할 파라미터 데이터가 저장된 버퍼
 - Address (IN): 기록할 파라미터의 시작 주소
 - Length (IN): 기록할 데이터의 길이(Byte 단위)
- 반환값:
 - MCAM_ERR_OK (성공)
 - 오류 코드 (실패)
- 예제 코드:

```
int64_t address = 0x1000; // Example address
int64_t length = 16;      // Write 16 bytes
unsigned char buffer[16] = {0xA1, 0xB2, 0xC3, 0xD4, 0xE5, 0xF6, 0x11, 0x22,
                             0x33, 0x44, 0x55, 0x66, 0x77, 0x88, 0x99, 0xAA};

CVS_ERROR result = ST_WriteParam(hDevice, buffer, address, length);

if (result != MCAM_ERR_OK)
{
    printf("Failed to write parameter: %d\n", result);
}
else
{
    printf("Successfully wrote parameter data.\n");
}
```

5.13 이벤트 및 콜백 관리

장치 이벤트 및 이미지 획득 이벤트 발생 시, 특정 콜백 함수를 등록하여 이를 처리할 수 있도록 합니다.

5.13.1 ST_RegisterGrabCallback

지정된 장치와 이벤트 ID에 대한 **Grab** 이벤트 콜백 함수를 등록합니다. 등록된 콜백 함수는 **Grab** 이벤트가 발생할 때 자동으로 호출됩니다.

```
CVS_IMPORT_EXPORT CVS_ERROR WINAPI ST_RegisterGrabCallback(
```

```

int32_t hDevice,
int32_t Event,
GrabCallback grabCallback,
void* UserData
);

```

- 매개변수:
 - hDevice (IN): 장치 핸들
 - Event (IN): Grab 이벤트 ID
 - grabCallback (IN): 등록할 콜백 함수
 - UserData (IN): 사용자 정의 데이터 (콜백 함수에서 참조 가능)
 -
- 반환값:
 - MCAM_ERR_OK (성공)
 - 오류 코드 (실패)
- 예제 코드:

```

void GrabEventHandler(int32_t eventID, const CVS_BUFFER* pBuffer, void*
pUserData)
{
    printf("Grab event occurred! Event ID: %d\n", eventID);
}

...

CVS_ERROR result = ST_RegisterGrabCallback(hDevice, EVENT_NEW_IMAGE,
GrabEventHandler, NULL);

if (result != MCAM_ERR_OK)
{
    printf("Failed to register grab callback: %d\n", result);
}
else
{
    printf("Successfully registered grab callback.\n");
}

```

5.13.2 ST_UnregisterGrabCallback

지정된 장치와 이벤트 ID에 대해 등록된 Grab 이벤트 콜백 함수를 해제합니다.

```

CVS_IMPORT_EXPORT CVS_ERROR WINAPI ST_UnregisterGrabCallback(
    int32_t hDevice,
    int32_t Event
);

```

- 매개변수:
 - hDevice (IN): 장치 핸들
 - Event (IN): Grab 이벤트 ID
- 반환값:

- MCAM_ERR_OK (성공)
- 오류 코드 (실패)
- 예제 코드:

```
CVS_ERROR result = ST_UnregisterGrabCallback(hDevice, EVENT_NEW_IMAGE);

if (result != MCAM_ERR_OK)
{
    printf("Failed to unregister grab callback: %d\n", result);
}
else
{
    printf("Successfully unregistered grab callback.\n");
}
```

5.13.3 ST_RegisterEventCallback

지정된 장치와 이벤트 ID에 대한 장치 이벤트 콜백 함수를 등록합니다. 등록된 콜백 함수는 특정 장치 이벤트가 발생할 때 자동으로 호출됩니다.

```
CVS_IMPORT_EXPORT CVS_ERROR WINAPI ST_RegisterEventCallback(
    int32_t hDevice,
    int32_t Event,
    EventCallback eventCallback,
    void* UserData
);
```

- 매개변수:
 - hDevice (IN): 장치 핸들
 - Event (IN): 장치 이벤트 ID
 - eventCallback (IN): 등록할 콜백 함수
 - UserData (IN): 사용자 정의 데이터 (콜백 함수에서 참조 가능)
- 반환값:
 - MCAM_ERR_OK (성공)
 - 오류 코드 (실패)
- 예제 코드:

```
void DeviceEventHandler(const CVS_EVENT* pEvent, void* pUserData)
{
    printf("Device event occurred! Event ID: %d\n", pEvent->eventID);
}

...

CVS_ERROR result = ST_RegisterEventCallback(hDevice, EVENT_EXPOSURE_END,
DeviceEventHandler, NULL);

if (result != MCAM_ERR_OK)
{
    printf("Failed to register device event callback: %d\n", result);
}
```

```

}
else
{
    printf("Successfully registered device event callback.\n");
}

```

5.13.4 ST_UnregisterEventCallback

지정된 장치와 이벤트 ID에 대해 등록된 장치 이벤트 콜백 함수를 해제합니다.

```

CVS_IMPORT_EXPORT CVS_ERROR WINAPI ST_UnregisterEventCallback(
    int32_t hDevice,
    int32_t Event
);

```

- 매개변수:
 - hDevice (IN): 장치 핸들
 - Event (IN): 장치 이벤트 ID
- 반환값:
 - MCAM_ERR_OK (성공)
 - 오류 코드 (실패)
- 예제 코드:

```

CVS_ERROR result = ST_UnregisterEventCallback(hDevice, EVENT_EXPOSURE_END);

if (result != MCAM_ERR_OK)
{
    printf("Failed to unregister device event callback: %d\n", result);
}
else
{
    printf("Successfully unregistered device event callback.\n");
}

```

5.14 XML 및 JSON 파일 관리

카메라 장치의 설정을 XML 또는 JSON 파일로 저장하고 불러오는 기능을 제공합니다.

5.14.1 ST_ImportXML

지정된 장치의 XML 설정 파일을 가져옵니다.

```

CVS_IMPORT_EXPORT CVS_ERROR WINAPI ST_ImportXML(
    int32_t hDevice,
    const char* FileName

```

```
);
```

- 매개변수:
 - hDevice (IN): 장치 핸들
 - FileName (IN): 가져올 XML 파일 경로
- 반환값:
 - MCAM_ERR_OK (성공)
 - 오류 코드 (실패)
- 예제 코드:

```
CVS_ERROR result = ST_ImportXML(hDevice, "config.xml");

if (result != MCAM_ERR_OK)
{
    printf("Failed to import XML: %d\n", result);
}
else
{
    printf("Successfully imported XML configuration.\n");
}
```

5.14.2 ST_ExportXML

현재 장치의 설정을 XML 파일로 내보냅니다.

```
CVS_IMPORT_EXPORT CVS_ERROR WINAPI ST_ExportXML(
    int32_t hDevice,
    const char* FileName,
    int Version = XML_VERSION_1_1,
    int Visibility = VISIBILITY_GURU
);
```

- 매개변수:
 - hDevice (IN): 장치 핸들
 - FileName (IN): 내보낼 XML 파일 경로
 - Version (IN, 선택): XML 버전 (기본값 XML_VERSION_1_1)
 - Visibility (IN, 선택): 내보낼 설정의 가시성 수준 (기본값 VISIBILITY_GURU)
- 반환값:
 - MCAM_ERR_OK (성공)
 - 오류 코드 (실패)
- 예제 코드:

```
CVS_ERROR result = ST_ExportXML(hDevice, "output_config.xml");

if (result != MCAM_ERR_OK)
{
    printf("Failed to export XML: %d\n", result);
}
else
```

```
{
    printf("Successfully exported XML configuration.\n");
}
```

5.14.3 ST_ImportJson

지정된 장치의 JSON 설정 파일을 가져옵니다.

```
CVS_IMPORT_EXPORT CVS_ERROR WINAPI ST_ImportJson(
    int32_t hDevice,
    const char* FileName
);
```

- 매개변수:
 - hDevice (IN): 장치 핸들
 - FileName (IN): 가져올 JSON 파일 경로
- 반환값:
 - MCAM_ERR_OK (성공)
 - 오류 코드 (실패)
- 예제 코드:

```
CVS_ERROR result = ST_ImportJson(hDevice, "config.json");

if (result != MCAM_ERR_OK)
{
    printf("Failed to import JSON: %d\n", result);
}
else
{
    printf("Successfully imported JSON configuration.\n");
}
```

5.14.4 ST_ExportJson

현재 장치의 설정을 JSON 파일로 내보냅니다.

```
CVS_IMPORT_EXPORT CVS_ERROR WINAPI ST_ExportJson(
    int32_t hDevice,
    const char* FileName,
    int Visibility = VISIBILITY_GURU
);
```

- 매개변수:
 - hDevice (IN): 장치 핸들
 - FileName (IN): 내보낼 JSON 파일 경로
 - Visibility (IN, 선택): 내보낼 설정의 가시성 수준 (기본값 VISIBILITY_GURU)
- 반환값:

- MCAM_ERR_OK (성공)
- 오류 코드 (실패)
- 예제 코드:

```
CVS_ERROR result = ST_ExportJson(hDevice, "output_config.json");

if (result != MCAM_ERR_OK)
{
    printf("Failed to export JSON: %d\n", result);
}
else
{
    printf("Successfully exported JSON configuration.\n");
}
```

6. 주의사항

실제 사용 시 **SDK**/라이브러리 버전에 따라 일부 함수 동작이나 파라미터가 상이할 수 있습니다. 네트워크 카메라(**GEV**)나 **USB3** 카메라(**U3V**) 등에 따라 지원되는 이벤트나 파라미터 범위가 다를 수 있습니다.

모든 함수의 반환값은 반드시 확인하여 예외 처리를 수행하는 것이 좋습니다.

콜백 함수 등록 시 중복 등록 또는 해제 여부에 유의해야 합니다.

7. 라이선스 및 저작권

본 라이브러리는 ©2000 - 2025 CREVIS CO., LTD.의 소유입니다.

무단 복제 및 재배포는 제한될 수 있으니 라이선스 정책을 준수하십시오.
