USB 3.0 Frame Grabber API Manual

Model Number:

개발자 API 매뉴얼

경기도 안양시 동안구 학의로 250 관양두산벤처다임 6층 605호
TEL: 82-31-596-6770
FAX: 82-31-596-6772
http://www.envylogic.com
기술지원 support@envylogic.com

1 응용 프로그램 인터페이스

개요

USB 3.0 버스 기반 EnvyLogic Frame Grabber의 편하고 직관적인 사용을 위해 제공하는 API들에 대한 사용 설명서 입니다.

API Function List

초기화 및 종료

DCU_Init

API 내부 객체를 생성, 초기화 하고, 디바이스 이벤트를 받을 수 있도록 윈도우를 생성한다.

int DCU_Init(LPVOID pCallbackContext,
DCUSB_CALLBACK_DEVICE_FUNCTION pfnDevcieEventFunction);

Parameters

pCallbackContext

Conetext 객체로 이벤트가 발생하여 *pfnDevcieEventFunction* 함수가 호출될 때 인수로 넘겨진다.

pfnDevcieEventFunction

디바이스 이벤트가 발생하면 호출할 Callback 함수

Return Value

객체 생성과 초기화가 성공하면 DCU_ERR_SUCCESS를, 실패한다면 DCU_ERR_UNSUCCESSFUL을 리턴한다.

Comment

객체들이 성공적으로 생성되고 초기화 되고 난 후, USB 버스에 Device가 연결되거나 연결이 끊기는 경우 그에 맞는 이벤트가 생성됩니다.
이벤트가 생성되면 인수로 받은 Callback 함수를 Context 인수와 함께 호출합니다.
아래는 Callback 함수의 Prototype 입니다.

*void (*DCUSB_CALLBACK_DEVICE_FUNCTION)(HANDLE hApiHandle,*

```
LPVOID pCallbackContext, DCUSB_DEVICE_STATUS nDeviceStatus);
```

또한 Callback 함수를 호출할 때는 생성된 이벤트를 나타내는 아래의 상수가 인수로 전달 됩니다.

```
enum DCUSB_DEVICE_STATUS
{
    DCU_DEVICE_PLUGOUT = 0,
    DCU_DEVICE_PLUGIN,
    DCU_DEVICE_CHANGED
};
```

DCU_End

기존에 할당된 객체들과 이벤트를 위한 윈도우를 해제하고 초기화 한다.

```
int DCU_End();
```

Parameters

Return Value

DCU_ERR_SUCCESS 값을 리턴

Comment

DCU_Init 함수에 의해 생성된 객체들을 모두 해제하고 이벤트를 받기 위한 윈도우를 파괴합니다. 그리고 전역 변수들을 초기화 합니다.

DCU_GetMaxDevices

현재 연결된 전체 디바이스 수를 얻습니다.

```
int   DCU_GetMaxDevices(int *pDeviceNum);
```

Parameters

pDeviceNum

현재 연결된 전체 디바이스 수가 담길 integer type pointer.

Return Value

호출이 성공하면 DCU_ERR_SUCCESS 값을 리턴. 만약 pDeviceNum 인수가 NULL이면 DCU_ERR_INVALID_PARAMETER를 리턴하고, 장치들을 관리할 객체가 할당되지 않았다면 DCU_ERR_NOT_INIT를 리턴.

Comment

장치들을 관리하는 객체로부터 USB 버스에 연결된 EnvyLogic USB Frame Grabber 수를 얻어옵니다.

DCU_GetDeviceInfo

인수로 주어진 디바이스 넘버에 해당하는 장치의 정보를 읽어온다.

int DCU_GetDeviceInfo(int iNumDevice, PDCU_DEVICE_INFO pDcDeviceInfo)

Parameters

iNumDevice

정보를 얻고자 하는 디바이스의 넘버

pDcDeviceInfo

정보를 얻어오기 위한 DCU_DEVICE_INFO 구조체

Return Value

정보를 얻는데 성공하면 DCU_ERR_SUCCESS를 리턴.
만약 iNumDevice의 넘버가 유효하지 않거나 구조체 포인터가 NULL이라면 DCU_ERR_INVALID_PARAMETER를 리턴.
CyUSBDevice 객체의 할당에 실패한다면 DCU_ERR_INSUFFICIENT_RESOURCE를 만약 iNumDevice에 해당하는 디바이스를 Open 하는데 실패한다면 DCU_ERR_DEVICE_CANT_OPEN을 리턴합니다.

Comment

인수의 구조체 정보는 아래와 같습니다.

```
typedef struct _DCU_DEVICE_INFO_TAG {
        ULONG       DriverVersion;
        ULONG       USBDIVersion;
        USHORT      BcdUSB;

        USHORT      VendorID;
        USHORT      ProductID;

        bool        bHighSpeed;
        bool        bSuperSpeed;

        char        DeviceName[DCU_STRING_MAXLENGTH];
        char        FriendlyName[DCU_STRING_MAXLENGTH];
        wchar_t     Manufacturer[DCU_STRING_MAXLENGTH];
        wchar_t     Product[DCU_STRING_MAXLENGTH];
        wchar_t     SerialNumber[DCU_STRING_MAXLENGTH];
} DCU_DEVICE_INFO, *PDCU_DEVICE_INFO;
```

DriverVersion

디바이스 드라이버의 버전 정보

USBDIVersion

BCD 포맷으로 된 USB Host Controller Driver 버전

BcdUSB

USB 디스크립터의 bcdUSB 값

VendorID

USB 디스크립터의 VendorID 값

ProductID

USB 디스크립터의 ProductID 값

bHighSpeed

이 디바이스가 High Speed 전송을 지원하는지 나타내는 값

bSuperSpeed

이 디바이스가 Super Speed 전송을 지원하는지 나타내는 값

DeviceName

USB 디스크립터의 iProduct 값

FriendlyName

드라이버의 inf 파일에 있는 디바이스를 나타내는 이름

Manufacturer

USB 디스크립터의 iManufacturer 값

Product

USB 디스크립터의 iProduct 값

SerialNumber

USB 디스크립터의 iSerialNumber 값

Capture 동작 제어

DCU_Open

인수로 주어진 넘버의 디바이스를 오픈하고 내부 객체들을 연결합니다. 오픈한 디바이스의 핸들을 리턴한다.

*int DCU_Open(int iNumDevice, HANDLE *pDeviceHandle);*

Parameters

iNumDevice

오픈 하고자 하는 디바이스의 넘버

pDeviceHandle

오픈 한 디바이스의 핸들을 얻어올 포인터

Return Value

장치를 성공적으로 Open 하면 DCU_ERR_SUCCESS를 리턴.
만약 iNumDevice의 값이 유효하지 않거나 pDeviceHandle의 포인터가 유효하지 않다면 DCU_ERR_INVALID_PARAMETER 리턴. 내부 객체가 생성되지 않았다면 DCU_ERR_DEVICE_NOT_INIT를 리턴하고 유효한 iNumDevice의 디바이스를 Open하는데 실패한다면 DCU_ERR_DEVICE_CANT_OPEN를 리턴한다.
디바이스를 Open하고 확인했는데 요구하는 디바이스와 구조가 다르다면 DCU_ERR_INVALIED_DEVICE를 리턴한다.

Comment

인수로 주어진 디바이스를 성공적으로 Open하면 디바이스의 Endpoint들을 확인하고 디바이스와 내부 객체들과 연결한다.

DCU_Close

인수로 주어진 핸들과 현재 열려진 디바이스가 맞는지 확인하고 맞으면 디바

이스를 Close하고 연결된 객체들을 초기화 한다.

int DCU_Close(HANDLE hDevice);

Parameters

hHandle

닫고자 하는 디바이스의 핸들

Return Value

장치를 성공적으로 Open 하면 DCU_ERR_SUCCESS를 리턴. 만약 hHandle의 값이 현재 열려진 디바이스의 핸들과 다르다면 DCU_ERR_INVALID_PARAMETER 를 리턴하고 내부 객체들이 생성되지 않은 상태면 DCU_ERR_DEVICE_NOT_INIT 을 리턴한다.

Comment

DCU_RunScript

특정 장치의 API 버전과 Device Driver의 버전 그리고 Chip의 버전 정보를 돌려준다.

*int DCU_RunScript(HANDLE hDevice, const char *lpszScriptFile);*

Parameters

hHandle

SCRIPT를 실행할 디바이스의 핸들

lpszScriptFile

실행할 SCRIPT 파일의 path

Return Value

SCRIPT를 성공적으로 실행하였다면 DCU_ERR_SUCCESS를 리턴 한다. 만약 SCRIPT 파일을 열지 못하였다면 DCU_ERR_SCRIPT_CANT_OPEN 리턴하고, SCRIPT 파일에 문법적인 오류가 있다면 DCU_ERR_SCRIPT_SYNTAX_ERROR를 리턴하고 이 경우는 DCU_GetScriptError 함수를 호출하여 에러가 발생한 line 번호와 에러가 발생한 문장을 받을 수 있다. SCRIPT를 실행하는 중에 발생하였다면 해당 에러 코드를 리턴한다.

Comment

DCU_ERR_SCRIPT_SYNTAX_ERROR가 리턴 되었을 경우 DCU_GetScriptError 함수를 호출하여 에러 정보를 받을 수 있다.

DCU_GetScripError

DCU_RunScript 중에 DCU_ERR_SCRIPT_SYNTAX_ERROR 에러가 발생하였을 경우 에러가 발생한 line 번호와 에러가 발생한 문장을 얻을 수 있다.

*int DCU_GetScriptError(HANDLE hDevice, int *pLine, char *pCause, int iLength);*

Parameters

hHandle

SCRIPT를 실행한 디바이스의 핸들

pLine

에러가 발생한 Line 정보를 얻어올 포인터 변수

pCause

에러가 발생한 문장을 얻어올 포인터 변수

pLine

pCause 버퍼의 길이

Return Value

에러 정보를 성공적으로 얻었다면 DCU_ERR_SUCCESS를 리턴 한다. 에러 정보를 얻는데 실패한다면 해당하는 에러 코드를 리턴한다.

Comment

DCU_GetVersion

특정 장치의 API 버전과 Device Driver의 버전 그리고 Chip의 버전 정보를 돌려준다.

*int DCU_GetVersion (HANDLE hDevice, DWORD *pDllVer, DWORD *pDrvVer, DWORD *pChipVer);*

Parameters

hHandle

버전 정보를 얻고자 하는 디바이스의 핸들

pDllVer

API 버전 정보가 들어갈 DWORD형 포인터

pDrvVer

디바이스 드라이버의 버전 정보가 들어갈 DWORD형 포인터

pChipVer

디바이스에 장착된 FPGA의 버전 정보가 들어갈 DWORD형 포인터

Return Value

버전 정보를 성공적으로 얻어오면 DCU_ERR_SUCCESS를 리턴. 인수의 값이 NULL이면 DCU_ERR_INVALID_PARAMETER를 리턴하고, 유효한 핸들 값이 아니면 DCU_ERR_INVALID_HANDLE를 리턴, FPGA로 부터 버전 정보를 읽어드리는데 실패하면 해당 에러 코드를 리턴한다.

Comment

버전 정보는 해당 디바이스를 Open한 이후 호출 할 수 있다.

DCU_Start

인수로 주어진 핸들이 유효한지 확인하고 맞으면 FRAME 정보를 설정하고 FPGA를 실행시킨다. 그리고 FRAME을 계속 읽어오는 작업 Thread를 실행 시킨다.

int DCU_Start(HANDLE hDevice);

Parameters

hHandle

시작하고자 하는 디바이스의 핸들

Return Value

만약 hHandle의 값이 유효하지 않으면 DCU_ERR_INVALID_HANDLE을 리턴하고 내부 객체들이 생성되지 않은 상태면 DCU_ERR_DEVICE_NOT_INIT 을 리턴한다. FPGA를 성공적으로 동작시키면 DCU_ERR_SUCCESS를 리턴.

Comment

DCU_Stop

인수로 주어진 핸들이 유효한지 확인하고 맞으면 핸들에 해당하는 디바이스의 FPGA의 동작을 멈추고, FRAME을 읽어오는 작업 Thread를 중단 시킨다.

int DCU_Stop(HANDLE hDevice);

Parameters

hHandle

멈추고자 하는 디바이스의 핸들

Return Value

FPGA의 동작을 성공적으로 멈추면 DCU_ERR_SUCCESS를 리턴. 만약 hHandle의 값이 NULL이면 DCU_ERR_INVALID_PARAMETER를 리턴하고, 유효하지 않은 핸들이라면 DCU_ERR_INVALID_HANDLE를 리턴한다.

Comment

DCU_GetFrame

한 FRAME의 데이터를 가져온다.

*int _stdcall DCU_GetFrame(HANDLE hDevice, int iBlocked, int *pCount, PBYTE pBuf, int *pLen);*

Parameters

hDevice

FRAME 데이터를 읽어올 디바이스의 핸들

iBlocked

값이 1이면 프레임을 읽을 때까지 기다림. 값이 0이면 무조건 결과를 가지고 리턴.

pCount

프레임을 읽었을 때 Frame Buffer queue에 저장된 FRAME 수를 기록할 int type 포인터

pBuf

FRAME 데이터를 읽어드릴 버퍼의 포인터

pLen

FRAME 데이터의 길이를 기록할 int type 포인터

Return Value

FRAME 데이터를 성공적으로 읽어오면 DCU_ERR_SUCCESS를 리턴. 만약 hDevice의 값이 NULL이면 DCU_ERR_INVALID_PARAMETER 를 리턴하고, 유효하지 않은 값이면 DCU_ERR_INVALID_HANDLE을 리턴한다.
Frame Buffer queue가 생성되지 않았다면 DCU_ERR_FRMGR_NOT_CREATE를 리턴하고, blocked가 0이고 Frame Buffer queue가 비었다면 DCU_ERR_DEVICE_FRAME_EMPTY를 리턴한다.

Comment

DCU_CancelFrame

Block되어 있는 DCU_GetFrame 함수의 Block을 해제하고 취소 시킨다.

int _stdcall DCU_CancelFrame(HANDLE hDevice);

Parameters

hDevice

취소시킬 디바이스의 핸들

Return Value

DCU_GetFrame을 정상적으로 취소하였으면 DCU_ERR_SUCCESS를 리턴. 만약 hDevice의 값이 NULL이면 DCU_ERR_INVALID_PARAMETER를 리턴 하고, 유효하지 않은 값이면 DCU_ERR_INVALID_HANDLE을 리턴 한다.

Comment

DCU_GetFrameAsync

비동기적으로 Frame을 가져오도록 Callback 함수를 설정하는 함수. Callback 설정되면 Frame이 들어올 때 마다 Callback 함수가 호출된다.

int _stdcall DCU_GetFrameAsync(HANDLE hDevice, DCUSB_FRAME_CALLBACK_FUNCTION pfnFrameCallback, LPVOID pCallbackContext);

Parameters

hDevice

비동기로 FRAME 데이터를 받을 디바이스의 핸들

pfnFrameCallback

Frame이 들어올 때 마다 호출될 Callback 함수. 만약 NULL이면 Callback 을 해제한다.

pCallbackContext

Callback이 호출될 때마다 같이 전달될 Context 객체

Return Value

DCU_GetFrameAsync에서 Callback을 정상적으로 설정되었다면 DCU_ERR_SUCCESS를 리턴. 만약 hDevice의 값이 NULL이면 DCU_ERR_INVALID_PARAMETER를 리턴 하고, 유효하지 않은 값이면 DCU_ERR_INVALID_HANDLE을 리턴 한다.

Comment

pfnFrameCallback 인수에 NULL을 넣고 호출하면 기존에 설정된 Callback을 해제한다.

Capture 파라메터 제어 함수

DCU_SetCapInfo

Capture Controller의 동작 방식을 지정하는 함수이다.

int DCU_SetCapInfo(HANDLE hDevice, int iSource, int iWidth, int iHeight, int iBpp, int iSkip, int iFormat);

Parameters

hDevice

설정하고자 하는 Device의 핸들

iSource

Frame 입력 소스를 설정
0: TEST PATTERN
1: MIPI
2: Paralle

iWidth

Frame의 폭

iHeight

Frame의 높이

iBpp

데이터 크기를 계산하기 위한 bits per pixel 값

iSkip

입력되는 Frame에서 얼마를 건너뛸지 설정

iFormat

Frame의 Format정보를 세트.

0: YUV16

1: RGB16

2: RAW10

Return Value

정성적으로 동작을 지정하면 DCU_ERR_SUCCESS를 리턴 한다.
만약 객체가 생성되지 않은 상태면 DCU_ERR_DEVICE_NOT_INIT 를 리턴하고, 주어진 핸들이 디바이스의 핸들과 다르다면 DCU_ERR_INVALID_HANDLE 를 리턴한다. 만약 장치가 동작중이면 설정을 변경하지 않고 DCU_ERR_DEVICE_XFERING 에러를 리턴한다.

Comment

TBD

DCU_GetCapInfo

Capture Controller의 설정된 값을 얻어오는 함수.

*int DCU_SetCapInfo(HANDLE hDevice, int *pSource, int *pWidth, int *pHeight, int *pBpp, int *pSkip, int *pFormat);*

Parameters

hDevice

설정 값을 얻어올 Device의 핸들

pSource

설정된 Frame 입력 소스

pWidth

설정된 Frame의 폭

pHeight

설정된 Frame의 높이

pBpp

설정된 bits per pixel 값

iSkip

설정된 skip 값

iFormat

설정된 Format 값

Return Value

정성적으로 설정 값을 얻어오면 DCU_ERR_SUCCESS를 리턴한다.
만약 객체가 생성되지 않은 상태면 DCU_ERR_DEVICE_NOT_INIT 를 리턴하고, 주어진 핸들이 디바이스의 핸들과 다르다면 DCU_ERR_INVALID_HANDLE 를 리턴한다.

Comment

그 외

리턴코드

함수 리턴 값의 의미들을 정리

DCU_ERR_SUCCESS

어떠한 에러없이 성공적으로 호출되었을경우

DCU_ERR_NOT_INIT

API가 초기화 되지 않았음.

DCU_ERR_ALREADY_INIT

이미 초기화되어 있는데 또 DCU_Init를호출

DCU_ERR_WINDOW_CANT_CREATE

이벤트를 받을 자체윈도우를 생성하는 도중 실패하는 경우

DCU_ERR_INVALID_PARAMETER

함수를 호출할 때 유효하지 않은 인수를 이용하여 호출

DCU_ERR_UNSUCCESSFUL

함수의 호출이 실패하는 경우

DCU_ERR_INSUFFICIENT_RESOURCE

메모리나 리소스를 할당하는데 실패

DCU_ERR_INVALID_HANDLE

인수의 핸들이 유효하지 않음

DCU_ERR_DEVICE_CANT_OPEN

디바이스가 오픈하지 못 하였음.

DCU_ERR_DEVICE_INVALID

디바이스는 오픈 되었지만 유효하지 않은 디바이스.

DCU_ERR_DEVICE_CANT_THREAD

Thread를 생성하는데 실패함

DCU_ERR_DEVICE_INVALID_FRAME_BUFFER

유효하지 않은 Frame Buffer

DCU_ERR_DEVICE_FRAME_NOT_READY

Frame Buffer Queue에 프레임이 준비되지 않았음.

DCU_ERR_DEVICE_NOT_CAP_INFO

DCU_SetCapInfo 함수가 호출되지 않았는데 Start가 실행됨.

DCU_ERR_DEVICE_CANT_CREATE_EVENT

동기화를 위한 이벤트 객체를 생성하는데 실패함

DCU_ERR_DEVICE_FRAME_TIME_OUT

Block 모드에서 Frame을 읽어오는데 Time Out이 발생

DCU_ERR_DEVICE_FRAME_EMPTY

Frame Buffer Queue가 비어있음

DCU_ERR_DEVICE_XFERING

이미 동작중인 디바이스에 DCU_SetCapInfo 호출

DCU_ERR_BULKBASE_WRITE_WAIT_XFER_FAIL

USB Bulk Endpoint로 데이터를 쓰고 기다리는 동안 에러 발생

DCU_ERR_BULKBASE_WRITE_FINISH_XFER_FAIL

USB Bulk Endpoint로 데이터를 쓴 후 Finish 동작 중 에러 발생

DCU_ERR_BULKBASE_READ_WAIT_XFER_FAIL

USB Bulk Endpoint로 데이터를 읽고 기다리는 동안 에러 발생

DCU_ERR_BULKBASE_READ_FINISH_XFER_FAIL

USB Bulk Endpoint로 데이터를 읽고 Finish 동작 중 에러 발생

DCU_ERR_CTRLBASE_WRITE_FAIL

USB Control Endpoint로 데이터를 쓰는 도중 에러 발생

DCU_ERR_CTRLBASE_READ_FAIL

USB Control Endpoint에서 데이터를 읽는 도중 에러 발생

DCU_ERR_CTRLBASE_NOT_CTRL_EP

USB Control Endpoint가 NULL

DCU_ERR_SERIALBASE_NOT_REGISTER

Serial I/O를 하기위한 Register 객체가 NULL

DCU_ERR_SREAILBASE_WAIT_TIME_OUT

Serial I/O를 전송시 Time Out 발생

DCU_ERR_FRMGR_BUFFER_EMPTY

Frame Manager에 Frame 버퍼가 비어있음.

DCU_ERR_FRMGR_BUFFER_OVERRUN

버퍼 갯수를 초과하여 기록됨.

DCU_ERR_FRMGR_NOT_CREATE

Frame 버퍼가 생성되지 않았음.

DCU_ERR_FRMGR_CANT_CREATE_EVENT

이벤트를 생성하는데 에러 발생

DCU_ERR_FRAME_NOT_CREATE

버퍼를 생성하지 않았음.

DCU_ERR_FRAME_NOT_ENOUGH_MEMORY

할당된 메모리보다 큰 데이터를 저장하려고 시도함.

DCU_ERR_SCRIPT_CANT_OPEN

스크립트 파일을 오픈하는데 실패

DCU_ERR_SCRIPT_SYNTAX_ERROR

스크립트 파일에 문법에 맞지 않는 문장이 있음.

DCU_ERR_SCRIPT_EXECUTE

스크립트를 실행하는데 에러 발생

DCU_ERR_SCRIPT_NOT_ERROR

에러가 발생하지 않았는데 DCU_GetLastError 발생