

Sunlogics USBScope SDK

Programmer's Reference v1.1

1. USB Functions

int USB_RegisterDevice()

USBScope 장치를 등록한다. USBScope를 사용하기 전에 먼저 장치 등록을 해 줘야 한다. 약 3초 동안의 대기시간이 필요하기 때문에 다음 작업은 SetTimer 윈도우 API를 이용해 타이머 메시지에서 처리해야 한다.

return 등록할 장치의 개수

0 : 등록할 장치가 없음. 즉, 장치가 없거나 이미 등록이 완료 됨

int USB_FindDevice()

등록 된 USBScope 장치의 개수를 얻어온다.

return 등록 된 장치의 개수

int USB_Detect(int index)

등록 된 장치가 USB 포트에 연결되어 있는지 여부를 알려준다.

parameters

index : 장치의 인덱스 값. 기본값은 0

return

2 : 장치가 연결되어 있고 바로 open 가능한 상태임

1 : 장치가 연결되어 있음

-1 : 장치가 연결되어 있지 않음

int USB_Open(HWND hWnd, int index)

등록 된 USBScope 장치를 연다.

parameters

hWnd : 어플리케이션의 메인 창의 핸들

index : 열려는 장치의 인덱스 값. 기본값은 0

return

- 1 : 연결됨
- 1 : 장치를 열 수 없음
- 2 : 라이선스 파일이 없음

void USB_Close()

USBScope 장치를 닫는다.

어플리케이션에서 USBScope 사용을 모두 마친 경우에 호출한다.

어플리케이션이 종료될 때는 내부적으로 자동 호출된다.

BOOL USB_IsOpen()

장치가 열려있고 사용 가능한지 여부를 알려준다.

return

TRUE : 장치가 열려 있음

FALSE : 장치가 열려있지 않음

BOOL USB_IsHighSpeed()

장치가 USB 2.0 포트에 연결되어 있는지 여부를 알려 준다. USB 2.0 High Speed 480 MBps 규격이 아니면 USBScope를 정상적으로 사용 할 수 없다.

return

TRUE : USB 2.0 규격의 480 Mbps 속도로 연결되어 있음

FALSE : USB 1.1 규격의 12 Mbps 속도로 연결되어 있음

HANDLE USB_GetHandle()

열려있는 장치의 윈도우 장치 핸들을 얻어 온다.

return USB 장치의 윈도우 장치 핸들

BOOL USB_ADChannel2_Enable(BOOL bADC2Enable)

아날로그 채널2를 켜거나 끈다. 아날로그 채널1은 항상 켜져있는 상태이다. 오직 채널2만 켜거나 끌 수 있다. 그리고 한 채널은 8비트로 구성되어 있다. 따라서 로직아날라이저 모드에서는 아날로그 한 채널은 디지털 8채널과 그 데이터 폭이 같다.

오실로스코프 모드에서 채널1만 필요한 경우에 이 함수를 이용해 채널2를 끌 수 있다. 채널2를 끄게 되면 채널1 데이터만 캡처하므로 필요한 메모리가 줄어들거나 동일한 메모리에서 두 배로 긴시간 동안 캡처할 수 있다. 그리고 20 MHz 이상의 주파수를 지원하는 제품에서 좀더 정확한 타이밍에 데이터를 캡처할 수 있다.

단, 이 함수는 아날로그 2채널을 지원하는 제품에서만 동작한다.

parameters

TRUE : 전 채널을 켜다.

FALSE : 아날로그 채널2 또는 디지털 채널 9~16을 끈다.

return

TRUE : 성공

FALSE : 실패

BOOL USB_IsADChannel2_Enable()

아날로그 채널2의 온 오프 상태를 얻어 온다.

return

TRUE : 켜져 있음

FALSE : 꺼져 있음

BOOL USB_Read(BYTE *readBuf, DWORD length)

USB를 통해 데이터를 캡처한다. 필요한 채널수에 따라 USB_ADChannel2_Enable 함수로 아날로그 채널2를 켜거나 끄고, UTIL_GetRequiredBytes 함수를 이용해 원하는 시간 동안의 데이터 캡처에 필요한 메모리 크기를 계산할 수 있다.

parameters

readBuf : 데이터를 캡처할 메모리 포인터.

length : 메모리의 크기

return

TRUE : 데이터 캡처 성공

FALSE : 실패

**BOOL USB_ReadLoop(DWORD length, BYTE *pBufA, BYTE *pBufB,
HANDLE hEvtA, HANDLE hEvtB, BOOL *pbStopFlag)**

USB를 통해 데이터를 캡처하는데 이중 버퍼를 사용한다. 버퍼A를 모두 채우면 이벤트A가 SET 되고 버퍼B에 채우는 작업을 진행한다. 버퍼B가 모두 차면 이벤트B를 SET하고 다시 버퍼A를 대상으로 진행된다. 스레드를 만들어 스레드 내에서 호출하도록 한다.

parameters

length : 메모리의 크기

pBufA : 메모리 포인터 첫 번째 메모리 포인터

pBufB : 데이터를 캡처할 두 번째 메모리 포인터

hEvtA : 이벤트 핸들

hEvtB : 이벤트 핸들

pbStopFlag : 동작 정지 플래그

return

TRUE : 데이터 캡처 성공

FALSE : 실패

char* USB_GetErrorString()

USB_Read에서 에러가 발생한 경우에 에러 문자열을 얻어 온다.

return 에러 문자열의 포인터

double USB_HW_Frequency()

USBScope의 하드웨어적 샘플링 주파수를 얻어 온다.

return 샘플링 주파수 (12000000.0, 20000000.0 또는 40000000.0)

int USB_HW_AnalogChannels()

USBScope의 하드웨어적 아날로그 채널수를 얻어 온다.

return 아날로그 채널수. (1 또는 2)

int USB_HW_DigitalChannels()

USBScope의 하드웨어적 디지털 채널수를 얻어 온다.

return 디지털 채널수 (8 또는 16)

char* USB_ModelName()

USBScope의 모델명을 얻어 온다.

return 모델명 문자열 (“USBScope S102” 등)

void USB_Change_HW_Frequency(double frequency)

USBScope의 하드웨어를 직접 변경하여 오실레이터 부품을 교체한 경우에 사용할 수 있다.

교체한 오실레이터 주파수를 입력하여 내부 계산을 정확하게 하도록 한다.

단, S102 모델에서는 지원하지 않는다.

parameters

frequency : 새로운 샘플링 주파수

int USB_Active_AnalogChannels()

채널2의 온 오프 상태를 고려해 현재 활성화된 아날로그 채널수를 얻어 온다.

return 활성화된 아날로그 채널 수

2. Utility Functions

double UTIL_GetAvailableSampleTime(DWORD buffer_size, int analog_channels, double capture_frequency)

지정한 메모리 크기에 따라 캡처할 수 있는 최대 시간을 계산해 준다.

parameters

buffer_size : 메모리 크기

analog_channels : 실제 켜져있는 아날로그 채널 수. 채널2를 켜고 끌 수 있으므로 채널2의 상태를 고려하여 값을 입력해야 한다. USB_Active_AnalogChannels 함수를 사용할 수 있다.

capture_frequency : 샘플링 주파수

return 캡처가 가능한 최대 시간. 단위는 micro second 이다.

DWORD UTIL_GetRequiredBytes(double capture_time, int analog_channels, double capture_frequency)

지정된 시간 동안 캡처하는데 필요한 메모리의 크기를 계산해 준다.

parameters

capture_time : 캡처 시간. 단위는 micro second 이다.

analog_channels : 실제 켜져있는 아날로그 채널 수. 채널2를 켜고 끌 수 있으므로 채널2의 상태를 고려하여 값을 입력해야 한다. USB_Active_AnalogChannels 함수를 사용할 수 있다.

capture_frequency : 샘플링 주파수

return 필요한 메모리 크기

void UTIL_Set_SW_Frequency(double frequency)

UTIL_DownSampling 함수에 사용할 샘플링 주파수를 지정한다. 만약 하드웨어 샘플링 주파수의 절반으로 다운 샘플링한 다면, nDownRate는 2가 되고 이 때의 주파수는 다음과 같이 계산할 수 있다.

$$\text{frequency} = \text{USB_HW_Frequency}() / \text{nDownRate};$$

parameters

frequency : 소프트웨어적 샘플링 주파수

void UTIL_DownSampling(BYTE *buffer, DWORD buflen, int analog_channels, int nDownRate)

데이터 처리량을 줄이기 위해 버퍼에 위치한 데이터를 프로그램적으로 다운 샘플링할 경우에 사용한다. 다운 샘플링한 결과는 버퍼에 다시 저장된다. 결과의 데이터 크기는 줄어들어

buflen / nDownRate 가 된다.

parameters

buffer : 데이터가 들어 있는 메모리 포인터

bufLen : 메모리 버퍼의 크기

analog_channels : 실제 켜져있는 아날로그 채널 수. 채널2를 켜고 끌 수 있으므로 채널2의 상태를 고려하여 값을 입력해야 한다. USB_Active_AnalogChannels 함수를 사용할 수 있다.

nDownRate : 다운 샘플링할 배율. (2, 3, 4, ...)

double UTIL_GetSampleTime(DWORD sampleIndex)

데이터의 인덱스에 따라 샘플링 시간을 계산해 준다. 아날로그 채널1 만을 캡처한 경우에는 샘플링 데이터의 인덱스와 메모리 버퍼의 인덱스가 동일하지만 채널2 까지 캡처한 경우에는 인덱스가 서로 일치하지 않으므로 주의해야 한다.

parameters

sampleIndex : 샘플링 데이터의 인덱스

return 샘플링 시간. 단위는 micro second 이다.

DWORD UTIL_GetFreeMemory()

사용 가능한 남아있는 물리적 메모리의 크기를 얻어 온다. 4GB 이상의 RAM이 장착된 컴퓨터의 경우에는 최대 4GB 까지 만 알려준다. 만약 이미 임시 메모리를 할당하고 있다면 실제 사용할 수 있는 메모리는 이 두가지의 합으로 계산해야 한다.

return 남아있는 물리적 메모리의 크기

3. Data File Functions

BOOL FILE_SaveScope(BYTE buffer, DWORD buffLen, int analog_channels, double frequency, char* path)

데이터를 오실로스코프용 데이터 파일 형식으로 저장한다.

parameters

buffer : 데이터 메모리 버퍼의 포인터

buffLen : 데이터 메모리의 크기

analog_channels : 실제 켜져있는 아날로그 채널 수. 채널2를 켜고 끌 수 있으므로 채널2의 상태를 고려하여 값을 입력해야 한다. USB_Active_AnalogChannels 함수를 사용할 수 있다.

frequency : 데이터의 샘플링 주파수

path : 저장할 파일이름. 확장자는 .DSO 나 .DSZ 만 가능하다. .DSZ는 압축파일이다.

return

TRUE : 성공

FALSE : 실패

DWORD FILE_LoadScope(BYTE *pBuf, char* path)

오실로스코프용 데이터 파일을 읽어 온다. pBuf 가 NULL 인 경우 헤더 부분을 제외한 실제 데이터의 크기를 돌려 준다. 따라서 NULL을 입력해 먼저 데이터의 크기를 알면 그에 따라 버퍼를 할당하고, 함수를 다시 호출해 그 버퍼의 포인터를 pBuf 인자로 주는 방법이 좋다.

parameters

pBuf : 저장할 메모리의 포인터. NULL인 경우는 데이터부의 크기를 돌려줌

path : 읽어 올 파일명

return 데이터 부분의 크기

DSO_HEADER* FILE_GetScopeHeader()

FILE_LoadScope 에서 읽은 파일의 헤더 부분을 가져온다.

return 헤더 부분의 포인터

BOOL FILE_SaveLogic(BYTE *buffer, DWORD buffLen, int analog_channels, double frequency, char* path)

데이터를 로직아날라이저용 파일 형식으로 저장한다.

parameters

buffer : 데이터 메모리 버퍼의 포인터

buffLen : 데이터 메모리의 크기

analog_channels : 실제 켜져있는 아날로그 채널 수. 채널2를 켜고 끌 수 있으므로 채널2의 상태를 고려하여 값을 입력해야 한다. USB_Active_AnalogChannels 함수를 사용할 수 있다.

frequency : 데이터의 샘플링 주파수

path : 저장할 파일이름. 확장자는 .LGA 나 .LGZ 만 가능하다. .LGZ는 압축파일이다.

return

TRUE : 성공

FALSE : 실패

DWORD FILE_LoadLogic(BYTE *pBuf, char* path)

로직아날라이저용 데이터 파일을 읽어온다. pBuf 가 NULL 인 경우 헤더 부분을 제외한 실제 데이터의 크기를 돌려 준다. 따라서 NULL을 입력해 먼저 데이터의 크기를 알면 그에 따

라 버퍼를 할당하고, 함수를 다시 호출해 그 버퍼의 포인터를 pBuf 인자로 주는 방법이 좋다.

parameters

pBuf : 저장할 메모리의 포인터. NULL인 경우는 데이터부의 크기를 돌려줌

path : 읽어 올 파일명

return 데이터 부분의 크기

LGA_HEADER* FILE_GetLogicHeader()

FILE_LoadLogic 함수에서 읽어온 파일의 헤더 부분을 가져온다.

return 헤더 부분의 포인터