SMARTX® Framework

프로그래밍 가이드

SmartX Framework Ver. 3.2.3

C++

업데이트 : 2021.10.14

Revision No.1(2020.07.09 기준)

※ C++의 경우 SmartX Framework의 일부 기능을 지원하며, 모든 기능을 사용할 수 있는 C#, VB의 사용을 권장합니다.

HNS www.hnsts.co.kr
(주)에이치앤에스
Hardware & Software Total Solution
Expert Embedded Total Solution

목 차

1. SmartGPIO

SmartX Framework는 기본적으로 .Net Compact Framework에서만 지원되도록 설계되어 있지만, 일부 기능을 C++에서 편리하게 사용할 수 있도록 외부 DLL 파일을 제공하고 있습니다. 따라서 SmartGPIO를 사용하기 위해서는 반드시 외부 DLL 파일을 실행 파일과 같이 IEC-Series에 배포해야 하며, "1. 인터페이스 함수 선언 → 2. 라이브러리 메모리 로드 처리 → 3. 인터페이스 함수 얻기" 과정을 거쳐야 합니다. 이때 IEC-Series 및 기능별로 로드해야 할 외부 DLL 파일이 다르므로 다음 내용을 확인하시기 바랍니다.

참조	SmartGPIO의 자세한 설명은 "홈페이지 → 자료실 → SmartX Framework 관련 → 프로그래밍 가이드 → SmartGPIO"를 참조하시기 바랍니다.

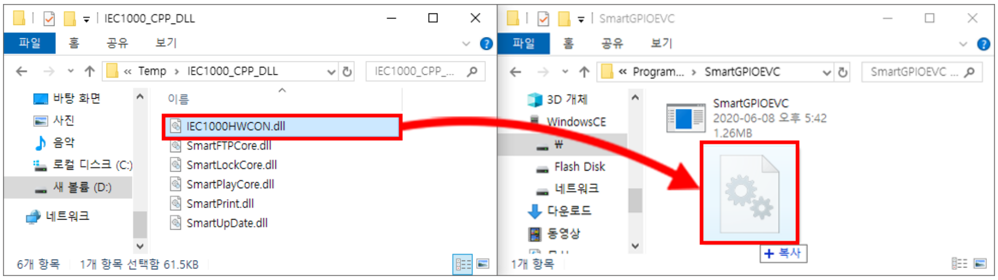
1) DLL 파일 및 복사 경로

HNS 홈페이지에서 사용하는 IEC-Series에 맞는 C++ For DLL 파일을 다운로드 받은 후 압축을 풉니다. 아래 표를 확인하여 DLL 파일을 IEC-Series에 실행 파일과 함께 복사합니다.

[표] IEC-Series별 DLL 파일과 다운로드 경로

IEC-Series	IEC266-Series	IEC667-Series	IEC1000-Series
DLL 파일	IEC266HWCON.dll	IEC667HWCON.dll	IEC1000HWCON.dll
다운로드 위치	홈페이지 → 자료실 → SmartX Framework 관련 → 5. SmartX C++ For DLL 파일		

DLL 파일을 IEC-Series에 복사합니다. DLL 파일을 사용하는 실행 파일이 저장된 폴더에 저장하는 것을 권장하며, 다른 경로에 저장하는 경우 "인터페이스 선언 및 얻기"의 참고 내용을 확인하시기 바랍니다.
※ 개발 PC에서 프로그램을 배포하는 경우 실행파일은 "Program File" 폴더에 배포됩니다.



2) 인터페이스 선언 및 얻기

인터페이스 사용을 위해 사용할 인터페이스의 선언 후 외부 DLL 파일을 메모리에 로드하여 인터페이스를 얻는 과정을 거쳐야 합니다. 이때 메모리 로드 처리를 위해 LoadLibrary() 함수 호출 시 상단 표의 DLL 파일의 이름을 인자로 설정합니다. 아래 코드는 예시이며, 필요로 하는 인터페이스 함수를 사용하기 위해서는 다음 내용을 참고하시기 바랍니다.

참고	DLL 파일 저장 경로에 따른 LoadLibrary() 함수 인자값

DLL 파일이 실행 파일과 같은 경로에 저장된 경우, LoadLibrary() 함수 인자값을 DLL 파일 이름만으로 설정해 호출할 수 있습니다. 만약 DLL 파일이 실행 파일과 다른 경로에 저장된 경우, LoadLibray() 함수의 인자값에는 저장된 DLL 파일의 전체 경로를 인자값으로 설정해야 합니다.

[C++] 인터페이스 얻기 및 사용하기 예시 코드

```
// 1. 인터페이스 함수 선언
typedef void (*SetPortData)(int iPortData);
SetPortData  m_pfSetPortADatas;
// 2. 라이브러리 메모리 로드 처리
HINSTANCE  m_hInst;
// SmartGPIO의 인터페이스를 사용하기 위해 IEC1000HWCON.dll 파일을 메모리에 로드
// IEC-Series 및 사용하는 기능별로 외부 DLL 파일이 다르며, 상단 표를 참고하시기 바랍니다.
m_hInst = LoadLibrary(L"IEC1000HWCON.dll");
// 외부 DLL 파일이 다른 경로(Flash Disk\\extdlls)에 저장된 경우
// m_hInst = LoadLibrary(L"Flash Disk\\extdlls\\IEC1000HWCON.dll");
if(m_hInst == NULL)
{
  // 메모리 로드 실패
  return FALSE;
}
// 3. 인터페이스 함수 얻기
m_pfSetPortADatas = (SetPortData)GetProcAddress(hInst, _T("SetPortAData"));
// 4. 함수 사용
m_pfSetPortADatas(0xFF);  // Port-A0~7(all) HIGH
```

참고 Wrapping Class 제공안내

사용할 인터페이스를 각각 선언 및 얻어오는 복잡한 과정을 거칠 필요 없이 한 번의 선언과 인스턴싱만으로 편리하게 SmartGPIO의 인터페이스를 사용할 수 있도록 Wrapping Class를 제공하고 있습니다.
자세한 내용은 "CSmartGPIOWrap Class 설명" 내용을 참고하시기 바랍니다.

3) C++ 인터페이스 설명

함수	SetPortABitCfg, SetPortBBitCfg, SetPortCBitCfg. SetPortDBitCfg, SetPortEBitCfg, SetPortFBitCfg, SetPortGBitCfg, SetPortHBitCfg

Port-A, B, C, D, E, F, G, H의 방향(입력/출력)을 Pin 단위로(1Bit) 설정합니다.

◉ `typedef void (*SetPortBitCfg)(int iBitIndex, int iBitCfg)`

[인자]

- `int iBitIndex` : Port에서 초기화 할 Pin 번호. 0번 Pin의 iBitIndex값은 0입니다. (0 ~ 7)
- `int iBitCfg` : Pin의 방향(입력/출력)을 설정합니다. (1 → 출력, 0 → 입력)

[표] Port-X의 각 Pin 방향설정 방법

Pin 번호	Pin 0	Pin 1	Pin 2	Pin 3	Pin 4	Pin 5	Pin 6	Pin 7
iBitIndex	0	1	2	3	4	5	6	7
iBitCfg	1	1	0	0	1	1	0	0
방향	출력	출력	입력	입력	출력	출력	입력	입력

C++ 사용법

```
typedef void (*SetPortBitCfg)(int iBitIndex, int iBitCfg);
SetPortBitCfg        m_pfSetPortABitCfg;
SetPortBitCfg        m_pfSetPortBBitCfg;
// -- 중략 --
HINSTANCE            hInst;
hInst = LoadLibrary(L"IEC1000HWCON.DLL");
if(hInst == NULL)
{
  return;
```

```
}
m_pfSetPortABitCfg = (SetPortBitCfg)GetProcAddress(hInst, _T("SetPortABitCfg"));
m_pfSetPortBBitCfg = (SetPortBitCfg)GetProcAddress(hInst, _T("SetPortBBitCfg"));
// -- 중략 --
m_pfSetPortABitCfg(0, 1);  // Port-A의 0번 핀 출력 모드 사용
m_pfSetPortABitCfg(3, 0);  // Port-A의 3번 핀 입력 모드 사용
m_pfSetPortBBitCfg(5, 0);  // Port-B의 5번 핀 입력 모드 사용
m_pfSetPortBBitCfg(7, 1);  // Port-B의 7번 핀 출력 모드 사용
FreeLibrary(hInst);
```

함수	SetPortACfg, SetPortBCfg, SetPortCCfg. SetPortDCfg, SetPortECfg, SetPortFCfg, SetPortGCfg, SetPortHCfg

Port-A, B, C, D, E, F ,G ,H의 방향(입력/출력)을 Port단위로(8Bit) 설정합니다.

각 Pin의 Bit 값이 1이면 출력(Output), Bit 값이 0이면 입력(Input)입니다.

◉ typedef bool (*SetPortCfg)(int iCon)

[인자]

• int iCon : Port의 방향(입력/출력)을 설정합니다.

[표] Port-X의 각 Pin 방향설정 방법

Pin 번호	Pin 0	Pin 1	Pin 2	Pin 3	Pin 4	Pin 5	Pin 6	Pin 7
Bit	1	1	0	0	1	1	0	0
방향	출력	출력	입력	입력	출력	출력	입력	입력
Hex	0x33							
Int	51							

[리턴값]

• bool : 모드 설정 성공 여부
  - true : 성공
  - false : 실패

C++ 사용법

```
typedef bool (*SetPortCfg)(int iCon);
SetPortCfg          m_pfSetPortACfgs;
// -- 중략 --
HINSTANCE hInst;
hInst = LoadLibrary(L"IEC1000HWCON.DLL");
if(hInst == NULL)
{
  return;
}
m_pfSetPortACfgs = (SetPortCfg)GetProcAddress(hInst, _T("SetPortACfg"));
// -- 중략 --
m_pfSetPortACfgs(0x00);  // Port-A 모든 핀을 입력 모드로 설정
FreeLibrary(hInst);
```

함수	SetPortABit, SetPortBBit, SetPortCBit. SetPortDBit, SetPortEBit, SetPortFBit, SetPortGBit, SetPortHBit

Port-A, B, C, D, E, F, G, H의 Pin 단위로(1Bit) 출력 상태를 제어합니다.

◉ typedef void (*SetPortBit)(int iBitIndex, int iBool)

[인자]

- `int iBitIndex` : Port에서 출력 상태를 제어할 Pin 번호. 0번 Pin의 iBitIndex값은 0입니다. (0 ~ 7)
- `int iBool` : Port의 출력 상태를 설정 (0 → LOW(0V), 1 → HIGH(3.3V))

[표] 인자값에 따른 Port-X의 각 Pin별 상태

Pin 번호	Pin 0	Pin 1	Pin 2	Pin 3	Pin 4	Pin 5	Pin 6	Pin 7
iBitIndex	0	1	2	3	4	5	6	7
iBool	1	1	0	0	1	1	0	0
출력 상태	HIGH	HIGH	LOW	LOW	HIGH	HIGH	LOW	LOW

C++ 사용법

```
typedef void (*SetPortBit)(int iBitIndex, int iBool);
SetPortBit          m_pfSetPortABitData;
// -- 중략 --
HINSTANCE           hInst;
hInst = LoadLibrary(L"IEC1000HWCON.DLL");
if(hInst == NULL)
{
  return;
}
m_pfSetPortABitData = (SetPortBit)GetProcAddress(hInst, _T("SetPortABit"));
// -- 중략 --
m_pfSetPortABitData(0, 1);  // Port-A0 HIGH
m_pfSetPortABitData(3, 0);  // Port-A3 LOW
m_pfSetPortABitData(7, 1);  // Port-A7 HIGH
FreeLibrary(hInst);
```

함수 SetPortAData, SetPortBData, SetPortCData. SetPortDData, SetPortEData, SetPortFData, SetPortGData, SetPortHData

Port-A, B, C, D, E, F, G, H를 Port 단위로(8Bit) 출력 상태를 제어합니다.

◉ `typedef void (*SetPortData)(int iPortData)`

[인자]

- `int iPortData` : 출력 상태 제어 설정 값 (0 → LOW(0V), 1 → HIGH(3.3V))

[표] Port-X의 각 Pin 방향설정 방법

Pin 번호	Pin 0	Pin 1	Pin 2	Pin 3	Pin 4	Pin 5	Pin 6	Pin 7
Bit	1	1	0	0	1	1	0	0
출력 상태	HIGH	HIGH	LOW	LOW	HIGH	HIGH	LOW	LOW
Hex	0x33							
Int	51							

C++ 사용법

```
typedef void (*SetPortData)(int iPortData);
SetPortData         m_pfSetPortADatas;
// -- 중략 --
HINSTANCE           hInst;
hInst = LoadLibrary(L"IEC1000HWCON.DLL");
if(hInst == NULL)
{
  return;
```

```
}
m_pfSetPortADatas = (SetPortData)GetProcAddress(hInst, _T("SetPortAData"));
// -- 중략 --
m_pfSetPortADatas(0xFF);  // Port-A0~7(all) HIGH
m_pfSetPortADatas(0x00);  // Port-A0~7(all) LOW
FreeLibrary(hInst);
```

함수	GetPortABit, GetPortBBit, GetPortCBit. GetPortDBit, GetPortEBit, GetPortFBit, GetPortGBit, GetPortHBit

Port-A, B, C, D, E, F, G, H의 Pin 단위로(1Bit) 입력 상태를 확인합니다.

◉ typedef unsigned char (*GetPortBit)(int iBitIndex)

[인자]

• int iBitIndex : Port에서 입력 상태를 확인할 Pin 번호. 0번 Pin의 iBitIndex값은 0입니다. (0 ~ 7)

[리턴값]

• unsigned char : Port의 특정 Pin의 입력 상태
  - 0 : LOW(0V)
  - 1 : HIGH(3.3V)

[표] 인자값에 따른 Port-X의 각 Pin별 상태

Pin 번호	Pin 0	Pin 1	Pin 2	Pin 3	Pin 4	Pin 5	Pin 6	Pin 7
iBitIndex	0	1	2	3	4	5	6	7
리턴값	1	1	0	0	1	1	0	0
입력 상태	HIGH	HIGH	LOW	LOW	HIGH	HIGH	LOW	LOW

C++ 사용법

```
typedef unsigned char (*GetPortBit)(int iBitIndex);
GetPortBit           m_pfGetPortABitData;
// -- 중략 --
HINSTANCE            hInst;
hInst = LoadLibrary(L"IEC1000HWCON.DLL");
if(hInst == NULL)
{
  return;
}
int iPortA3;
int iPortA7;
m_pfGetPortABitData = (GetPortBit)GetProcAddress(hInst, _T("GetPortABit"));
iPortA3 = m_pfGetPortABitData(3);  // Port-A3의 상태를 읽음
iPortA7 = m_pfGetPortABitData(7);  // Port-A7의 상태를 읽음
FreeLibrary(hInst);
```

함수	GetPortAData, GetPortBData, GetPortCData. GetPortDData, GetPortEData, GetPortFData, GetPortGData, GetPortHData

Port-A, B, C, D, E, F, G, H를 Port 단위로(8Bit) 입력 상태를 확인합니다.

◉ typedef unsigned char (*GetPortData)()

[리턴값]

• unsigned char : Port의 입력 상태

[표] Port-X의 각 Pin 방향설정 방법

Pin 번호	Pin 0	Pin 1	Pin 2	Pin 3	Pin 4	Pin 5	Pin 6	Pin 7
Bit	1	1	0	0	1	1	0	0
출력 상태	HIGH	HIGH	LOW	LOW	HIGH	HIGH	LOW	LOW
Hex	0x33							
Int	51							

C++ 사용법

```
typedef unsigned char (*GetPortData)();
GetPortData          m_pfGetPortADatas;
// -- 중략 --
HINSTANCE            hInst;
hInst = LoadLibrary(L"IEC1000HWCON.DLL");
if(hInst == NULL)
{
  return;
}
int iPortA;
m_pfGetPortADatas = (GetPortData)GetProcAddress(hInst, _T("GetPortAData"));
iPortB = GetPortADatas();  // Port-A의 상태를 읽음
FreeLibrary(hInst);
```

4) CSmartGPIOWrap Class 설명

SmartGPIOWrap Class는 사용할 인터페이스를 각각 선언 및 얻어오는 복잡한 과정을 거칠 필요 없이 한 번의 선언과 인스턴싱만으로 편리하게 SmartGPIO의 기능을 C++(C++, Visual Studio 2005, 2008)에서 지원하도록 Wrapping 한 Class입니다. 본 Class를 사용하기 위해서는 "SmartGPIOWrap.h" 파일과 "SmartGPIOWrap.cpp" 파일을 대상 프로젝트에 복사한 후 #include "SmartGPIOWrap.h"을 호출하여 사용하시기 바랍니다.

인터페이스 및 사용 방식은 위에서 설명한 C++ 인터페이스와는 약간의 차이가 있으므로 다음의 내용을 참고하시어 C++에서 적용하시기 바랍니다.

4-1) SmartGPIOWrap Class 사용하기

C++에서 SmartGPIOWrap Class를 사용하기 위해서는 아래와 STEP을 진행해야 합니다.

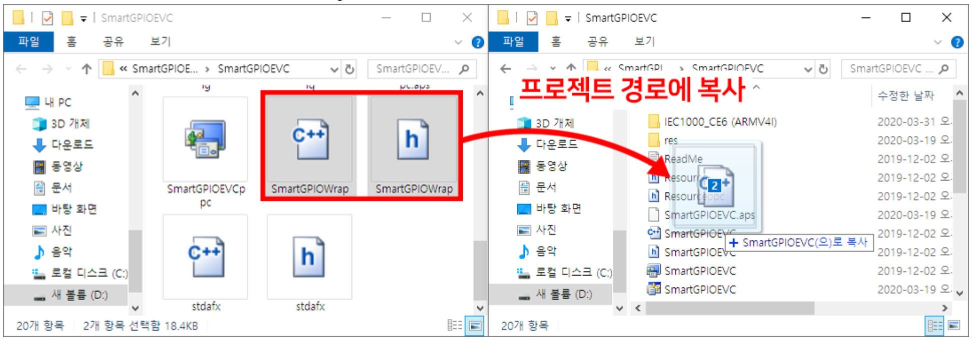
STEP-1	Wrap Class 파일 복사하기

HNS 홈페이지에서 사용하는 IEC-Series에 맞는 예제파일을 다운받은 후 압축을 풉니다.
"SmartGPIOEVC" 폴더에서 아래 표를 확인하여 Wrap Class 파일을 프로젝트 경로에 복사합니다.

[표] IEC-Series별 Wrap Class 파일

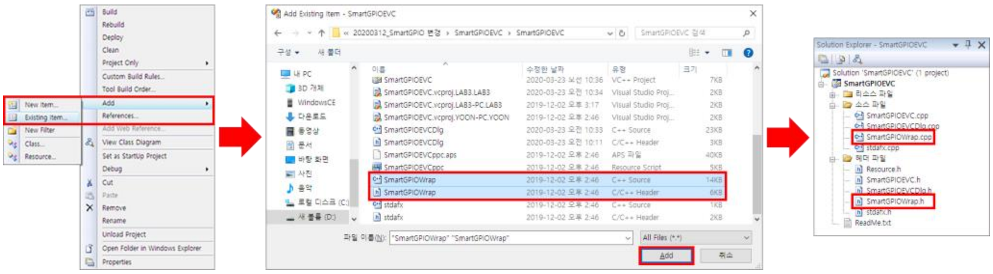
IEC-Series	IEC266-Series	IEC667-Series	IEC1000-Series
Wrap Class 파일	SmartGPIOWrap.h, SmartGPIOWrap.cpp		
다운로드 위치	홈페이지 → 자료실 → SmartX Framework 관련 → 6. SmartX Framework 예제파일 → Visual C++ 예제파일 → "SmartGPIOEVC"		

Wrap Class 파일을 프로젝트 경로에 복사합니다.



STEP-2 Wrap Class 추가하기

복사한 Wrap Class 파일을 프로젝트에 추가합니다.



STEP-3 Wrap Class 사용하기

사용할 소스 파일에 #include<SmartGPIOWrap.h> 추가한 후 선언 및 인스턴스하여 사용합니다.

```
#include "SmartGPIOWrap.h"
// SmartGPIO를 사용하기 위해 변수 선언 및 인스턴싱
CSmartGPIOWrap m_SmartGPIO;
// Port-A의 모든 Pin을 출력으로 설정
m_SmartGPIO.SetPortACfgs(0xFF);
// Port-B의 모든 Pin을 입력으로 설정
m_SmartGPIO.SetPortBCfgs(0x00);
// Port-A의 Pin0를 HIGH로 설정
m_SmartGPIO.SetPortABit(0, 1);
// PORT-B의 입력상태 확인
int iPortDatas = m_SmartGPIO.GetPortBData();
```

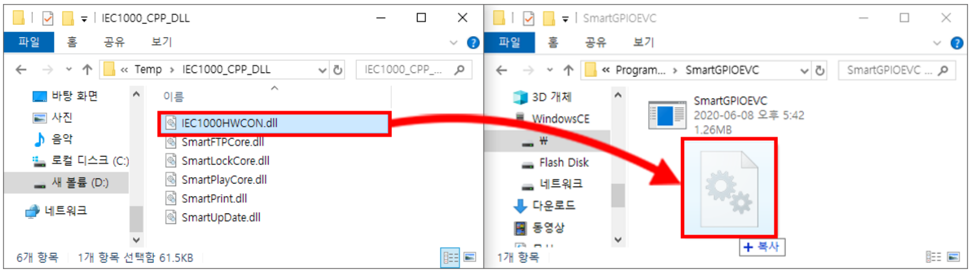
STEP-4 DLL 파일 복사하기

HNS 홈페이지에서 사용하는 IEC-Series에 맞는 예제파일을 다운받은 후 압축을 풉니다. "참조 DLL" 폴더에서 아래 표를 확인하여 사용하는 기능에 맞는 DLL 파일을 IEC-Series에 실행 파일과 함께 복사합니다.

[표] IEC-Series별 DLL 파일과 다운로드 경로

IEC-Series	IEC266-Series	IEC667-Series	IEC1000-Series
DLL 파일	IEC266HWCON.dll	IEC667HWCON.dll	IEC1000HWCON.dll
다운로드 위치	홈페이지 → 자료실 → SmartX Framework 관련 → 5. SmartX C++ For DLL 파일		

DLL 파일을 실행 파일 위치에 복사합니다.



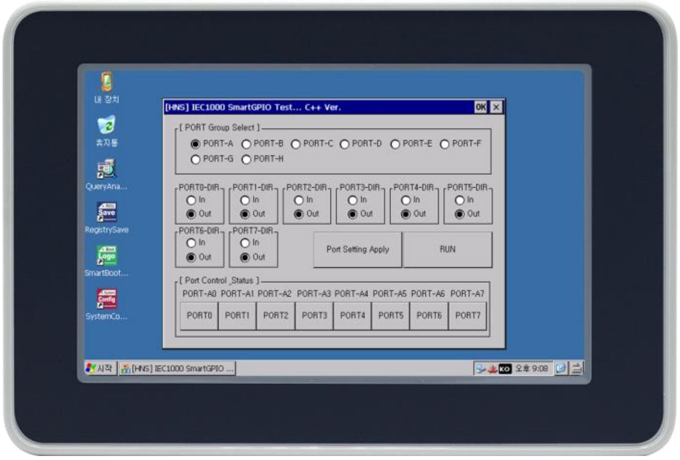
참고 자세한 DLL 파일의 복사 경로는 "DLL 파일 및 복사 경로" 내용을 참고하시기 바랍니다.

4-2) SmartGPIOWrap Class 예제 사용하기

SmartGPIOWrap Class를 활용한 예제를 제공하고 있으며, 예제파일은 다음 경로에서 다운로드하여 참고하시기 바랍니다.

[예제 파일 다운로드 위치]

홈페이지(www.hnsts.co.kr) → 자료실 → SmartX Framework 예제파일 다운로드 → “SmartGPIOEVC”



2. SmartADC

SmartX Framework는 기본적으로 .Net Compact Framework에서만 지원되도록 설계되어 있지만, 일부 기능을 C++에서 편리하게 사용할 수 있도록 외부 DLL 파일을 제공하고 있습니다. 따라서 SmartADC를 사용하기 위해서는 반드시 외부 DLL 파일을 실행 파일과 같이 IEC-Series에 배포해야 하며, "1. 인터페이스 함수 선언 → 2. 라이브러리 메모리 로드 처리 → 3. 인터페이스 함수 얻기" 과정을 거쳐야 합니다. 이때 IEC-Series 및 기능별로 로드해야 할 외부 DLL 파일이 다르므로 다음 내용을 확인하시기 바랍니다.

참조 SmartADC의 자세한 설명은 "홈페이지 → 자료실 → SmartX Framework 관련 → 프로그래밍 가이드 → SmartADC"를 참조하시기 바랍니다.

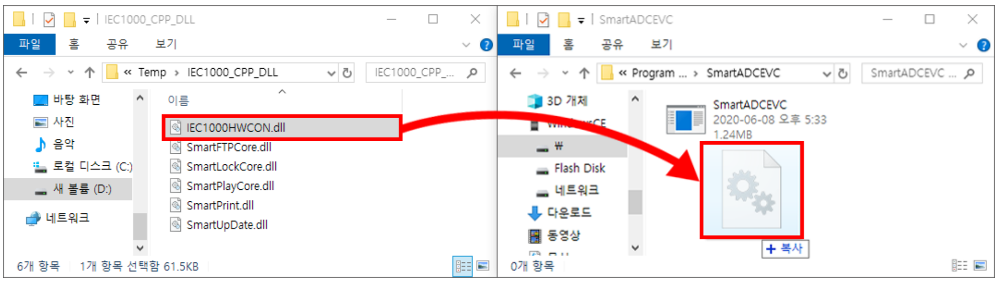
1) DLL 파일 및 복사 경로

HNS 홈페이지에서 사용하는 IEC-Series에 맞는 C++ For DLL 파일을 다운로드 받은 후 압축을 풉니다. 아래 표를 확인하여 DLL 파일을 IEC-Series에 실행파일과 함께 복사합니다.

[표] IEC-Series별 DLL 파일과 다운로드 경로

IEC-Series	IEC266-Series	IEC667-Series	IEC1000-Series
DLL 파일	IEC266HWCON.dll	IEC667HWCON.dll	IEC1000HWCON.dll
다운로드 위치	홈페이지 → 자료실 → SmartX Framework 관련 → 5. SmartX C++ For DLL 파일		

DLL 파일을 IEC-Series에 복사합니다. DLL 파일을 사용하는 실행 파일이 저장된 폴더에 저장하는 것을 권장하며, 다른 경로에 저장하는 경우 "인터페이스 선언 및 얻기"의 참고 내용을 확인하시기 바랍니다.
※ 개발 PC에서 프로그램을 배포하는 경우 실행 파일은 "Program File" 폴더에 배포됩니다.



2) 인터페이스 선언 및 얻기

인터페이스 사용을 위해 사용할 인터페이스의 선언 후 외부 DLL 파일을 메모리에 로드하여 인터페이스를 얻는 과정을 거쳐야 합니다. 이때 메모리 로드 처리를 위해 LoadLibrary() 함수 호출 시 상단 표의 DLL 파일의 이름을 인자로 설정합니다. 아래 코드는 예시이며, 필요로 하는 인터페이스 함수를 사용하기 위해서는 다음 내용을 참고하시기 바랍니다.

참고 DLL 파일 저장 경로에 따른 LoadLibrary() 함수 인자값

DLL 파일이 실행 파일과 같은 경로에 저장된 경우, LoadLibrary() 함수 인자값을 DLL 파일 이름만으로 설정해 호출할 수 있습니다. 만약 DLL 파일이 실행 파일과 다른 경로에 저장된 경우, LoadLibray() 함수의 인자값에는 저장된 DLL 파일의 전체 경로를 인자값으로 설정해야 합니다.

[C++] 인터페이스 얻기 및 사용하기 예시 코드

```
// 1. 인터페이스 함수 선언
typedef void (*SetADCPrescaleEx)(unsigned long lPrescale);
SetADCPrescaleEx        m_pfSetADCPrescale;
// 2. 라이브러리 메모리 로드 처리
HINSTANCE  m_hInst;
// SmartADC의 인터페이스를 사용하기 위해 IEC1000HWCON.dll 파일을 메모리에 로드
// IEC-Series 및 사용하는 기능별로 외부 DLL 파일이 다르며, 상단 표를 참고하시기 바랍니다.
m_hInst = LoadLibrary(L"IEC1000HWCON.dll");
// 외부 DLL 파일이 다른 경로(Flash Disk\\extdlls)에 저장된 경우
// m_hInst = LoadLibrary(L"Flash Disk\\extdlls\\IEC1000HWCON.dll");
if(m_hInst == NULL)
{
  // 메모리 로드 실패
  return FALSE;
}
// 3. 인터페이스 함수 얻기
m_pfSetADCPrescale = (SetADCPrescaleEx)GetProcAddress(m_hInst, _T("SetADCPrescale"));
// 4. 함수 사용
m_pfSetADCPrescale(49);
```

참고 Wrapping Class 제공안내

사용할 인터페이스를 각각 선언 및 얻어오는 복잡한 과정을 거칠 필요없이 한 번의 선언과 인스턴싱만으로 편리하게 SmartADC의 인터페이스를 사용할 수 있도록 Wrapping Class를 제공하고 있습니다.
자세한 내용은 “CSmartADCWrap Class 설명” 내용을 참고하시기 바랍니다.

3) C++ 인터페이스 설명

함수	SetADCPrescaleEx

A /D 변환 시간 관련 설정값입니다. (SetADCPrescaleEx() 함수의 설정 값은 49가 권장값입니다)

주의 SetADCPrescaleEx() 함수의 설정값은 49 이상으로 설정하시기 바랍니다.

◉ typedef void (*SetADCPrescaleEx)(unsigned long lPrescale)

[인자]

• unsigned long lPrescale : A/D 변환 시간 관련 설정 값

C++ 사용법

```
typedef void (*SetADCPrescaleEx)(unsigned long lPrescale);
SetADCPrescaleEx     m_pfSetADCPrescale;
HINSTANCE            m_hInst;
m_hInst = LoadLibrary(L"IEC1000HWCON.DLL");
if(m_hInst == NULL)
{
  return FALSE;
}
m_pfSetADCPrescale = (SetADCPrescaleEx)GetProcAddress(m_hInst, _T("SetADCPrescale"));
m_pfSetADCPrescale(49);
```

함수	SetFilteringEx

A/D 변환 시 노이즈 성분의 신호를 저감할 수 있도록 상위 20% 하위 20%의 데이터를 무시한 상태의 데이

터들의 평균을 계산하여 A/D 결과로 반환합니다.

◉ typedef void (*SetFilteringEx)(bool bEnable)

[인자]

- bool bEnable : 필터 사용 여부
  - true : 사용
  - false : 사용 안함

C++ 사용법

```
typedef void (*SetFilteringEx)(bool bEnable);
SetFilteringEx        m_pfSetFiltering;
HINSTANCE             m_hInst;
m_hInst = LoadLibrary(L"IEC1000HWCON.DLL");
if(m_hInst == NULL)
{
  return FALSE;
}
m_pfSetFiltering = (SetFilteringEx)GetProcAddress(m_hInst, _T(SetFiltering"));
m_pfSetFiltering(TRUE);  // Filtering 사용
```

함수 GetADCDataEx

A/D 변환된 결과 값(데이터 범위는 0~1024 또는 0~4096)을 반환합니다. 입력 채널의 선택은 메소드(함수) 인자를 통해 선택됩니다. 만약 SetFilteringEx() 함수의 인자값을 true로 하여 호출한 경우, 리턴값은 노이즈를 감소시키기 위한 알고리즘이 동작되어 결과 데이터를 적용하여 반환합니다.

◉ typedef unsigned int (*GetADCDataEx)(int iCh)

[인자]

- int iCh : 입력 채널 번호

[표] ADC Channel에 따른 인자값

ADC Channel	AIN0	AIN1	AIN2	AIN3	AIN4	AIN5
ch	0	1	2	3	4	5

※ AIN4, AIN5의 경우 IEC1000非Lite-Series에서만 지원됩니다.

[리턴값]

- unsigned int : A/D 변환된 결과 값

C++ 사용법

```
typedef unsigned int (*GetADCDataEx)(int iCh);
GetADCDataEx          m_pfGetADCData;
HINSTANCE             m_hInst;
m_hInst = LoadLibrary(L"IEC1000HWCON.DLL");
if(m_hInst == NULL)
{
  return FALSE;
}
m_pfGetADCData = (GetADCDataEx)GetProcAddress(m_hInst, _T("GetADCData"));
CString strADC;
// AIN0 채널의 아날로그 데이터를 변환하여 결과를 수치로 얻음
strADC.Format(L"A/D Converter : %d", m_pfGetADCData(0));
```

함수	GetADCDataFastEx

IEC1000-Series에서만 사용 가능한 함수이며, FilteringEnable 값에 영향을 받지 않습니다.
Sampling 속도가 GetADCDataEx() 함수에 비해 빠르지만 Tollerance(떨림값)이 큽니다.

주의 GetADCDataFastEx() 함수는 IEC1000-Series에만 적용됩니다.

◉ `typedef unsigned int (*GetADCDataFastEx)(int iCh)`

[인자]

- `int iCh` : 입력 채널 번호

[표] ADC Channel에 따른 인자값

ADC Channel	AIN0	AIN1	AIN2	AIN3	AIN4	AIN5
Ch	0	1	2	3	4	5

※ AIN4, AIN5의 경우 IEC1000非Lite-Series에서만 지원됩니다.

[리턴값]

- `unsigned int` : A/D 변환된 결과 값

C++ 사용법

```
typedef unsigned int (*GetADCDataFastEx)(int iCh);
GetADCDataFastEx     m_pfGetADCDataFast;
HINSTANCE            m_hInst;
m_hInst = LoadLibrary(L"IEC1000HWCON.DLL");
if(m_hInst == NULL)
{
  return FALSE;
}
m_pfGetADCDataFast = (GetADCDataFastEx)GetProcAddress(m_hInst, _T("GetADCDataFast"));
CString strADC;
// AIN0 채널의 아날로그 데이터를 변환하여 결과를 수치로 얻음
strADC.Format(L"A/D Converter : %d", m_pfGetADCDataFast(0));
```

함수	GetDetailFilteringADCDataEx

ADC를 통하여 외부의 Analog 신호를 읽을 경우 노이즈 등으로 인하여 데이터가 흔들리게 되며 이러한 이유로 데이터의 오차가 발생합니다. 이렇게 흔들리는 신호를 H/W적인 Filter를 설계하여 원인을 해결할 수도 있지만 실상 Filter를 설계해서 적용하기란 그리 쉽지 않으며 외부 환경의 변화에 따라서 많은 영향을 받는 부분이 있어 적용이 어렵습니다. 따라서 일반적으로 S/W적인 방법으로 Filtering한 효과를 얻을 수 있도록 Filtering Algorithm을 개발하여 적용하고 있습니다. 이에 따라 SmartADC에서 쉽게 Filtering 기능을 제어하여 원하는 입력값을 얻을 수 있도록 S/W적인 Filtering 기능을 지원하고 있습니다.

참고 "SetFilteringEx() 함수" 내용을 참고하시기 바랍니다.

◉ `typedef unsigned int (*GetDetailFilteringADCDataEx)(int iCh, int iRawdataCnt, int iAvailableCnt)`

[인자]

- `int iCh` : 아날로그 입력 채널
- `int iRawdataCnt` : 필터링 전의 외부 아날로그 신호 개수
- `int iAvailableCnt` : RawdataCnt에서 Max, Min 값을 제거한 데이터 신호 개수

[표] ADC Channel에 따른 인자값

ADC Channel	AIN0	AIN1	AIN2	AIN3	AIN4	AIN5
Ch	0	1	2	3	4	5

※ AIN4, AIN5의 경우 IEC1000非Lite-Series에서만 지원됩니다.

[리턴값]

- unsigned int : 필터링 후의 A/D 변환 된 결과 값

C++ 사용법

```
typedef unsigned int (*GetDetailFilteringADCDataEx)(int iCh , int iRawdataCnt,
                                                     int iAvailableCnt);
GetDetailFilteringADCDataEx    m_pfGetDetailFilteringADCData;
HINSTANCE                      m_hInst;
m_hInst = LoadLibrary(L"IEC1000HWCON.DLL");
if(m_hInst == NULL)
{
  return FALSE;
}
m_pfGetDetailFilteringADCData = (GetDetailFilteringADCDataEx)GetProcAddress(m_hInst,
                                        _T("GetDetailFilteringADCData"));
// AIN0 채널의 아날로그 데이터를 변환하여 결과를 수치로 얻음
unsigned int iFilteringData = m_pfGetDetailFilteringADCData (0, 11, 4));
```

4) CSmartADCWrap Class 설명

SmartADCWrap Class는 사용할 인터페이스를 각각 선언 및 얻어오는 복잡한 과정을 거칠 필요없이 한번의 선언과 인스턴싱만으로 편리하게 SmartADC의 기능을 C++(C++, Visual Studio 2005, 2008)에서 지원하도록 Wrapping한 Class입니다. 본 Class를 사용하기 위해서는 "SmartADCWrap.h" 파일과 "SmartADCWrap.cpp" 파일을 대상 프로젝트에 복사한 후 #include "SmartADCWrap.h"을 호출하여 사용하시기 바랍니다.

인터페이스 및 사용 방식은 위에서 설명한 C++ 인터페이스와는 약간의 차이가 있으므로 다음의 내용을 참고하시어 C++에서 적용하시기 바랍니다.

4-1) SmartADCWrap Class 사용하기

C++에서 SmartADCWrap Class를 사용하기 위해서는 아래와 STEP을 진행해야 합니다.

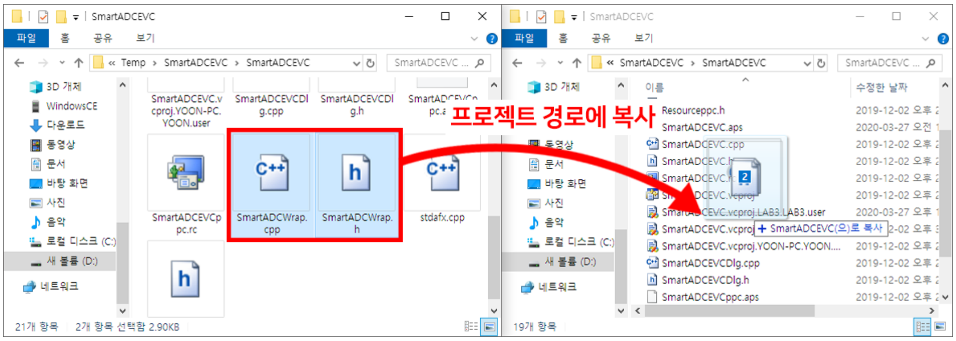
STEP-1	Wrap Class 파일 복사하기

HNS 홈페이지에서 사용하는 IEC-Series에 맞는 예제파일을 다운받은 후 압축을 풉니다.
"SmartADCEVC" 폴더에서 아래 표를 확인하여 Wrap Class 파일을 프로젝트 경로에 복사합니다.

[표] IEC-Series별 Wrap Class 파일

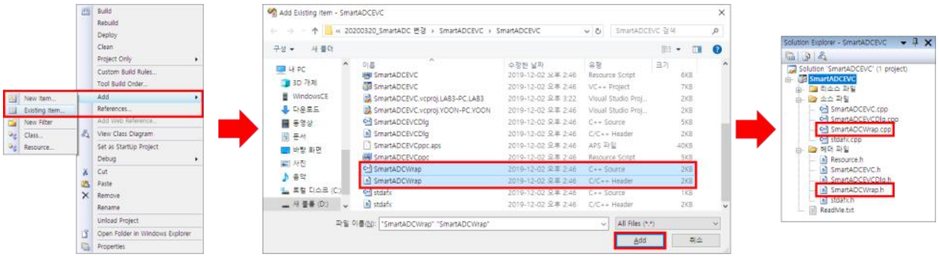
IEC-Series	IEC266-Series	IEC667-Series	IEC1000-Series
Wrap Class 파일	SmartADCWrap.h, SmartADCWrap.cpp		
다운로드 위치	홈페이지 → 자료실 → SmartX Framework 관련 → 6. SmartX Framework 예제파일 → Visual C++ 예제파일 → "SmartADCEVC"		

Wrap Class 파일을 프로젝트 경로에 복사합니다.



STEP-2 Wrap Class 추가하기

복사한 Wrap Class 파일을 프로젝트에 추가합니다.



STEP-3 Wrap Class 사용하기

사용할 소스 파일에 #include<SmartADCWrap.h> 추가한 후 선언 및 인스턴스하여 사용합니다.

```
#include "SmartADCWrap.h"
// SmartADC를 사용하기 위해 변수 선언 및 인스턴싱
CSmartADCWrap m_SmartADC;
// Filtering 기능 사용
m_SmartADC.SetFiltering(TRUE);
// Ch0의 ADC 데이터를 읽는다.
int iADCData = m_SmartADC.GetADCData(0);
// Ch0의 ADC 데이터를 고속으로 읽는다.(IEC1000-Series만 지원됨)
int iADCDataFast = m_SmartADC.GetADCDataFast(0);
// Ch0의 ADC 데이터를 필터링하여 읽는다.
int iADCDetail = m_SmartADC.GetDetailFilteringADCData(0, 11, 4);
```

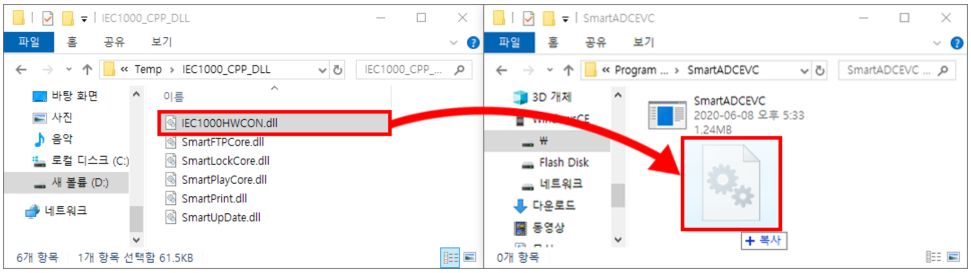
STEP-4 DLL 파일 복사하기

HNS 홈페이지에서 사용하는 IEC-Series에 맞는 예제파일을 다운받은 후 압축을 풉니다. "참조 DLL" 폴더에서 아래 표를 확인하여 사용하는 기능에 맞는 DLL 파일을 IEC-Series에 실행파일과 함께 복사합니다.

[표] IEC-Series별 DLL 파일과 다운로드 경로

IEC-Series	IEC266-Series	IEC667-Series	IEC1000-Series
DLL 파일	IEC266HWCON.dll	IEC667HWCON.dll	IEC1000HWCON.dll
다운로드 위치	홈페이지 → 자료실 → SmartX Framework 관련 → 5. SmartX C++ For DLL 파일		

DLL 파일을 실행파일 위치에 복사합니다.

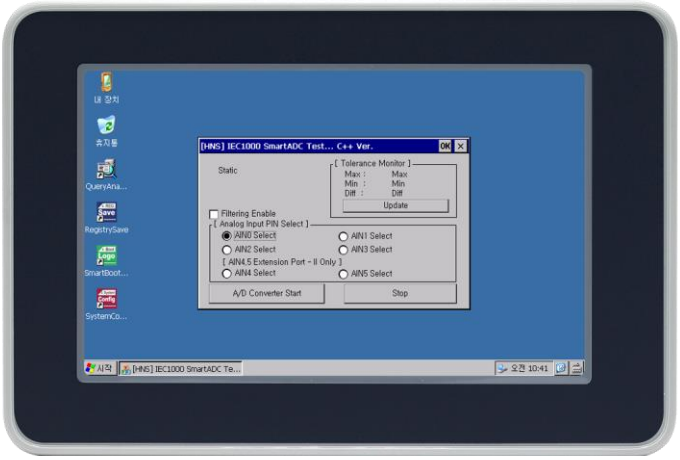


참고	자세한 DLL 파일의 복사 경로는 "DLL 파일 및 복사 경로" 내용을 참고하시기 바랍니다.

4-2) SmartADCWrap Class 예제 사용하기

SmartADCWrap Class를 활용한 예제를 제공하고 있으며, 예제파일은 다음 경로에서 다운로드하여 참고하시기 바랍니다.

[예제 파일 다운로드 위치]
홈페이지(www.hnsts.co.kr) → 자료실 → SmartX Framework 예제파일 다운로드 → “SmartADCEVC”



3. SmartPWM

SmartX Framework는 기본적으로 .Net Compact Framework에서만 지원되도록 설계되어 있지만, 일부 기능을 C++에서 편리하게 사용할 수 있도록 외부 DLL 파일을 제공하고 있습니다. 따라서 SmartPWM을 사용하기 위해서는 반드시 외부 DLL 파일을 실행 파일과 같이 IEC-Series에 배포해야 하며, "1. 인터페이스 함수 선언 → 2. 라이브러리 메모리 로드 처리 → 3. 인터페이스 함수 얻기" 과정을 거쳐야 합니다. 이때 IEC-Series 및 기능별로 로드해야 할 외부 DLL 파일이 다르므로 다음 내용을 확인하시기 바랍니다.

참조	SmartPWM의 자세한 설명은 "홈페이지 → 자료실 → SmartX Framework 관련 → 프로그래밍 가이드 → SmartPWM"을 참조하시기 바랍니다.

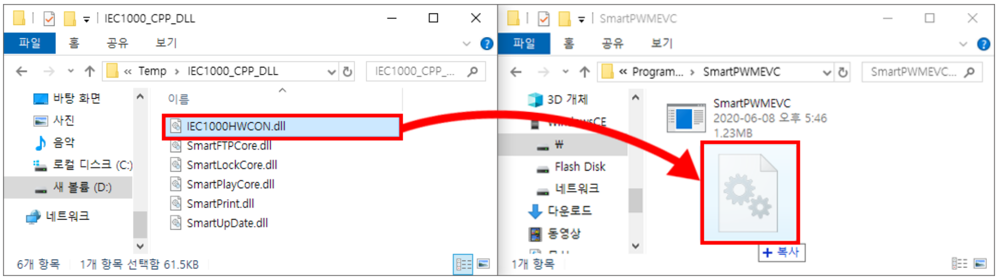
1) DLL 파일 및 복사 경로

HNS 홈페이지에서 사용하는 IEC-Series에 맞는 C++ For DLL 파일을 다운로드 받은 후 압축을 풉니다. 아래 표를 확인하여 DLL 파일을 IEC-Series에 실행 파일과 함께 복사합니다.

[표] IEC-Series별 DLL 파일과 다운로드 경로

IEC-Series	IEC266-Series	IEC667-Series	IEC1000-Series
DLL 파일	IEC266HWCON.dll	IEC667HWCON.dll	IEC1000HWCON.dll
다운로드 위치	홈페이지 → 자료실 → SmartX Framework 관련 → 5. SmartX C++ For DLL 파일		

DLL 파일을 IEC-Series에 복사합니다. DLL 파일을 사용하는 실행 파일이 저장된 폴더에 저장하는 것을 권장하며, 다른 경로에 저장하는 경우 "인터페이스 선언 및 얻기"의 참고 내용을 확인하시기 바랍니다.
※ 개발 PC에서 프로그램을 배포하는 경우 실행 파일은 "Program File" 폴더에 배포됩니다.



2) 인터페이스 선언 및 얻기

인터페이스 사용을 위해 사용할 인터페이스의 선언 후 외부 DLL 파일을 메모리에 로드하여 인터페이스를 얻는 과정을 거쳐야 합니다. 이때 메모리 로드 처리를 위해 LoadLibrary() 함수 호출 시 상단 표의 DLL 파일의 이름을 인자로 설정합니다. 아래 코드는 예시이며, 필요로 하는 인터페이스 함수를 사용하기 위해서는 다음 내용을 참고하시기 바랍니다.

참고	DLL 파일 저장 경로에 따른 LoadLibrary() 함수 인자값
DLL 파일이 실행 파일과 같은 경로에 저장된 경우, LoadLibrary() 함수 인자값을 DLL 파일 이름만으로 설정해 호출할 수 있습니다. 만약 DLL 파일이 실행 파일과 다른 경로에 저장된 경우, LoadLibray() 함수의 인자값에는 저장된 DLL 파일의 전체 경로를 인자값으로 설정해야 합니다.	

[C++] 인터페이스 얻기 및 사용하기 예시 코드

```
// 1. 인터페이스 함수 선언
typedef void (*PWMRestoreEx)();
PWMRestoreEx  m_pfPWMRestore;
// 2. 라이브러리 메모리 로드 처리
HINSTANCE  m_hInst;
// SmartPWM의 인터페이스를 사용하기 위해 IEC1000HWCON.dll 파일을 메모리에 로드
// IEC-Series 및 사용하는 기능별로 외부 DLL 파일이 다르며, 상단 표를 참고하시기 바랍니다.
m_hInst = LoadLibrary(L"IEC1000HWCON.dll");
// 외부 DLL 파일이 다른 경로(Flash Disk\\extdlls)에 저장된 경우
// m_hInst = LoadLibrary(L"Flash Disk\\extdlls\\IEC1000HWCON.dll");
if(m_hInst == NULL)
{
  // 메모리 로드 실패
  return FALSE;
}
// 3. 인터페이스 함수 얻기
m_pfPWMRestore = (PWMRestoreEx)GetProcAddress(m_hInst, _T("PWMRestore"));
// 4. 함수 사용
m_pfPWMRestore();
```

참고 Wrapping Class 제공안내

사용할 인터페이스를 각각 선언 및 얻어오는 복잡한 과정을 거칠 필요 없이 한 번의 선언과 인스턴싱만으로 편리하게 SmartPWM의 인터페이스를 사용할 수 있도록 Wrapping Class를 제공하고 있습니다. 자세한 내용은 "CSmartPWMWrap Class 설명" 내용을 참고하시기 바랍니다.

3) C++ 인터페이스 설명

함수	PWMRestoreEx

프로그램 종료 시 PWM 관련 레지스터를 초기 상태로 복구합니다. PWMRestoreEx() 함수 호출 시 PWM과 관련하여 사용중인 리소스를 모두 반환하여 시스템의 안정성을 높여줍니다.

◉ typedef void (*PWMRestoreEx)()

C++ 사용법

```
typedef void (*PWMRestoreEx)();
PWMRestoreEx         m_pfPWMRestore;
HINSTANCE            m_hInst;
m_hInst = LoadLibrary(L"IEC1000HWCON.DLL");
if(m_hInst == NULL)
{
  return FALSE;
}
m_pfPWMRestore = (PWMRestoreEx)GetProcAddress(m_hInst, _T("PWMRestore"));
m_pfPWMRestore();  // PWM을 사용하기 위해 변경된 레지스터를 복원
```

함수	SetPWMCarryFreq1Ex, SetPWMCarryFreq2Ex

PWM의 Carrier Frequency 관련 인자들을 모두 변경합니다. 즉 PreScaler, ClockDivider, PWMCounter 속성을 일괄 변경합니다.

◉ typedef void (*SetPWMCarryFreq1Ex)(unsigned char ucPreScaler, unsigned char ucDivider, unsigned int uiPWMCounter)

◉ typedef void (*SetPWMCarryFreq2Ex)(unsigned char ucPreScaler, unsigned char ucDivider,
unsigned int uiPWMCounter)

[인자]

- unsigned char ucPreScaler : PreScaler값
- unsigned char ucDivider : ClockDivider값
- unsigned int uiPWMCounter : PWMCounter값

C++ 사용법

```
typedef void (*SetPWMCarryFreq1Ex)(unsigned char ucPreScaler, unsigned char ucDivider,
unsigned int uiPWMCounter);
SetPWMCarryFreq1Ex     m_pfSetPWMCarryFreq1;
HINSTANCE            m_hInst;
m_hInst = LoadLibrary(L"IEC1000HWCON.DLL");
if(m_hInst == NULL)
{
  return FALSE;
}
m_pfSetPWMCarryFreq1= (SetPWMCarryFreq1Ex)GetProcAddress(m_hInst, _T("SetPWMCarryFreq1"));
m_pfSetPWMCarryFreq1(1, 0, 200); // 각각 SetPWMPrescaler, SetPWMClkDivider1, SetPWMCounter1 설정
```

함수	SetPWMClkDivider1Ex, SetPWMClkDivider2Ex

PreScaler에서 PWM Clock이 공급되면 이 Clock은 다시 ClockDivider에 의해서 분주됩니다. ClockDivider는 이 분주 값을 설정합니다.

주의 IEC266-Series는 1/1 분주비를 지원하지 않습니다.

◉ typedef void (*SetPWMClkDivider1Ex)(unsigned char ucClkDivider1)

◉ typedef void (*SetPWMClkDivider2Ex)(unsigned char ucClkDivider2)

[인자]

- unsigned char ucClkDivider1 : 분주 설정 값 0~4(IEC266-Series의 경우 0~3)
- unsigned char ucClkDivider2 : 분주 설정 값 0~4

[표] IEC-Series별 ucClkDivider1/2 인자값에 따른 분주비

인자값	0	1	2	3	4
IEC266-Series	1/2	1/4	1/8	1/16	-
IEC667-Series	1/1	1/2	1/4	1/8	1/16
IEC1000-Series	1/1	1/2	1/4	1/8	1/16

C++ 사용법

```
typedef void (*SetPWMClkDivider1Ex)(unsigned char ucClkDivider1);
SetPWMClkDivider1Ex            m_pfSetPWMClkDivider1;
HINSTANCE                      m_hInst;
m_hInst = LoadLibrary(L"IEC1000HWCON.DLL");
if(m_hInst == NULL)
{
  return FALSE;
}
m_pfSetPWMClkDivider1 = (SetPWMClkDivider1Ex)GetProcAddress(m_hInst, _T("SetPWMClkDivider1"));
m_pfSetPWMClkDivider1(2); // Clock Divider 설정
```

함수 SetPWMCounter1Ex, SetPWMCounter2Ex

PWMCounter는 16Bit(65536)이며 PWM의 캐리어 주파수를 세밀하게 제어할 수 있습니다.

주의 PWMCounter는 최소 100이상으로 설정하시기 바랍니다.

DutyRate가 0~100의 실수형이므로 PWMCounter는 최소 100이상 되어야 듀티비를 올바르게 조절 가능합니다.

◉ typedef void (*SetPWMCounter1Ex)(unsigned int uiPwmCounter1)

◉ typedef void (*SetPWMCounter2Ex)(unsigned int uiPwmCounter2)

[인자]

- unsigned int uiPwmCounter1 : PWMCounter값(0~65535)
- unsigned int uiPwmCounter2 : PWMCounter 값(0~65535)

C++ 사용법

```
typedef void (*SetPWMCounter1Ex)(unsigned int uiPwmCounter1);
SetPWMCounter1Ex      m_pfSetPWMCounter1;
HINSTANCE             m_hInst;
m_hInst = LoadLibrary(L"IEC1000HWCON.DLL");
if(m_hInst == NULL)
{
  return FALSE;
}
m_pfSetPWMCounter1 = (SetPWMCounter1Ex)GetProcAddress(m_hInst, _T("SetPWMCounter1"));
m_pfSetPWMCounter1(200); // PWMCounter 설정
```

함수 SetPWMDeadTimeEx

데드 타임 설정 기능이 있어 전력 제어용 PWM으로 사용하실 수 있습니다. 데드 타임 설정값이 0보다 큰 경우 PWM1과 PWM2는 서로 연동되어 동작합니다. 즉 PWM1은 High Side, PWM2는 Low Side로 연동되어 동작됩니다. 데드 타임이 설정되면 PWM1의 제어로 PWM2까지 제어됩니다.

◉ typedef void (*SetPWMDeadTimeEx)(unsigned char ucDeadTime)

[인자]

- unsigned char ucDeadTime : 0 → 데드 타임 설정 안함, 1 ~ 255 → 데드 타임 설정

C++ 사용법

```
typedef void (*SetPWMDeadTimeEx)(unsigned char ucDeadTime);
SetPWMDeadTimeEx      m_pfSetPWMDeadTime;
HINSTANCE             m_hInst;
m_hInst = LoadLibrary(L"IEC1000HWCON.DLL");
if(m_hInst == NULL)
{
  return FALSE;
}
m_pfSetPWMDeadTime = (SetPWMDeadTimeEx)GetProcAddress(m_hInst, _T("SetPWMDeadTime"));
m_pfSetPWMDeadTime (10); // 데드 타임 On 및 Time 10 설정
```

함수	SetPWMDutyRate1Ex, SetPWMDutyRate2Ex

PWM 출력의 Duty Rate(듀티비)를 백분율로 설정합니다.

◉ typedef void (*SetPWMDutyRate1Ex)(double dDutyRate1)

◉ typedef void (*SetPWMDutyRate2Ex)(double dDutyRate2)

[인자]

- double dDutyRate1 : 0 ~ 100.00 (실수형 데이터 : double type)
- double dDutyRate2 : 0 ~ 100.00 (실수형 데이터 : double type)

C++ 사용법

```
typedef void (*SetPWMDutyRate1Ex)(double dDutyRate1);
SetPWMDutyRate1Ex     m_pfSetPWMDutyRate1;
HINSTANCE             m_hInst;
m_hInst = LoadLibrary(L"IEC1000HWCON.DLL");
if(m_hInst == NULL)
{
  return FALSE;
}
m_pfSetPWMDutyRate1 = (SetPWMDutyRate1Ex)GetProcAddress(m_hInst, _T("SetPWMDutyRate1"));
m_pfSetPWMDutyRate1(77.0); // PWM Duty Rate 77.0% 설정
```

함수	SetPWMPolarity1Ex, SetPWMPolarity2Ex

PWM 출력 파형의 정•부 논리를 변경합니다. 즉 출력 파형을 반전하여 PWM 출력을 할 수 있도록 합니다.

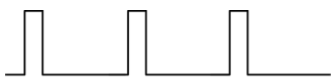
◉ typedef void (*SetPWMPolarity1Ex)(bool bPolarity1)

◉ typedef void (*SetPWMPolarity2Ex)(bool bPolarity2)

[인자]

- bool bPolarity1, bPolarity2 : PWM 출력 반전 처리 여부
  - true : 반전 처리
  - false : 비반전 처리

[표] Polarity 속성값 설정에 따른 출력 파형의 변화 (Duty Rate : 20)

속성값	false (HIGHACTIVE)	true (LOWACTIVE)
출력파형		

C++ 사용법

```
typedef void (*SetPWMPolarity1Ex)(bool bPolarity1);
SetPWMPolarity1Ex     m_pfSetPWMPolarity1;
HINSTANCE             m_hInst;
m_hInst = LoadLibrary(L"IEC1000HWCON.DLL");
if(m_hInst == NULL)
{
  return FALSE;
```

```
}
m_pfSetPWMPolarity1 = (SetPWMPolarity1Ex)GetProcAddress(m_hInst, _T("SetPWMPolarity1"));
m_pfSetPWMPolarity1(false); // HIGHACTIVE(PWM 출력 비반전 처리) 설정
```

함수 SetPWMPrescalerEx

Prescaler는 0~255값을 설정할 수 있으며, Prescaler의 변경으로 PWM의 Carrier Frequency를 조정할 수 있습니다. PWM1과 PWM2는 동일한 PreScaler를 사용합니다.

◉ typedef void (*SetPWMPrescalerEx)(unsigned char ucPrescaler)

[인자]

- unsigned char ucPrescaler : Prescaler 값(0~255)

C++ 사용법

```
typedef void (*SetPWMPrescalerEx)(unsigned char ucPrescaler);
SetPWMPrescalerEx    m_pfSetPWMPrescaler;
HINSTANCE            m_hInst;
m_hInst = LoadLibrary(L"IEC1000HWCON.DLL");
if(m_hInst == NULL)
{
  return FALSE;
}
m_pfSetPWMPrescaler = (SetPWMPrescalerEx)GetProcAddress(m_hInst, _T("SetPWMPrescaler"));
m_pfSetPWMPrescaler(2); // Prescaler 설정
```

함수 Start_PWM1Ex, Start_PWM2Ex

PWM 출력을 시작합니다.

◉ typedef void (*Start_PWM1Ex)()

◉ typedef void (*Start_PWM2Ex)()

C++ 사용법

```
typedef void (*Start_PWM1Ex)();
Start_PWM1Ex         m_pfStart_PWM1;
HINSTANCE            m_hInst;
m_hInst = LoadLibrary(L"IEC1000HWCON.DLL");
if(m_hInst == NULL)
{
  return FALSE;
}
m_pfStart_PWM1 = (Start_PWM1Ex)GetProcAddress(m_hInst, _T("Start_PWM1"));
m_pfStart_PWM1(); // PWM1 출력 시작
```

함수 Stop_PWM1Ex, Stop_PWM2Ex

PWM 출력을 중지합니다.

◉ typedef void (*Stop_PWM1Ex)()

◉ typedef void (*Stop_PWM2Ex)()

C++ 사용법

```
typedef void (*Stop_PWM1Ex)();
Stop_PWM1Ex          m_pfStop_PWM1;
HINSTANCE            m_hInst;
m_hInst = LoadLibrary(L"IEC1000HWCON.DLL");
if(m_hInst == NULL)
{
  return FALSE;
}
m_pfStop_PWM1 = (Stop_PWM1Ex)GetProcAddress(m_hInst, _T("Stop_PWM1"));
m_pfStop_PWM1(); // PWM1 출력 중지
```

4) CSmartPWMWrap Class 설명

SmartPWMWrap Class는 사용할 인터페이스를 각각 선언 및 얻어오는 복잡한 과정을 거칠 필요 없이 한 번의 선언과 인스턴싱만으로 편리하게 SmartPWM의 기능을 C++(C++, Visual Studio 2005, 2008)에서 지원하도록 Wrapping한 Class입니다. 본 Class를 사용하기 위해서는 “SmartPWMWrap.h” 파일과 “SmartPWMWrap.cpp” 파일을 대상 프로젝트에 복사한 후 #include "SmartPWMWrap.h"을 호출하여 사용하시기 바랍니다.

인터페이스 및 사용 방식은 위에서 설명한 C++ 인터페이스와는 약간의 차이가 있으므로 다음의 내용을 참고하시어 C++에서 적용하시기 바랍니다.

4-1) SmartPWMWrap Class 사용하기

C++에서 SmartPWMWrap Class를 사용하기 위해서는 아래와 STEP을 진행해야 합니다.

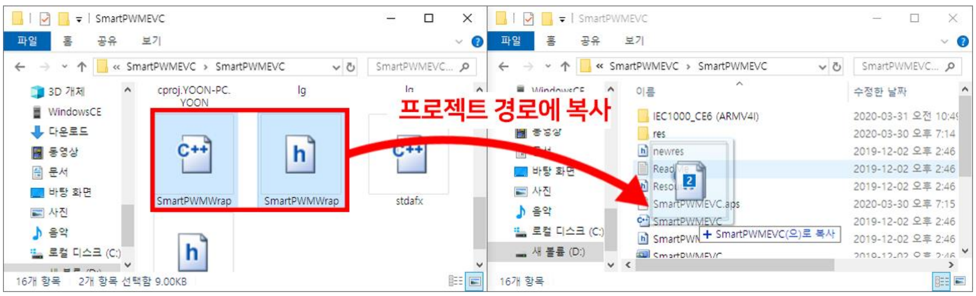
STEP-1	Wrap Class 파일 복사하기

HNS 홈페이지에서 사용하는 IEC-Series에 맞는 예제파일을 다운받은 후 압축을 풉니다.
“SmartPWMEVC” 폴더에서 아래 표를 확인하여 Wrap Class 파일을 프로젝트 경로에 복사합니다.

[표] IEC-Series별 Wrap Class 파일

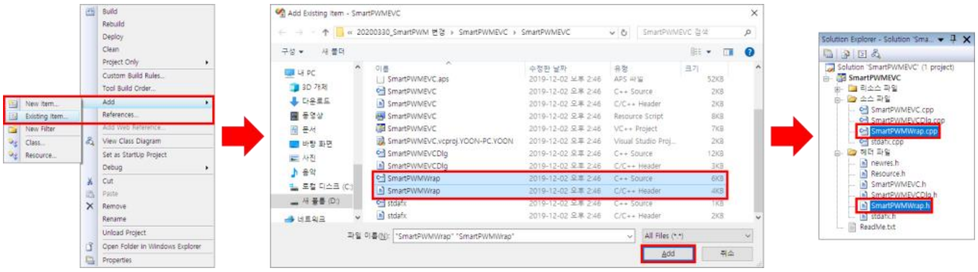
IEC-Series	IEC266-Series	IEC667-Series	IEC1000-Series
Wrap Class 파일	SmartPWMWrap.h, SmartPWMWrap.cpp		
다운로드 위치	홈페이지 → 자료실 → SmartX Framework 관련 → 6. SmartX Framework 예제파일 → Visual C++ 예제파일 → "SmartPWMEVC"		

Wrap Class 파일을 프로젝트 경로에 복사합니다.



STEP-2 Wrap Class 추가하기

복사한 Wrap Class 파일을 프로젝트에 추가합니다.



STEP-3 Wrap Class 사용하기

사용할 소스 파일에 #include〈SmartPWMWrap.h〉 추가한 후 선언 및 인스턴스하여 사용합니다.

```
#include "SmartPWMWrap.h"
// SmartPWM을 사용하기 위해 변수 선언 및 인스턴싱
CSmartPWMWrap m_SmartPWM;
// Prescaler 설정 : 100
m_SmartPWM.SetPWMPrescaler(100);
// ClockDivder 설정 : 1/4 (IEC1000-Series 기준)
m_SmartPWM.SetPWMClkDivider1(2);
// PWMCounter 설정 : 300000
m_SmartPWM.SetPWMCounter1(30000);
// Duty Rate 설정 : 50.0%
m_SmartPWM.SetPWMDutyRate1(50.0);
// DeadTime 기능 사용 안함 : 0
m_SmartPWM.SetPWMDeadTime(0);
// 반전되지 않은 파형을 출력
m_SmartPWM.SetPWMPolarity1(false);
// PWM 출력을 시작
m_SmartPWM.Start_PWM1();
```

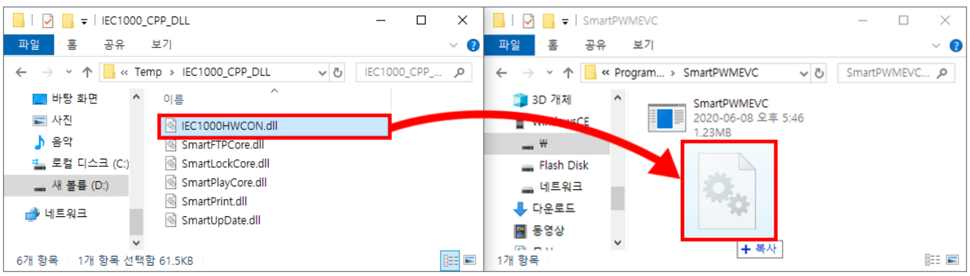
STEP-4 DLL 파일 복사하기

HNS 홈페이지에서 사용하는 IEC-Series에 맞는 예제파일을 다운받은 후 압축을 풉니다. "참조 DLL" 폴더에서 아래 표를 확인하여 사용하는 기능에 맞는 DLL 파일을 IEC-Series에 실행파일과 함께 복사합니다.

[표] IEC-Series별 DLL 파일과 다운로드 경로

IEC-Series	IEC266-Series	IEC667-Series	IEC1000-Series
DLL 파일	IEC266HWCON.dll	IEC667HWCON.dll	IEC1000HWCON.dll
다운로드 위치	홈페이지 → 자료실 → SmartX Framework 관련 → 5. SmartX C++ For DLL 파일		

DLL 파일을 실행 파일 위치에 복사합니다.



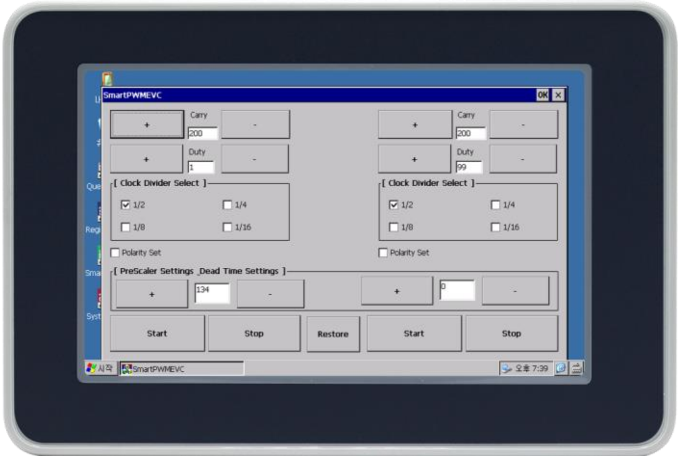
참고 자세한 DLL 파일의 복사 경로는 "DLL 파일 및 복사 경로" 내용을 참고하시기 바랍니다.

4-2) SmartPWMWrap Class 예제 사용하기

SmartPWMWrap Class를 활용한 예제를 제공하고 있으며, 예제파일은 다음 경로에서 다운로드하여 참고하시기 바랍니다.

[예제 파일 다운로드 위치]

홈페이지(www.hnsts.co.kr) → 자료실 → SmartX Framework 예제파일 다운로드 → “SmartPWMEVC”



4. SmartSound

SmartX Framework는 기본적으로 .Net Compact Framework에서만 지원되도록 설계되어 있지만, 일부 기능을 C++에서 편리하게 사용할 수 있도록 외부 DLL 파일을 제공하고 있습니다. 따라서 SmartSound를 사용하기 위해서는 반드시 외부 DLL 파일을 실행 파일과 같이 IEC-Series에 배포해야 하며, "1. 인터페이스 함수 선언 → 2. 라이브러리 메모리 로드 처리 → 3. 인터페이스 함수 얻기" 과정을 거쳐야 합니다. 이때 IEC-Series 및 기능별로 로드해야 할 외부 DLL 파일이 다르므로 다음 내용을 확인하시기 바랍니다.

참조 SmartSound의 자세한 설명은 "홈페이지 → 자료실 → SmartX Framework 관련 → 프로그래밍 가이드 → SmartSound"를 참조하시기 바랍니다.

1) DLL 파일 및 복사 경로

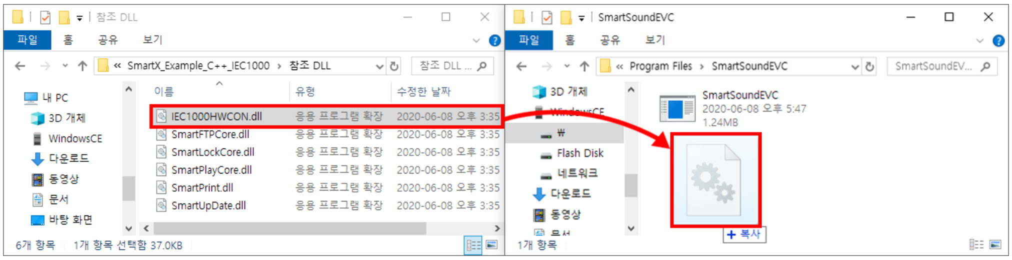
HNS 홈페이지에서 사용하는 IEC-Series에 맞는 C++ For DLL 파일을 다운로드 받은 후 압축을 풉니다. 아래 표를 확인하여 DLL 파일을 IEC-Series에 실행 파일과 함께 복사합니다.

[표] IEC-Series별 DLL 파일과 다운로드 경로

IEC-Series	IEC266-Series	IEC667-Series	IEC1000-Series
DLL 파일	IEC266HWCON.dll	IEC667HWCON.dll	IEC1000HWCON.dll
다운로드 위치	홈페이지 → 자료실 → SmartX Framework 관련 → 5. SmartX C++ For DLL 파일		

DLL 파일을 IEC-Series에 복사합니다. DLL 파일을 사용하는 실행 파일이 저장된 폴더에 저장하는 것을 권장하며, 다른 경로에 저장하는 경우 "인터페이스 선언 및 얻기"의 참고 내용을 확인하시기 바랍니다.

※ 개발 PC에서 프로그램을 배포하는 경우 실행 파일은 "Program File" 폴더에 배포됩니다.



2) 인터페이스 선언 및 얻기

인터페이스 사용을 위해 사용할 인터페이스의 선언 후 외부 DLL 파일을 메모리에 로드하여 인터페이스를 얻는 과정을 거쳐야 합니다. 이때 메모리 로드 처리를 위해 LoadLibrary() 함수 호출 시 상단 표의 DLL 파일의 이름을 인자로 설정합니다. 아래 코드는 예시이며, 필요로 하는 인터페이스 함수를 사용하기 위해서는 다음 내용을 참고하시기 바랍니다.

참고 DLL 파일 저장 경로에 따른 LoadLibrary() 함수 인자값

DLL 파일이 실행 파일과 같은 경로에 저장된 경우, LoadLibrary() 함수 인자값을 DLL 파일 이름만으로 설정해 호출할 수 있습니다. 만약 DLL 파일이 실행 파일과 다른 경로에 저장된 경우, LoadLibray() 함수의 인자값에는 저장된 DLL 파일의 전체 경로를 인자값으로 설정해야 합니다.

[C++] 인터페이스 얻기 및 사용하기 예시 코드

```
// 1. 인터페이스 함수 선언
```

```
typedef BOOL (*Play_SoundEx)();
Play_SoundEx          m_pfPlay_Sound;
// 2. 라이브러리 메모리 로드 처리
HINSTANCE  m_hInst;
// SmartSound의 인터페이스를 사용하기 위해 IEC1000HWCON.dll 파일을 메모리에 로드
// IEC-Series 및 사용하는 기능별로 외부 DLL 파일이 다르며, 상단 표를 참고하시기 바랍니다.
m_hInst = LoadLibrary(L"IEC1000HWCON.dll");
// 외부 DLL 파일이 다른 경로(Flash Disk\\extdlls)에 저장된 경우
// m_hInst = LoadLibrary(L"Flash Disk\\extdlls\\IEC1000HWCON.dll");
if(m_hInst == NULL)
{
  // 메모리 로드 실패
  return FALSE;
}
// 3. 인터페이스 함수 얻기
m_pfPlay_Sound = (Play_SoundEx)GetProcAddress(m_hInst, _T("Play_Sound"));
// 4. 함수 사용
m_pfPlay_Sound();
```

참고 Wrapping Class 제공안내

사용할 인터페이스를 각각 선언 및 얻어오는 복잡한 과정을 거칠 필요 없이 한 번의 선언과 인스턴싱만으로 편리하게 SmartSound의 인터페이스를 사용할 수 있도록 Wrapping Class를 제공하고 있습니다. 자세한 내용은 “CSmartSoundWrap Class 설명” 내용을 참고하시기 바랍니다.

3) C++ 인터페이스 설명

함수	SetPlayFilePathEx

재생할 Sound 파일의 경로를 설정합니다.

◉ typedef void (*SetPlayFilePathEx)(LPTSTR strPath)

[인자]

- LPTSTP strPath : 재생할 Sound 파일의 경로와 파일명을 설정

C++ 사용법

```
typedef void (*SetPlayFilePathEx)(LPTSTR strPath);
HINSTANCE                    m_hInst;
SetPlayFilePathEx            m_pfSetPlayFilePath;
m_hInst = LoadLibrary(L"IEC1000HWCON.DLL");
if(m_hInst == NULL)
{
     return FALSE;
}
m_pfSetPlayFilePath = (SetPlayFilePathEx)GetProcAddress(m_hInst, _T("SetPlayFilePathEx"));
CString strPathName;
CString strFileName;
TCHAR szFilter[] = L"*.*||";
// Play할 Sound 파일 선택
CFileDialog dlg(TRUE, NULL, NULL, OFN_HIDEREADONLY, szFilter);
if (IDOK != dlg.DoModal())
{
     return;
```

```
}
strPathName = dlg.GetPathName();
m_txtFilePath.SetWindowText(strPathName);
// Play할 Sound 파일 경로 지정
m_pfSetPlayFilePath(strPathName.GetBuffer(200));
```

함수 GetVolumeEx, SetVolumeEx

- GetVolumeEx() : IEC-Series의 소리 크기를 가져옵니다.
- SetVolumeEx() : IEC-Series의 소리 크기를 설정하거나 읽습니다.

◉ typedef DWORD (*GetVolumeEx)()

◉ typedef void (*SetVolumeEx)(DWORD dwVol)

[인자]

- DWORD dwVol : 볼륨 설정값

[리턴값]

- DWORD : 현재 볼륨 설정값

C++ 사용법

```
typedef unsigned long DWORD;
typedef DWORD (*GetVolumeEx)();
typedef void (*SetVolumeEx)(DWORD dwVol);
HINSTANCE                    m_hInst;
GetVolumeEx                  m_pfGetVolume;
SetVolumeEx                  m_pfSetVolume;
m_hInst = LoadLibrary(L"IEC1000HWCON.DLL");
if(m_hInst == NULL)
{
     return FALSE;
}
m_pfGetVolume = (GetVolumeEx)GetProcAddress(m_hInst, _T("GetVolume"));
m_pfSetVolume = (SetVolumeEx)GetProcAddress(m_hInst, _T("SetVolume"));
// 현재 소리 크기를 얻음
DWORD dwVol = m_pfGetVolume();
dwVol += 1000;
// 소리 크기 1000 증가
m_pfSetVolume(dwVol);
```

함수 Play_SoundEx, Pause_SoundEx, Stop_SoundEx

현재 지정된 Sound 파일을 제어합니다.

- PlayEx() : Sound 파일을 Play합니다.
- PauseEx() : 현재 Play 중인 Sound 파일을 일시 정지합니다.
- StopEx() : 현재 Play 중인 Sound 파일을 Stop합니다.

◉ typedef bool (*Play_SoundEx)()

◉ typedef bool (*Pause_SoundEx)()

◉ typedef bool (*Stop_SoundEx)()

[리턴값]

- bool : Sound 파일의 Play, Pause, Stop 성공 여부
  - true : 성공
  - false : 실패

C++ 사용법

```
typedef bool (*Play_SoundEx)();
typedef bool (*Pause_SoundEx)();
typedef void (*Stop_SoundEx)();
HINSTANCE           m_hInst;
Play_SoundEx        m_pfPlay_Sound;
Pause_SoundEx       m_pfPause_Sound;
Stop_SoundEx        m_pfStop_Sound;
m_hInst = LoadLibrary(L"IEC1000HWCON.DLL");
if(m_hInst == NULL)
{
     return FALSE;
}
m_pfPlay_Sound = (Play_SoundEx)GetProcAddress(m_hInst, _T("Play_Sound"));
m_pfPause_Sound = (Pause_SoundEx)GetProcAddress(m_hInst, _T("Pause_Sound"));
m_pfStop_Sound = (Stop_SoundEx)GetProcAddress(m_hInst, _T("Stop_Sound"));
m_pfPlay_Sound();  // 지정된 파일을 Play
m_pfPause_Sound(); // 일시 정지
m_pfPlay_Sound();  // 정지한 곳부터 다시 Play
m_pfStop_Sound();  // Sound 파일 재생을 Stop
```

함수	Is_Play

현재 재생 중인 Sound 파일의 Play 상태를 확인합니다.

◉ typedef bool Is_Play()

[리턴값]

- bool : Sound 파일의 Play 여부
  - true : Play 중
  - false : Stop

C++ 사용법

```
typedef bool (*Is_Play)();
Is_Play             m_pfIsPlay;
HINSTANCE           m_hInst;
m_hInst = LoadLibrary(L"IEC1000HWCON.DLL");
if(m_hInst == NULL)
{
     return FALSE;
}
m_pfIsPlay = (Is_Play)GetProcAddress(m_hInst, _T("Is_Play"));
CString strMsg;
// 동영상이 재생 중인지 확인
```

```
if(m_pfIsPlay() == true)
{
    strMsg.Format(L"재생 중 …");  // 음원 재생 중
}
```

4) CSmartSoundWrap Class 설명

SmartSoundWrap Class는 사용할 인터페이스를 각각 선언 및 얻어오는 복잡한 과정을 거칠 필요 없이 한 번의 선언과 인스턴싱만으로 편리하게 SmartSound의 기능을 C++(C++, Visual Studio 2005, 2008)에서 지원하도록 Wrapping 한 Class입니다. 본 Class를 사용하기 위해서는 “SmartSoundWrap.h” 파일과 “SmartSoundWrap.cpp” 파일을 대상 프로젝트에 복사한 후 #include "SmartSoundWrap.h"을 호출하여 사용하시기 바랍니다.

인터페이스 및 사용 방식은 위에서 설명한 C++ 인터페이스와는 약간의 차이가 있으므로 다음의 내용을 참고하시어 C++에서 적용하시기 바랍니다.

4-1) SmartSoundWrap Class 사용하기

C++에서 SmartSoundWrap Class를 사용하기 위해서는 아래와 STEP을 진행해야 합니다.

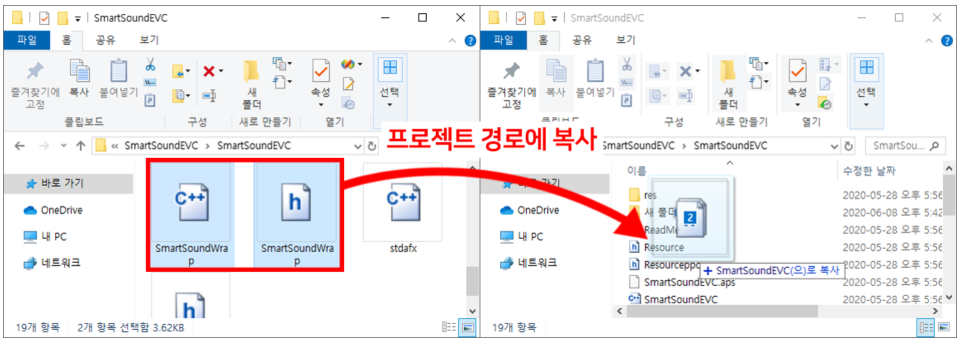
STEP-1 Wrap Class 파일 복사하기

HNS 홈페이지에서 사용하는 IEC-Series에 맞는 예제파일을 다운받은 후 압축을 풉니다.
“SmartSoundEVC” 폴더에서 아래 표를 확인하여 Wrap Class 파일을 프로젝트 경로에 복사합니다.

[표] IEC-Series별 Wrap Class 파일

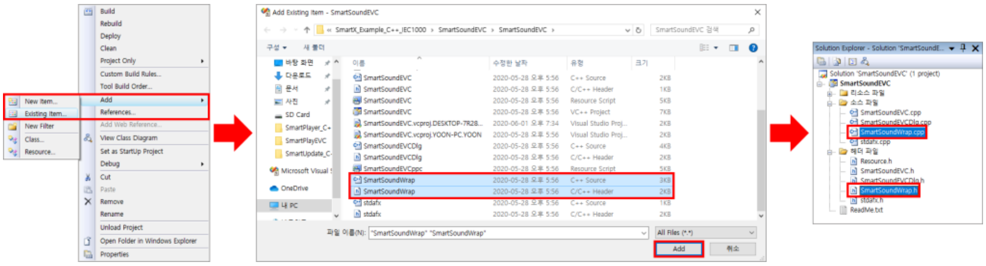
IEC-Series	IEC266-Series	IEC667-Series	IEC1000-Series
Wrap Class 파일	SmartSoundWrap.h, SmartSoundWrap.cpp		
다운로드 위치	홈페이지 → 자료실 → SmartX Framework 관련 → 6. SmartX Framework 예제파일 → Visual C++ 예제파일 → "SmartSoundEVC"		

Wrap Class 파일을 프로젝트 경로에 복사합니다.



STEP-2 Wrap Class 추가하기

복사한 Wrap Class 파일을 프로젝트에 추가합니다.



STEP-3 Wrap Class 사용하기

사용할 소스 파일에 #include<SmartSoundWrap.h> 추가한 후 선언 및 인스턴스하여 사용합니다.

```
#include "SmartSoundWrap.h"
// SmartSound를 사용하기 위해 변수 선언 및 인스턴싱
CSmartSoundWrap               m_SmartSound;
// Play할 Sound 파일 지정
m_SmartSound.SetPlayFilePath(strPath);
// Sound 볼륨 설정 : 500
m_SmartSound.SetVolume(500);
// Sound 파일을 Play
m_SmartSound.Play_Sound();
// 일시 정지
m_SmartSound.Pause_Sound();
// 정지한 곳부터 다시 Play
m_SmartSound.Play_Sound();
// Sound 파일을 Stop
m_SmartSound.Stop_Sound();
```

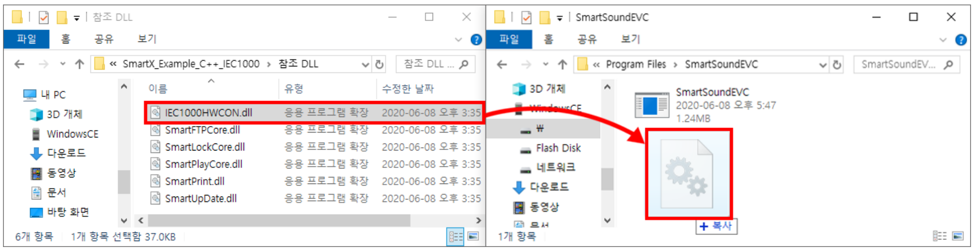
STEP-4 DLL 파일 복사하기

HNS 홈페이지에서 사용하는 IEC-Series에 맞는 예제파일을 다운받은 후 압축을 풉니다. "참조 DLL" 폴더에서 아래 표를 확인하여 사용하는 기능에 맞는 DLL 파일을 IEC-Series에 실행파일과 함께 복사합니다.

[표] IEC-Series별 DLL 파일과 다운로드 경로

IEC-Series	IEC266-Series	IEC667-Series	IEC1000-Series
DLL 파일	IEC266HWCON.dll	IEC667HWCON.dll	IEC1000HWCON.dll
다운로드 위치	홈페이지-자료실-SmartX Framework 관련-5. SmartX C++ For DLL 파일		

DLL 파일을 실행 파일 위치에 복사합니다.

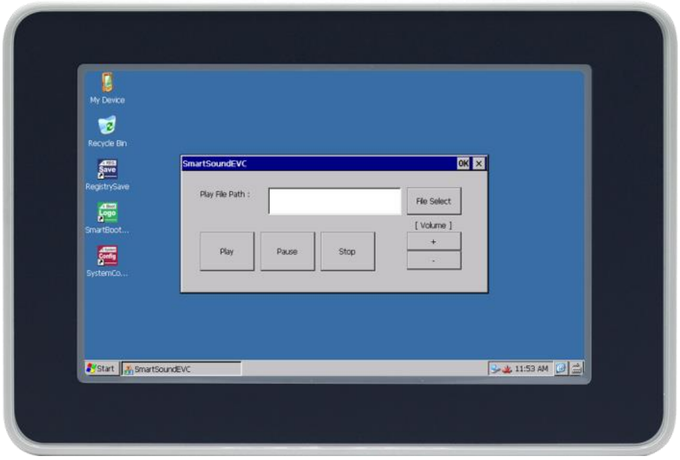


참고 자세한 DLL 파일의 복사 경로는 "DLL 파일 및 복사 경로" 내용을 참고하시기 바랍니다.

4-2) SmartSoundWrap Class 예제 사용하기

SmartSoundWrap Class를 활용한 예제를 제공하고 있으며, 예제파일은 다음 경로에서 다운로드하여 참고하시기 바랍니다.

[예제 파일 다운로드 위치]
홈페이지(www.hnsts.co.kr) → 자료실 → SmartX Framework 예제파일 다운로드 → "SmartSoundEVC"



5. SmartIIC

SmartX Framework는 기본적으로 .Net Compact Framework에서만 지원되도록 설계되어 있지만, 일부 기능을 C++에서 편리하게 사용할 수 있도록 외부 DLL 파일을 제공하고 있습니다. 따라서 SmartIIC를 사용하기 위해서는 반드시 외부 DLL 파일을 실행 파일과 같이 IEC-Series에 배포해야 하며, "1. 인터페이스 함수 선언 → 2. 라이브러리 메모리 로드 처리 → 3. 인터페이스 함수 얻기" 과정을 거쳐야 합니다. 이때 IEC-Series 및 기능별로 로드해야 할 외부 DLL 파일이 다르므로 다음 내용을 확인하시기 바랍니다.

참조 SmartIIC의 자세한 설명은 "홈페이지 → 자료실 → SmartX Framework 관련 → 프로그래밍 가이드 → SmartIIC"를 참조하시기 바랍니다.

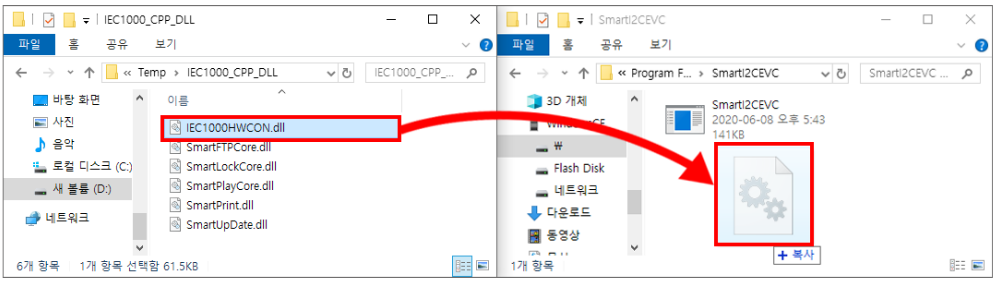
1) DLL 파일 및 복사 경로

HNS 홈페이지에서 사용하는 IEC-Series에 맞는 C++ For DLL 파일을 다운로드 받은 후 압축을 풉니다. 아래 표를 확인하여 DLL 파일을 IEC-Series에 실행 파일과 함께 복사합니다.

[표] IEC-Series별 DLL 파일과 다운로드 경로

IEC-Series	IEC266-Series	IEC667-Series	IEC1000-Series
DLL 파일	IEC266HWCON.dll	IEC667HWCON.dll	IEC1000HWCON.dll
다운로드 위치	홈페이지 → 자료실 → SmartX Framework 관련 → 5. SmartX C++ For DLL 파일		

DLL 파일을 IEC-Series에 복사합니다. DLL 파일을 사용하는 실행 파일이 저장된 폴더에 저장하는 것을 권장하며, 다른 경로에 저장하는 경우 "인터페이스 선언 및 얻기"의 참고 내용을 확인하시기 바랍니다.
※ 개발 PC에서 프로그램을 배포하는 경우 실행 파일은 "Program File" 폴더에 배포됩니다.



2) 인터페이스 선언 및 얻기

인터페이스 사용을 위해 사용할 인터페이스의 선언 후 외부 DLL 파일을 메모리에 로드하여 인터페이스를 얻는 과정을 거쳐야 합니다. 이때 메모리 로드 처리를 위해 LoadLibrary() 함수 호출 시 상단 표의 DLL 파일의 이름을 인자로 설정합니다. 아래 코드는 예시이며, 필요로 하는 인터페이스 함수를 사용하기 위해서는 다음 내용을 참고하시기 바랍니다.

참고 DLL 파일 저장 경로에 따른 LoadLibrary() 함수 인자값

DLL 파일이 실행 파일과 같은 경로에 저장된 경우, LoadLibrary() 함수 인자값을 DLL 파일 이름만으로 설정해 호출할 수 있습니다. 만약 DLL 파일이 실행 파일과 다른 경로에 저장된 경우, LoadLibray() 함수의 인자값에는 저장된 DLL 파일의 전체 경로를 인자값으로 설정해야 합니다.

[C++] 인터페이스 얻기 및 사용하기 예시 코드

```
// 1. 인터페이스 함수 선언
typedef void (*InitEx)();
InitEx      m_pfInit;
// 2. 라이브러리 메모리 로드 처리
HINSTANCE  m_hInst;
// SmartIIC의 인터페이스를 사용하기 위해 IEC1000HWCON.dll 파일을 메모리에 로드
// IEC-Series 및 사용하는 기능별로 외부 DLL 파일이 다르며, 상단 표를 참고하시기 바랍니다.
m_hInst = LoadLibrary(L"IEC1000HWCON.dll");
// 외부 DLL 파일이 다른 경로(Flash Disk\\extdlls)에 저장된 경우
// m_hInst = LoadLibrary(L"Flash Disk\\extdlls\\IEC1000HWCON.dll");
if(m_hInst == NULL)
{
  // 메모리 로드 실패
  return FALSE;
}
// 3. 인터페이스 함수 얻기
m_pfInit = (InitEx)GetProcAddress(m_hInst, _T("Init"));
// 4. 함수 사용
m_pfInit();
```

참고 Wrapping Class 제공안내

사용할 인터페이스를 각각 선언 및 얻어오는 복잡한 과정을 거칠 필요 없이 한 번의 선언과 인스턴싱만으로 편리하게 SmartIIC의 인터페이스를 사용할 수 있도록 Wrapping Class를 제공하고 있습니다.
자세한 내용은 “CSmartIICWrap Class 설명” 내용을 참고하시기 바랍니다.

3) C++ 인터페이스 설명

함수	InitEx

IIC(I2C) 통신을 사용하기 위해서 Speed 및 Port의 IIC(I2C) 관련 기능을 초기화합니다.

◉ `typedef void (*InitEx)()`

C++ 사용법

```
typedef void (*InitEx)();
InitEx              m_pfInit;
HINSTANCE           hInst;
hInst = LoadLibrary(L"IEC1000HWCON.DLL");
if(hInst == NULL)
{
  return;
}
m_pfInit = (InitEx)GetProcAddress(m_hInst, _T("Init"));
m_pfInit();  // Speed 및 Port의 IIC(I2C) 관련 기능을 초기화
FreeLibrary(hInst);
```

함수	ReadByteEx

IIC(I2C) 버스에서 데이터를 수신합니다. 인자로는 ACK와 NOACK가 있으며, ACK인 경우 ACK가 수신될 때까지 기다립니다. 만약 ACK 응답이 없을 경우 SetACKWaitTimeOutEx() 함수에서 설정된 값에 따라 ACK Wait TimeOut 처리됩니다.

◉ `typedef unsigned char (*ReadByteEx)(EACK eACK)`

[인자]

EACK eACK : 데이터 수신 시 ACK 응답을 기다릴지 여부를 설정

- NOACK : ACK 응답을 기다리지 않음

- ACK : ACK 응답을 기다림

[리턴값]

- unsigned char : 수신받은 데이터

C++ 사용법

```
typedef unsigned char (*ReadByteEx)();
HINSTANCE           hInst;
ReadByteEx          pfReadByte;
hInst = LoadLibrary(L"IEC1000HWCON.DLL");
if(hInst == NULL)
{
  return;
}
pfReadByte = (ReadByteEx)GetProcAddress(hInst, _T("ReadByte"));
int iSlaveTemp = pfReadByte(ACK);  // 수신 데이터를 읽고 ACK 응답을 기다림
int iMemData = pfReadByte(NOACK);  // 수신 데이터를 읽고 NOACK 응답을 기다림
FreeLibrary(hInst);
```

함수	SetACKWaitTimeOutEx

IIC(I2C) 통신 중 ReadByteEx() 함수 호출 시 인자값을 ACK로 한 경우, 내부적으로 ACK 응답이 수신될 때까지 기다리게 됩니다. 이러한 경우 만약 상대방 디바이스에서 ACK를 보내지 않게 되면 계속 응답을 기다리게 되어 프로그램이 정상적으로 동작하지 않게 되는 문제가 발생합니다. 이러한 문제를 방지하기 위하여 일정 시간 ACK 응답이 없을 경우 ACK를 SKIP 할 수 있도록 ACK Wait Timeout하는 기능입니다.

◉ typedef void (*SetACKWaitTimeOutEx)(long lInterval)

[인자]

- long lInterval : ACK Wait Timeout 값

C++ 사용법

```
typedef void (*SetACKWaitTimeOutEx)(long lInterval);
HINSTANCE           hInst;
SetACKWaitTimeOut   m_pfSetACKWaitTimeOut;
hInst = LoadLibrary(L"IEC1000HWCON.DLL");
if(hInst == NULL)
{
  return FALSE;
}
m_pfSetACKWaitTimeOut = (SetACKWaitTimeOutEx)GetProcAddress(hInst, _T("SetACKWaitTimeOut"));
m_pfSetACKWaitTimeOut(0xFFFFFFF);  // ACK Wait Timeout 시간 설정
FreeLibrary(hInst);
```

함수	SetSpeedEx

IIC(I2C) 통신 속도와 관련된 설정값입니다. 하나의 단위 신호당 1/1000초로 설정되며, 최대 속도는 90Khz 입니다.

◉ typedef void (*SetSpeedEx)(long lSpeed)

[인자]

- long lSpeed : IIC 통신 속도 (기본값 : 0)

C++ 사용법

```
typedef void (*SetSpeedEx)(long lSpeed);
HINSTANCE           hInst;
SetSpeedEx          m_pfSetSpeed;
hInst = LoadLibrary(L"IEC1000HWCON.DLL");
if(hInst == NULL)
{
  return FALSE;
}
m_pfSetSpeed = (SetSpeedEx)GetProcAddress(hInst, _T("SetSpeed"));
m_pfSetSpeed(0);  // IIC 통신 속도를 설정
FreeLibrary(hInst);
```

함수	StartSignalEx, StopSignalEx

IIC(I2C) 버스 상에 있는 마스터는(IEC-Series) 데이터 송/수신을 시작할 때에 시작 조건을 발생하고 데이터의 송/수신을 끝낼 때는 정지 조건을 발생시킵니다. 시작 조건을 발생시킨 마스터는 정지 조건을 발생할 때까지 버스를 점유하게 되고 이때 다른 마스터는 버스를 사용할 수 없게 됩니다.
SmartIIC의 동작 모드는 마스터 수신 모드와 마스터 송신 모드가 있으며, 각 함수는 SmartIIC 동작 모드의 시작 및 정지 조건 신호를 발생시키는 함수입니다.

- StartSignalEx() : 마스터 송/수신 모드의 시작 조건 신호를 발생시킵니다.
- StopSignalEx() : 마스터 송/수신 모드의 시작 조건 신호를 발생시킵니다.

◉ typedef bool (*StartSignalEx)()

◉ typedef bool (*StopSignalEx)()

[리턴값]

- bool : 시작 및 정지 조건 신호 발생 성공 여부
  - true : 성공
  - false : 실패

C++ 사용법

```
typedef bool (*StartSignalEx)();
typedef bool (*StopSignalEx)();
HINSTANCE hInst;
StartSignalEx                pfStartSignal;
StopSignalEx                 pfStopSignal;
hInst = LoadLibrary(L"IEC1000HWCON.DLL");
if(hInst == NULL)
{
```

```
 return;
}
pfStartSignal = (StartSignalEx)GetProcAddress(hInst, _T("StartSignal"));
pfStopSignal = (StopSignalEx)GetProcAddress(hInst, _T("StopSignal"));
// 시작 및 정지 조건 신호를 발생
pfStartSignal();
pfStopSignal();
FreeLibrary(hInst);
```

함수	WriteByteEx

IIC(I2C) 버스에 Byte 데이터를 송신합니다.

◉ `typedef bool (*WriteByteEx)(unsigned char uchData)`

[인자]

unsigned char uchData : 송신 Byte 데이터

[리턴값]

- bool : 데이터 송신 성공 여부
  - true : 성공
  - false : 실패

C++ 사용법

```
typedef bool (*WriteByteEx)();
HINSTANCE           hInst;
WriteByteEx         pfWriteByte;
hInst = LoadLibrary(L"IEC1000HWCON.DLL");
if(hInst == NULL)
{
  return;
}
pfWriteByte = (WriteByteEx)GetProcAddress(hInst, _T("WriteByte"));
pfWriteByte(0x100);  // 0x100을 송신
FreeLibrary(hInst);
```

4) CSmartIICWrap Class 설명

SmartIICWrap Class는 사용할 인터페이스를 각각 선언 및 얻어오는 복잡한 과정을 거칠 필요 없이 한 번의 선언과 인스턴싱만으로 편리하게 SmartIIC의 기능을 C++(C++, Visual Studio 2005, 2008)에서 지원하도록 Wrapping 한 Class입니다. 본 Class를 사용하기 위해서는 “SmartI2CWrap.h” 파일과 “SmartI2CWrap.cpp” 파일을 대상 프로젝트에 복사한 후 #include "SmartI2CWrap.h"을 호출하여 사용하시기 바랍니다.

인터페이스 및 사용 방식은 위에서 설명한 C++ 인터페이스와는 약간의 차이가 있으므로 다음의 내용을 참고하시어 C++에서 적용하시기 바랍니다.

4-1) SmartIICWrap Class 사용하기

C++에서 SmartIICWrap Class를 사용하기 위해서는 아래와 STEP을 진행해야 합니다.

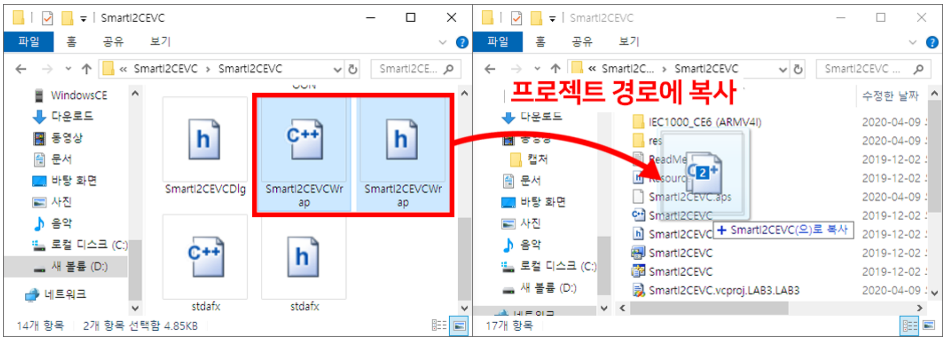
STEP-1 Wrap Class 파일 복사하기

HNS 홈페이지에서 사용하는 IEC-Series에 맞는 예제파일을 다운받은 후 압축을 풉니다.
“SmartI2C EVC” 폴더에서 아래 표를 확인하여 Wrap Class 파일을 프로젝트 경로에 복사합니다.

[표] IEC-Series별 Wrap Class 파일

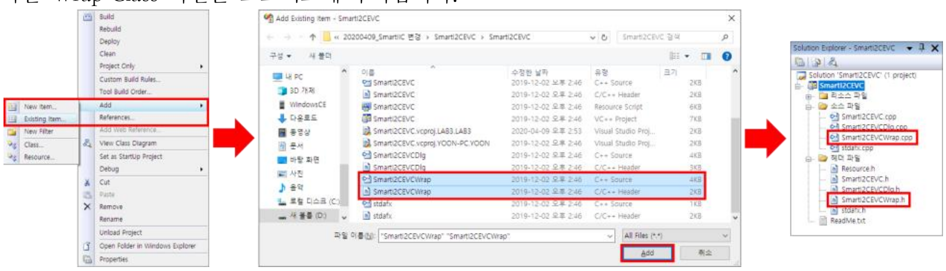
IEC-Series	IEC266-Series	IEC667-Series	IEC1000-Series
Wrap Class 파일	SmartI2CWrap.h, SmartI2CWrap.cpp		
다운로드 위치	홈페이지 → 자료실 → SmartX Framework 관련 → 6. SmartX Framework 예제파일 → Visual C++ 예제파일 → "SmartI2CEVC"		

Wrap Class 파일을 프로젝트 경로에 복사합니다.



STEP-2 Wrap Class 추가하기

복사한 Wrap Class 파일을 프로젝트에 추가합니다.



STEP-3 Wrap Class 사용하기

사용할 소스 파일에 #include<SmartI2CWrap.h> 추가한 후 선언 및 인스턴스하여 사용합니다.

```
#include "SmartI2CEVCWrap.h"
// SmartIIC를 사용하기 위해 변수 선언 및 인스턴싱
CSmartI2CWrap m_SmartI2C;
// SmartIIC 통신을 위해 Port 및 Speed 초기화
m_SmartI2C.Init();
// ACK Wait Timeout 값 설정
m_SmartI2C.SetACKWaitTimeOut = (0xFFFF);
// 데이터 송/수신 전 시작 조건 신호를 발생
m_SmartI2C.StartSignal();
// IIC(I2C) 버스에 Byte 데이터를 송신
m_SmartI2C.WriteByte(0x100);
```

```
// IIC(I2C) 버스에서 ACK 응답을 기다리지 않고 데이터를 수신
int iMemData = m_SmartI2C.ReadByte(NOACK);
// 데이터 송/수신 후 정지 조건 신호를 발생
m_SmartI2C.StopSignal();
```

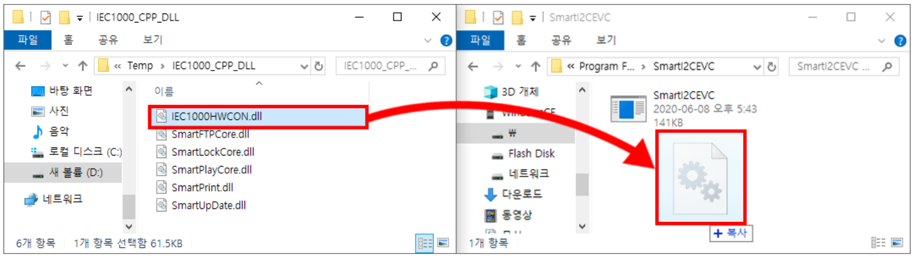
STEP-4 DLL 파일 복사하기

HNS 홈페이지에서 사용하는 IEC-Series에 맞는 예제파일을 다운받은 후 압축을 풉니다. "참조 DLL" 폴더에서 아래 표를 확인하여 사용하는 기능에 맞는 DLL 파일을 IEC-Series에 실행파일과 함께 복사합니다.

[표] IEC-Series별 DLL 파일과 다운로드 경로

IEC-Series	IEC266-Series	IEC667-Series	IEC1000-Series
DLL 파일	IEC266HWCON.dll	IEC667HWCON.dll	IEC1000HWCON.dll
다운로드 위치	홈페이지-자료실-SmartX Framework 관련-5. SmartX C++ For DLL 파일		

DLL 파일을 실행 파일 위치에 복사합니다.



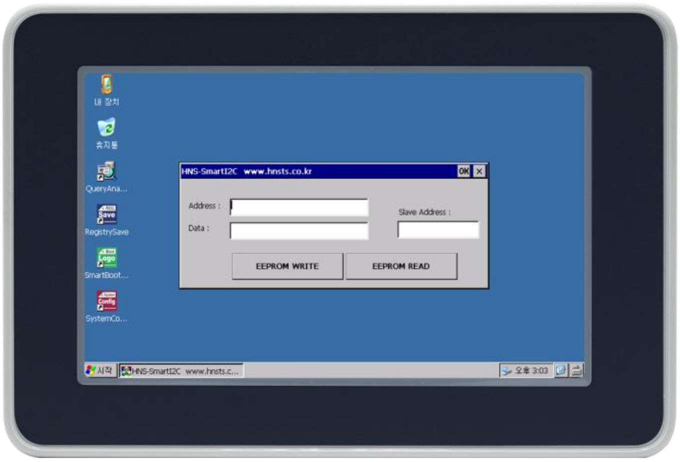
참고 자세한 DLL 파일의 복사 경로는 "DLL 파일 및 복사 경로" 내용을 참고하시기 바랍니다.

4-2) SmartIICWrap Class 예제 사용하기

SmartIICWrap Class를 활용한 예제를 제공하고 있으며, 예제파일은 다음 경로에서 다운로드하여 참고하시기 바랍니다.

[예제 파일 다운로드 위치]

홈페이지(www.hnsts.co.kr) → 자료실 → SmartX Framework 예제파일 다운로드 → “SmartI2CEVC”



6. SmartDAC

SmartX Framework는 기본적으로 .Net Compact Framework에서만 지원되도록 설계되어 있지만, 일부 기능을 C++에서 편리하게 사용할 수 있도록 외부 DLL 파일을 제공하고 있습니다. 따라서 SmartDAC를 사용하기 위해서는 반드시 외부 DLL 파일을 실행 파일과 같이 IEC-Series에 배포해야 하며, "1. 인터페이스 함수 선언 → 2. 라이브러리 메모리 로드 처리 → 3. 인터페이스 함수 얻기" 과정을 거쳐야 합니다. 이때 IEC-Series 및 기능별로 로드해야 할 외부 DLL 파일이 다르므로 다음 내용을 확인하시기 바랍니다.

참조 SmartDAC의 자세한 설명은 "홈페이지 → 자료실 → SmartX Framework 관련 → 프로그래밍 가이드 → SmartDAC"를 참조하시기 바랍니다.

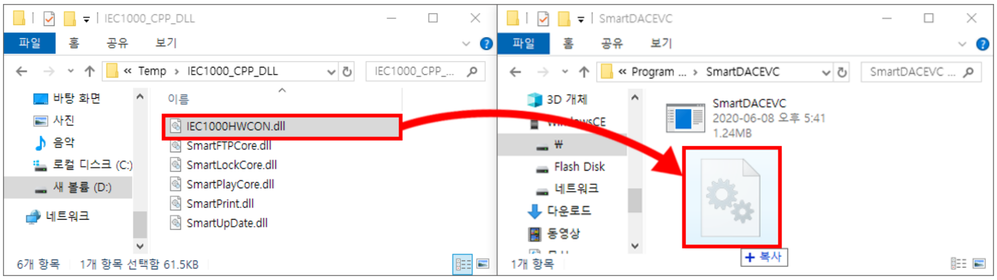
1) DLL 파일 및 복사 경로

HNS 홈페이지에서 사용하는 IEC-Series에 맞는 C++ For DLL 파일을 다운로드 받은 후 압축을 풉니다. 아래 표를 확인하여 DLL 파일을 IEC-Series에 실행 파일과 함께 복사합니다.

[표] IEC-Series별 DLL 파일과 다운로드 경로

IEC-Series	IEC266-Series	IEC667-Series	IEC1000-Series
DLL 파일	IEC266HWCON.dll	IEC667HWCON.dll	IEC1000HWCON.dll
다운로드 위치	홈페이지 → 자료실 → SmartX Framework 관련 → 5. SmartX C++ For DLL 파일		

DLL 파일을 IEC-Series에 복사합니다. DLL 파일을 사용하는 실행 파일이 저장된 폴더에 저장하는 것을 권장하며, 다른 경로에 저장하는 경우 "인터페이스 선언 및 얻기"의 참고 내용을 확인하시기 바랍니다.
※ 개발 PC에서 프로그램을 배포하는 경우 실행 파일은 "Program File" 폴더에 배포됩니다.



2) 인터페이스 선언 및 얻기

인터페이스 사용을 위해 사용할 인터페이스의 선언 후 외부 DLL 파일을 메모리에 로드하여 인터페이스를 얻는 과정을 거쳐야 합니다. 이때 메모리 로드 처리를 위해 LoadLibrary() 함수 호출 시 상단 표의 DLL 파일의 이름을 인자로 설정합니다. 아래 코드는 예시이며, 필요로 하는 인터페이스 함수를 사용하기 위해서는 다음 내용을 참고하시기 바랍니다.

참고 DLL 파일 저장 경로에 따른 LoadLibrary() 함수 인자값

DLL 파일이 실행 파일과 같은 경로에 저장된 경우, LoadLibrary() 함수 인자값을 DLL 파일 이름만으로 설정해 호출할 수 있습니다. 만약 DLL 파일이 실행 파일과 다른 경로에 저장된 경우, LoadLibray() 함수의 인자값에는 저장된 DLL 파일의 전체 경로를 인자값으로 설정해야 합니다.

[C++] 인터페이스 얻기 및 사용하기 예시 코드

```
// 1. 인터페이스 함수 선언
typedef void (*SetOutData1Ex)(int iDataOut1)
SetOutData1Ex  m_pfSetOutData1;
// 2. 라이브러리 메모리 로드 처리
HINSTANCE  m_hInst;
// SmartDAC의 인터페이스를 사용하기 위해 IEC1000HWCON.dll 파일을 메모리에 로드
// IEC-Series 및 사용하는 기능별로 외부 DLL 파일이 다르며, 상단 표를 참고하시기 바랍니다.
m_hInst = LoadLibrary(L"IEC1000HWCON.dll");
// 외부 DLL 파일이 다른 경로(Flash Disk\\extdlls)에 저장된 경우
// m_hInst = LoadLibrary(L"Flash Disk\\extdlls\\IEC1000HWCON.dll");
if(m_hInst == NULL)
{
  // 메모리 로드 실패
  return FALSE;
}
// 3. 인터페이스 함수 얻기
m_pfSetOutData1 = (SetOutData1Ex)GetProcAddress(m_hInst, _T("SetOutData1"));
// 4. 함수 사용
m_pfSetOutData1(2048);
```

참고 Wrapping Class 제공안내

사용할 인터페이스를 각각 선언 및 얻어오는 복잡한 과정을 거칠 필요 없이 한 번의 선언과 인스턴턴싱만으로 편리하게 SmartDAC의 인터페이스를 사용할 수 있도록 Wrapping Class를 제공하고 있습니다.
자세한 내용은 "CSmartDACWrap Class 설명" 내용을 참고하시기 바랍니다.

3) C++ 인터페이스 설명

함수	SetOutData1Ex, SetOutData2Ex

각각의 DAC 채널에 전압의 출력을 제어합니다.

◉ typedef void (*SetOutData1Ex)(int iDataOut1)

◉ typedef void (*SetOutData2Ex)(int iDataOut2)

[인자]

• int iDataOut1, iDataOut2 : DAC 채널의 값을 설정 (0~4095)

C++ 사용법

```
typedef void (*SetOutData1Ex)(int iDataOut1);
typedef void (*SetOutData2Ex)(int iDataOut2);
SetOutData1Ex        m_pfSetOutData1;
SetOutData2Ex        m_pfSetOutData2;
HINSTANCE            m_hInst;
m_hInst = LoadLibrary(L"IEC1000HWCON.DLL");
if(m_hInst == NULL)
{
  return FALSE;
}
m_pfSetOutData1 = (SetOutData1Ex)GetProcAddress(m_hInst, _T("SetOutData1"));
m_pfSetOutData2 = (SetOutData2Ex)GetProcAddress(m_hInst, _T("SetOutData2"));
m_pfSetOutData1(2048);  // DAC 채널1로 2048에 해당하는 전압 출력
m_pfSetOutData2(2048);  // DAC 채널2로 2048에 해당하는 전압 출력
FreeLibrary(hInst);
```

4) CSmartDACWrap Class 설명

SmartDACWrap Class는 사용할 인터페이스를 각각 선언 및 얻어오는 복잡한 과정을 거칠 필요 없이 한 번의 선언과 인스턴싱만으로 편리하게 SmartDAC의 기능을 C++(C++, Visual Studio 2005, 2008)에서 지원하도록 Wrapping 한 Class입니다. 본 Class를 사용하기 위해서는 “SmartDACWrap.h” 파일과 “SmartDACWrap.cpp” 파일을 대상 프로젝트에 복사한 후 #include "SmartDACWrap.h"을 호출하여 사용하시기 바랍니다.

인터페이스 및 사용 방식은 위에서 설명한 C++ 인터페이스와는 약간의 차이가 있으므로 다음의 내용을 참고하시어 C++에서 적용하시기 바랍니다.

4-1) SmartDACWrap Class 사용하기

C++에서 SmartDACWrap Class를 사용하기 위해서는 아래와 STEP을 진행해야 합니다.

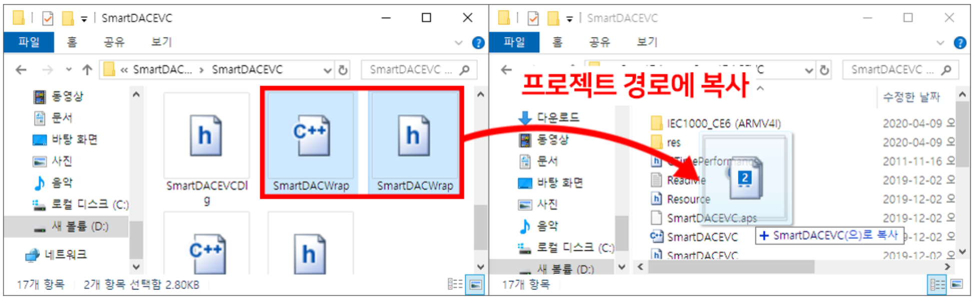
STEP-1	Wrap Class 파일 복사하기

HNS 홈페이지에서 사용하는 IEC-Series에 맞는 예제파일을 다운받은 후 압축을 풉니다.
“SmartDACEVC” 폴더에서 아래 표를 확인하여 Wrap Class 파일을 프로젝트 경로에 복사합니다.

[표] IEC-Series별 Wrap Class 파일

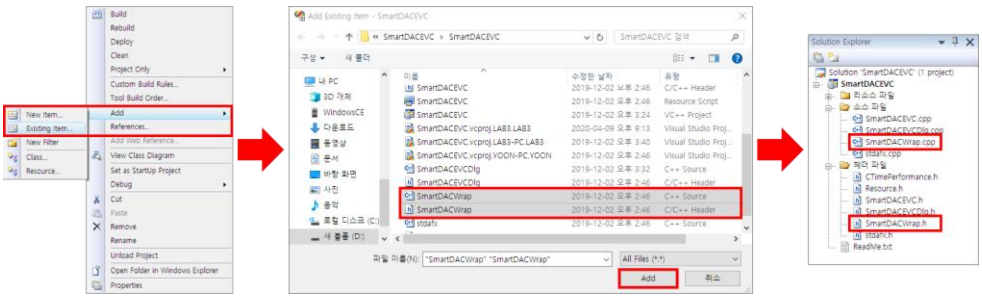
IEC-Series	IEC266-Series	IEC667-Series	IEC1000-Series
Wrap Class 파일	SmartDACWrap.h, SmartDACWrap.cpp		
다운로드 위치	홈페이지 → 자료실 → SmartX Framework 관련 → 6. SmartX Framework 예제파일 → Visual C++ 예제파일 → "SmartDACEVC"		

Wrap Class 파일을 프로젝트 경로에 복사합니다.



STEP-2	Wrap Class 추가하기

복사한 Wrap Class 파일을 프로젝트에 추가합니다.



STEP-3	Wrap Class 사용하기

사용할 소스 파일에 #include<SmartDACWrap.h> 추가한 후 선언 및 인스턴스하여 사용합니다.

```
#include "SmartDACWrap.h"
// SmartDAC를 사용하기 위해 변수 선언 및 인스턴싱
CSmartDACWrap m_SmartDAC;
// DAC 채널1로 2048에 해당하는 전압 출력
m_SmartDAC.SetOutData1(2048);
```

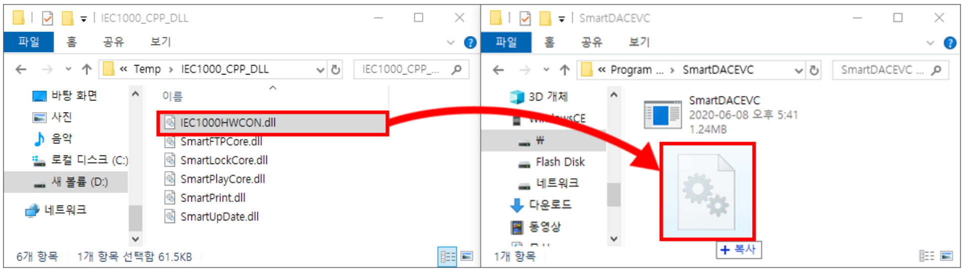
STEP-4	DLL 파일 복사하기

HNS 홈페이지에서 사용하는 IEC-Series에 맞는 예제파일을 다운받은 후 압축을 풉니다. "참조 DLL" 폴더에서 아래 표를 확인하여 사용하는 기능에 맞는 DLL 파일을 IEC-Series에 실행파일과 함께 복사합니다.

[표] IEC-Series별 DLL 파일과 다운로드 경로

IEC-Series	IEC266-Series	IEC667-Series	IEC1000-Series
DLL 파일	IEC266HWCON.dll	IEC667HWCON.dll	IEC1000HWCON.dll
다운로드 위치	홈페이지 → 자료실 → SmartX Framework 관련 → 5. SmartX C++ For DLL 파일		

DLL 파일을 실행 파일 위치에 복사합니다.



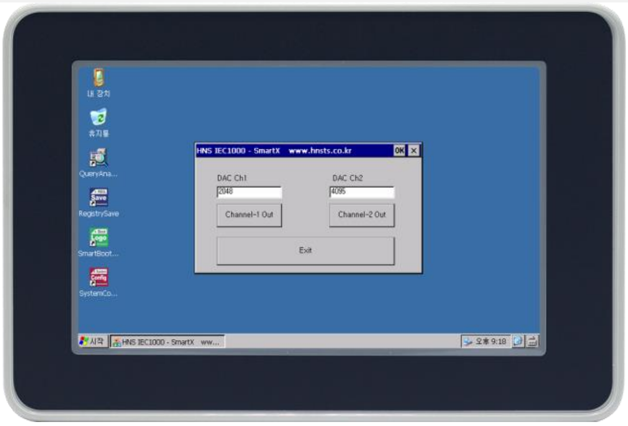
참고	자세한 DLL 파일의 복사 경로는 "DLL 파일 및 복사 경로" 내용을 참고하시기 바랍니다.

4-2) SmartDACWrap Class 예제 사용하기

SmartDACWrap Class를 활용한 예제를 제공하고 있으며, 예제파일은 다음 경로에서 다운로드하여 참고하시기 바랍니다.

[예제 파일 다운로드 위치]

홈페이지(www.hnsts.co.kr) → 자료실 → SmartX Framework 예제파일 다운로드 → “SmartDACEVC”



7. SmartBattery

SmartX Framework는 기본적으로 .Net Compact Framework에서만 지원되도록 설계되어 있지만, 일부 기능을 C++에서 편리하게 사용할 수 있도록 외부 DLL 파일을 제공하고 있습니다. 따라서 SmartBattery를 사용하기 위해서는 반드시 외부 DLL 파일을 실행 파일과 같이 IEC-Series에 배포해야 하며, "1. 인터페이스 함수 선언 → 2. 라이브러리 메모리 로드 처리 → 3. 인터페이스 함수 얻기" 과정을 거쳐야 합니다. 이때 IEC-Series 및 기능별로 로드해야 할 외부 DLL 파일이 다르므로 다음 내용을 확인하시기 바랍니다.

참조 SmartBattery의 자세한 설명은 "홈페이지 → 자료실 → SmartX Framework 관련 → 프로그래밍 가이드 → SmartBattery"를 참조하시기 바랍니다.

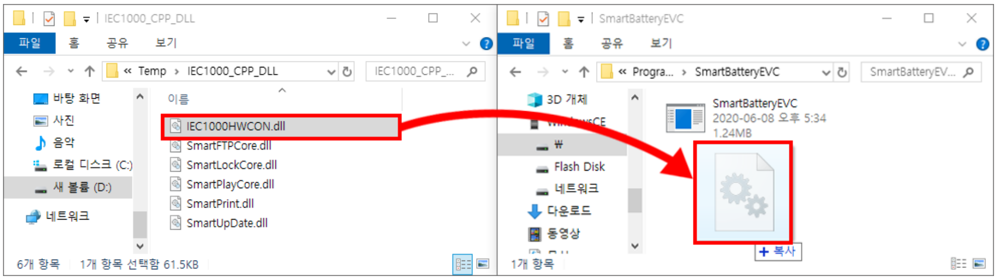
1) DLL 파일 및 복사 경로

HNS 홈페이지에서 사용하는 IEC-Series에 맞는 C++ For DLL 파일을 다운로드 받은 후 압축을 풉니다. 아래 표를 확인하여 DLL 파일을 IEC-Series에 실행 파일과 함께 복사합니다.

[표] IEC-Series별 DLL 파일과 다운로드 경로

IEC-Series	IEC266-Series	IEC667-Series	IEC1000-Series
DLL 파일	IEC266HWCON.dll	IEC667HWCON.dll	IEC1000HWCON.dll
다운로드 위치	홈페이지 → 자료실 → SmartX Framework 관련 → 5. SmartX C++ For DLL 파일		

DLL 파일을 IEC-Series에 복사합니다. DLL 파일을 사용하는 실행 파일이 저장된 폴더에 저장하는 것을 권장하며, 다른 경로에 저장하는 경우 "인터페이스 선언 및 얻기"의 참고 내용을 확인하시기 바랍니다.
※ 개발 PC에서 프로그램을 배포하는 경우 실행 파일은 "Program File" 폴더에 배포됩니다.



2) 인터페이스 선언 및 얻기

인터페이스 사용을 위해 사용할 인터페이스의 선언 후 외부 DLL 파일을 메모리에 로드하여 인터페이스를 얻는 과정을 거쳐야 합니다. 이때 메모리 로드 처리를 위해 LoadLibrary() 함수 호출 시 상단 표의 DLL 파일의 이름을 인자로 설정합니다. 아래 코드는 예시이며, 필요로 하는 인터페이스 함수를 사용하기 위해서는 다음 내용을 참고하시기 바랍니다.

참고 DLL 파일 저장 경로에 따른 LoadLibrary() 함수 인자값

DLL 파일이 실행 파일과 같은 경로에 저장된 경우, LoadLibrary() 함수 인자값을 DLL 파일 이름만으로 설정해 호출할 수 있습니다. 만약 DLL 파일이 실행 파일과 다른 경로에 저장된 경우, LoadLibray() 함수의 인자값에는 저장된 DLL 파일의 전체 경로를 인자값으로 설정해야 합니다.

[C++] 인터페이스 얻기 및 사용하기 예시 코드

```
// 1. 인터페이스 함수 선언
typedef void (*PowerOFFEx)()
PowerOFFEx  m_pfPowerOFF;
// 2. 라이브러리 메모리 로드 처리
HINSTANCE  m_hInst;
// SmartBattery의 인터페이스를 사용하기 위해 IEC1000HWCON.dll 파일을 메모리에 로드
// IEC-Series 및 사용하는 기능별로 외부 DLL 파일이 다르며, 상단 표를 참고하시기 바랍니다.
m_hInst = LoadLibrary(L"IEC1000HWCON.dll");
// 외부 DLL 파일이 다른 경로(Flash Disk\\extdlls)에 저장된 경우
// m_hInst = LoadLibrary(L"Flash Disk\\extdlls\\IEC1000HWCON.dll");
if(m_hInst == NULL)
{
  // 메모리 로드 실패
  return FALSE;
}
// 3. 인터페이스 함수 얻기
m_pfPowerOFF = (PowerOFFEx)GetProcAddress(m_hInst, _T("PowerOFF"));
// 4. 함수 사용
m_pfPowerOFF();
```

참고 Wrapping Class 제공안내

사용할 인터페이스를 각각 선언 및 얻어오는 복잡한 과정을 거칠 필요 없이 한 번의 선언과 인스턴싱만으로 편리하게 SmartBattery의 인터페이스를 사용할 수 있도록 Wrapping Class를 제공하고 있습니다.
자세한 내용은 "CSmartBatteryWrap Class 설명" 내용을 참고하시기 바랍니다.

3) C++ 인터페이스 설명

함수	GetBatteryPercentEx

현재 Battery의 남은 용량의 비율을 얻습니다.

◉ typedef int (*GetBatteryPercentEx)()

[리턴값]

- int : 현재 Battery의 남은 용량 (단위 : %)

C++ 사용법

```
typedef int (*GetBatteryPercentEx)()
GetBatteryPercentEx            m_pfGetBatteryPercent;
HINSTANCE                      hInst;
hInst = LoadLibrary(L"IEC1000HWCON.DLL");
if(hInst == NULL)
{
  return;
}
m_pfGetBatteryPercent = (GetBatteryPercentEx)GetProcAddress(m_hInst, _T("GetBatteryPercent"));
CString strTemp;
strTemp.Format(L"%d[%%]", m_pfGetBatteryPercent());  // 현재 Battery의 남은 용량의 비율을 저장
FreeLibrary(hInst);
```

함수 GetBatteryVoltageEx

현재 Battery의 전압값을 얻습니다.

◉ typedef double (*GetBatteryVoltageEx)()

[리턴값]

- double : 현재 Battery의 전압 값 (단위 : Volt)

C++ 사용법

```
typedef double (*GetBatteryVoltageEx)()
GetBatteryVoltageEx          m_pfGetBatteryVoltage;
HINSTANCE                    hInst;
hInst = LoadLibrary(L"IEC1000HWCON.DLL");
if(hInst == NULL)
{
  return;
}
m_pfGetBatteryVoltage = (GetBatteryVoltageEx)GetProcAddress(m_hInst, _T("GetBatteryVoltage"));
CString strTemp;
strTemp.Format(L"%d[%%]", m_pfGetBatteryVoltage());  // 현재 Battery의 남은 용량을 저장
FreeLibrary(hInst);
```

함수 IsExtDCPowerINEx

외부 DC 전원의 공급 상태를 확인합니다. 즉, Battery의 충전을 위하여 외부 DC 전원이 공급되는지 여부를 확인 할 수 있습니다.

◉ typedef bool (*IsExtDCPowerINEx)()

[리턴값]

- bool : 외부 DC 전원 공급 여부
  - true : 외부 DC 전원이 공급됨
  - false : 외부 DC 전원이 공급되지 않음 (내부 Battery로 전원 공급됨)

C++ 사용법

```
typedef bool (*IsExtDCPowerINEx)()
IsExtDCPowerINEx             m_pfIsExtDCPowerIN;
HINSTANCE                    hInst;
hInst = LoadLibrary(L"IEC1000HWCON.DLL");
if(hInst == NULL)
{
  return;
}
m_pfIsExtDCPowerIN = (IsExtDCPowerINEx)GetProcAddress(m_hInst, _T("IsExtDCPowerIN"));
// 현재 외부 DC 전원이 공급 되는지 확인
if(m_pfIsExtDCPowerIN() == true)
{
  // 현재 Batery로만 동작 중
}
else
```

```
{
  // 현재 Battery가 충전 중이며 외부 DC 전원이 공급되는 상태
}
FreeLibrary(hInst);
```

함수	PowerOFFEx

시스템 전원을 OFF(끄기)합니다. 함수 호출 후 2초 후에 전원이 OFF됩니다.

주의	1. 시스템의 전원을 OFF 하기 앞서 중요한 데이터는 BackUp 하시기 바라며, 만약 파일을 쓰고 있다면 파일을 닫아 데이터의 손상을 방지하시기 바랍니다. 2. SmartBattery Board의 J7 점퍼가 OFF 되어 있어야 전원이 OFF됩니다.

◉ typedef void (*PowerOFFEx)()

C++ 사용법

```
typedef void (*PowerOFFEx)()
PowerOFFEx          m_pfPowerOFF;
HINSTANCE           hInst;
hInst = LoadLibrary(L"IEC1000HWCON.DLL");
if(hInst == NULL)
{
  return;
}
m_pfPowerOFF = (PowerOFFEx)GetProcAddress(m_hInst, _T("PowerOFF"));
m_pfPowerOFF();  // 전원을 OFF합니다. 2초 후 전원 OFF
FreeLibrary(hInst);
```

함수	SetBatteryVoltageLimitEx

Battery의 완충 전압과 최소 전압을 설정합니다.

주의	SmartBattery Board는 Battery의 전압이 8.2V가 되면 완충으로 인식하도록 설계되어 있습니다. 따라서 fMaxVoltage 인자값을 변경하지 마시고 기본값(8.2)로 사용하시기 바랍니다.

◉ typedef void (*SetBatteryVoltageLimitEx)(double fMaxVoltage, double fMinVoltage)

[인자]

- double fMaxVoltage : Battery의 완충 전압 (단위 : Volt, 기본값 : 8.2)
- double fMinVoltage : IEC-Series의 동작 가능한 Battery의 최소 전압 (단위 : Volt, 기본값 : 6.5)

C++ 사용법

```
typedef void (*SetBatteryVoltageLimitEx)(double fMaxVoltage, double fMinVoltage);
SetBatteryVoltageLimitEx       m_pfSetBatteryVoltageLimit;
HINSTANCE                      hInst;
hInst = LoadLibrary(L"IEC1000HWCON.DLL");
if(hInst == NULL)
{
  return;
}
m_pfSetBatteryVoltageLimit = (SetBatteryVoltageLimitEx)GetProcAddress(m_hInst,
                                                     _T("SetBatteryVoltageLimit"));
```

```
m_pfSetBatteryVoltageLimit(8.2, 6.3);  // 완충 전압 8.2V, 최소 전압 6.3V로 설정
FreeLibrary(hInst);
```

함수	SetExtDCPowerInOffsetVoltageEx

Battery를 충전하기 위해 외부 DC 전원을 공급하게 되면 인가되는 전압과 현재 Battery의 전압의 차이(Offset 전압)가 발생하게 되며, 이 전압의 차이를 설정합니다. Offset 전압은 Battery의 용량 및 제조 업체에 따라 달라지며 정확한 사용을 위해서는 다음과 같은 방법으로 설정하시기 바랍니다.

중요 Offset 전압 측정 방법

1. 우선 외부 DC 전원을 연결하지 않은 상태에서 테스터기로 Battery의 양극 전압을 측정합니다.
2. 다음은 외부 DC 전원을 연결한 상태에서 테스터기로 Battery의 양극 전압을 측정합니다.
3. 측정된 두 값의 차이가 Offset 전압값입니다. 본 측정을 보다 정확하게 하기 위해서는 위 측정을 배터리의 전압량의 20%대, 50%대 80%대 이렇게 3개의 값을 평균하여 적용하시면 오차를 최소화하실 수 있습니다.

권장 위 측정 및 테스트는 SmartBattery 예제 프로그램을 활용하시길 권장합니다.

◉ typedef void (*SetExtDCPowerInOffsetVoltageEx)(double fOffsetVoltage)

[인자]

- double : Offset 전압 (단위 : Volt)

C++ 사용법

```
typedef void (*SetExtDCPowerInOffsetVoltageEx)(double fOffsetVoltage);
SetExtDCPowerInOffsetVoltageEx  m_pfSetExtDCPowerInOffsetVoltage;
HINSTANCE                       hInst;
hInst = LoadLibrary(L"IEC1000HWCON.DLL");
if(hInst == NULL)
{
  return;
}
m_pfSetExtDCPowerInOffsetVoltage = (SetExtDCPowerInOffsetVoltageEx)GetProcAddress(m_hInst,
                                             _T("SetExtDCPowerInOffsetVoltage"));
m_pfSetExtDCPowerInOffsetVoltage(0.23); // Offset 전압을 0.23V로 설정
FreeLibrary(hInst);
```

4) CSmartBatteryWrap Class 설명

SmartBatteryWrap Class는 사용할 인터페이스를 각각 선언 및 얻어오는 복잡한 과정을 거칠 필요 없이 한 번의 선언과 인스턴싱만으로 편리하게 SmartBattery의 기능을 C++(C++, Visual Studio 2005, 2008)에서 지원하도록 Wrapping 한 Class입니다. 본 Class를 사용하기 위해서는 “SmartBatteryWrap.h” 파일과 “SmartBatteryWrap.cpp” 파일을 대상 프로젝트에 복사한 후 #include "SmartBatteryWrap.h"을 호출하여 사용하시기 바랍니다.

인터페이스 및 사용 방식은 위에서 설명한 C++ 인터페이스와는 약간의 차이가 있으므로 다음의 내용을 참고하시어 C++에서 적용하시기 바랍니다.

4-1) SmartBatteryWrap Class 사용하기

C++에서 SmartBatteryWrap Class를 사용하기 위해서는 아래와 STEP을 진행해야 합니다.

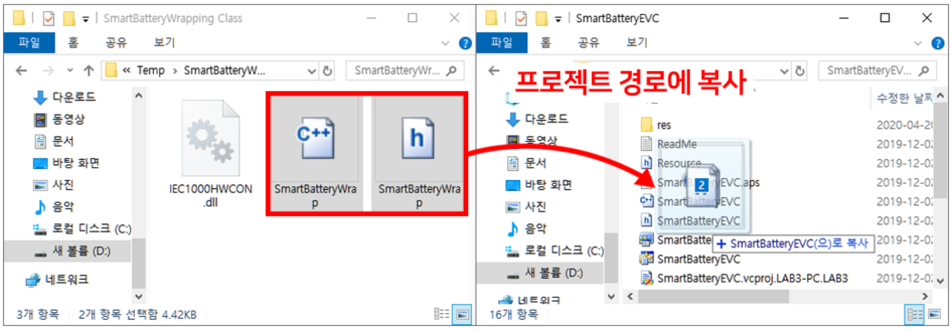
STEP-1 Wrap Class 파일 복사하기

HNS 홈페이지에서 사용하는 IEC-Series에 맞는 예제파일을 다운받은 후 압축을 풉니다.
“SmartBattery EVC” 폴더에서 아래 표를 확인하여 Wrap Class 파일을 프로젝트 경로에 복사합니다.

[표] IEC-Series별 Wrap Class 파일

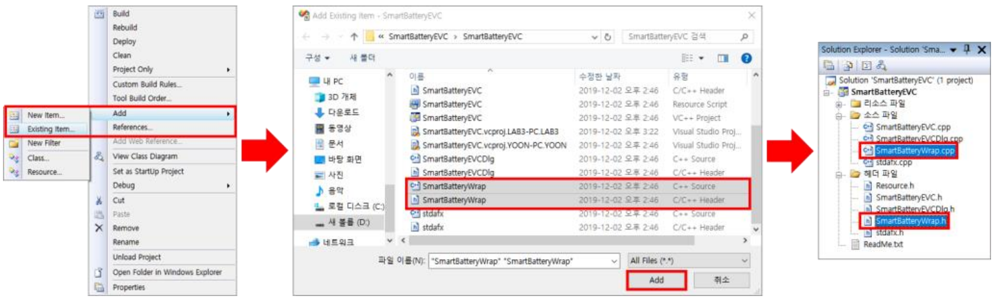
IEC-Series	IEC266-Series	IEC667-Series	IEC1000-Series
Wrap Class 파일	SmartBatteryWrap.h, SmartBatteryWrap.cpp		
다운로드 위치	홈페이지 → 자료실 → SmartX Framework 관련 → 6. SmartX Framework 예제파일 → Visual C++ 예제파일 → "SmartBatteryEVC"		

Wrap Class 파일을 프로젝트 경로에 복사합니다.



STEP-2 Wrap Class 추가하기

복사한 Wrap Class 파일을 프로젝트에 추가합니다.



STEP-3 Wrap Class 사용하기

사용할 소스 파일에 #include<SmartBatteryWrap.h> 추가한 후 선언 및 인스턴스하여 사용합니다.

```
#include "SmartBatteryWrap.h"
// SmartBattery를 사용하기 위해 변수 선언 및 인스턴싱
CSmartBatteryWrap m_SmartBattery;
// 현재 Battery 전압 값 저장
CString strTemp1;
strTemp1.Format(L"%.3f[V]", m_SmartBattery.GetBatteryVoltage());
// 현재 Battery의 충전량을 백분율을 저장
CString strTemp2;
strTemp2.Format(L"%d[%%]", m_SmartBattery.GetBatteryPercent());
// 완충 전압 8.2V, 최소 전압 6.3V로 설정
```

```
m_SmartBattery.SetBatteryVoltageLimit(8.2, 6.3);
// Offset 전압을 0.23V로 설정
m_SmartBattery.SetExtDCPowerInOffsetVoltage(0.23);
// 전원을 OFF합니다. 2초 후 전원 OFF
m_SmartBattery.PowerOFF();
```

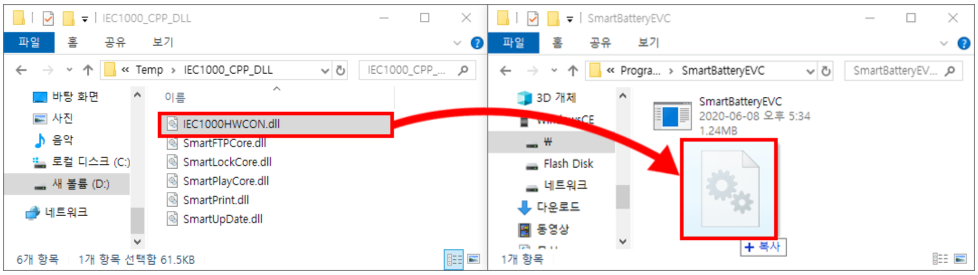
STEP-4 DLL 파일 복사하기

HNS 홈페이지에서 사용하는 IEC-Series에 맞는 예제파일을 다운받은 후 압축을 풉니다. "참조 DLL" 폴더에서 아래 표를 확인하여 사용하는 기능에 맞는 DLL 파일을 IEC-Series에 실행파일과 함께 복사합니다.

[표] IEC-Series별 DLL 파일과 다운로드 경로

IEC-Series	IEC266-Series	IEC667-Series	IEC1000-Series
DLL 파일	IEC266HWCON.dll	IEC667HWCON.dll	IEC1000HWCON.dll
다운로드 위치	홈페이지 → 자료실 → SmartX Framework 관련 → 5. SmartX C++ For DLL 파일		

DLL 파일을 실행 파일 위치에 복사합니다.



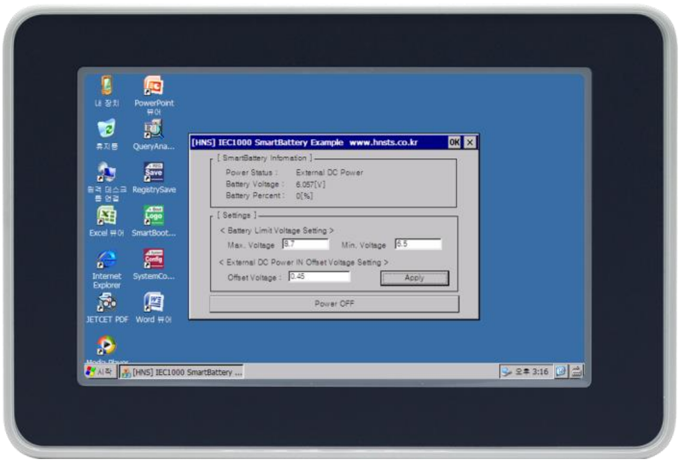
참고 자세한 DLL 파일의 복사 경로는 "DLL 파일 및 복사 경로" 내용을 참고하시기 바랍니다.

4-2) SmartBatteryWrap Class 예제 사용하기

SmartBatteryWrap Class를 활용한 예제를 제공하고 있으며, 예제파일은 다음 경로에서 다운로드하여 참고하시기 바랍니다.

[예제 파일 다운로드 위치]

홈페이지(www.hnsts.co.kr) → 자료실 → SmartX Framework 예제파일 다운로드 → "SmartBatteryEVC"



8. SmartPrint

SmartX Framework는 기본적으로 .Net Compact Framework에서만 지원되도록 설계되어 있지만, 일부 기능을 C++에서 편리하게 사용할 수 있도록 외부 DLL 파일을 제공하고 있습니다. 따라서 SmartPrint를 사용하기 위해서는 반드시 외부 DLL 파일을 실행 파일과 같이 IEC-Series에 배포해야 하며, "1. 인터페이스 함수 선언 → 2. 라이브러리 메모리 로드 처리 → 3. 인터페이스 함수 얻기" 과정을 거쳐야 합니다. 이때 IEC-Series 및 기능별로 로드해야 할 외부 DLL 파일이 다르므로 다음 내용을 확인하시기 바랍니다.

참조 SmartPrint의 자세한 설명은 "홈페이지 → 자료실 → SmartX Framework 관련 → 프로그래밍 가이드 → SmartPrint"를 참조하시기 바랍니다.

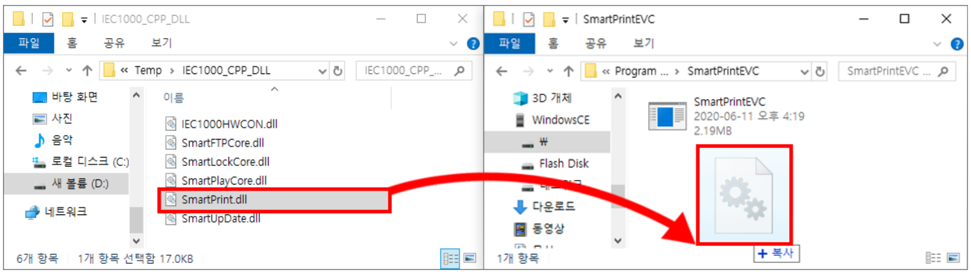
1) DLL 파일 및 복사 경로

HNS 홈페이지에서 사용하는 IEC-Series에 맞는 C++ For DLL 파일을 다운로드 받은 후 압축을 풉니다. 아래 표를 확인하여 DLL 파일을 IEC-Series에 실행 파일과 함께 복사합니다.

[표] IEC-Series별 DLL 파일과 다운로드 경로

IEC-Series	IEC266-Series	IEC667-Series	IEC1000-Series
DLL 파일	SmartPrint.dll		
다운로드 위치	홈페이지 → 자료실 → SmartX Framework 관련 → 5. SmartX C++ For DLL 파일		

DLL 파일을 IEC-Series에 복사합니다. DLL 파일을 사용하는 실행 파일이 저장된 폴더에 저장하는 것을 권장하며, 다른 경로에 저장하는 경우 "인터페이스 선언 및 얻기"의 참고 내용을 확인하시기 바랍니다.
※ 개발 PC에서 프로그램을 배포하는 경우 실행 파일은 "Program File" 폴더에 배포됩니다.



2) 인터페이스 선언 및 얻기

인터페이스 사용을 위해 사용할 인터페이스의 선언 후 외부 DLL 파일을 메모리에 로드하여 인터페이스를 얻는 과정을 거쳐야 합니다. 이때 메모리 로드 처리를 위해 LoadLibrary() 함수 호출 시 상단 표의 DLL 파일의 이름을 인자로 설정합니다. 아래 코드는 예시이며, 필요로 하는 인터페이스 함수를 사용하기 위해서는 다음 내용을 참고하시기 바랍니다.

참고 DLL 파일 저장 경로에 따른 LoadLibrary() 함수 인자값

DLL 파일이 실행 파일과 같은 경로에 저장된 경우, LoadLibrary() 함수 인자값을 DLL 파일 이름만으로 설정해 호출할 수 있습니다. 만약 DLL 파일이 실행 파일과 다른 경로에 저장된 경우, LoadLibray() 함수의 인자값에는 저장된 DLL 파일의 전체 경로를 인자값으로 설정해야 합니다.

[C++] 인터페이스 얻기 및 사용하기 예시 코드

```
// 1. 인터페이스 함수 선언
typedef void (*PrintOutEx)()
PrintOutEx  m_pfPrintOut;
// 2. 라이브러리 메모리 로드 처리
HINSTANCE  m_hInst;
// SmartPrint의 인터페이스를 사용하기 위해 SmartPrint.dll 파일을 메모리에 로드
// IEC-Series 및 사용하는 기능별로 외부 DLL 파일이 다르며, 상단 표를 참고하시기 바랍니다.
m_hInst = LoadLibrary(L"SmartPrint.dll");
// 외부 DLL 파일이 다른 경로(Flash Disk\\extdlls)에 저장된 경우
// m_hInst = LoadLibrary(L"Flash Disk\\extdlls\\SmartPrint.dll");
if(m_hInst == NULL)
{
  // 메모리 로드 실패
  return FALSE;
}
// 3. 인터페이스 함수 얻기
m_pfPrintOut = (PrintOutEx)GetProcAddress(m_hInst, _T("PrintOut"));
// 4. 함수 사용
m_pfPrintOut();
```

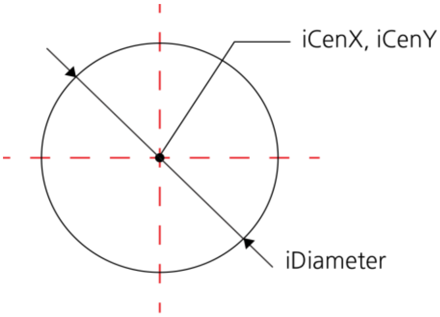
참고 Wrapping Class 제공안내

사용할 인터페이스를 각각 선언 및 얻어오는 복잡한 과정을 거칠 필요 없이 한 번의 선언과 인스턴싱만으로 편리하게 SmartPrint의 인터페이스를 사용할 수 있도록 Wrapping Class를 제공하고 있습니다.
자세한 내용은 "CSmartPrintWrap Class 설명" 내용을 참고하시기 바랍니다.

3) C++ 인터페이스 설명

함수	CircleEx

프린터 버퍼에 원을 그립니다. 선의 색상 및 두께는 SetPenStyleEx() 함수로 설정하며, 내부 채움은 SetBrushStyleEx() 함수로 설정합니다.



◉ `typedef void (*CircleEx)(int iCenX, int iCenY, int iDiameter)`

[인자]

- `int iCenX` : 원 중심의 X좌표
- `int iCenY` : 원 중심의 Y좌표
- `int iDiameter` : 원의 지름

C++ 사용법

```
typedef bool (*CircleEx)(int iCenX, int iCenY, int iDiameter);
CircleEx          m_pfCircle;
HINSTANCE         hInst;
```

```
hInst = LoadLibrary(L"SmartPrint.dll");
if(hInst == NULL)
{
  return;
}
m_pfCircle = (CircleEx)GetProcAddress(m_hInst, _T("Circle"));
m_pfCircle(300, 1400, 170);  // 원을 그린다.
FreeLibrary(hInst);
```

함수	DisplayToPrint1Ex, DisplayToPrint2Ex

현재 장치(IEC)의 LCD화면에 출력된 상태를 그대로 프린터로 출력합니다.

참조	"홈페이지(www.hnsts.co.kr) → 자료실 → SmartX Framework 관련 → 프로그래밍 가이드 → SmartPrint → DisplayToPrint() 메소드"를 참조하시기 바랍니다.

◉ typedef void (*DisplayToPrint1Ex)(int iXpos, int iYpos, int iWidth, int iHeight)

◉ typedef void (*DisplayToPrint2Ex)(int iXpos, int iYpos)

[인자]

- int iXpos : 출력될 이미지 좌측 상단의 X좌표 (필수)
- int iYpos : 출력될 이미지 좌측 상단의 Y좌표 (필수)
- int iWidth : 출력될 이미지 폭 (옵션 : Scale 조정 시 사용)
- int iHeight : 출력될 이미지 높이 (옵션 : Scale 조정 시 사용)

C++ 사용법

```
typedef void  (*DisplayToPrint1Ex)(int iXpos, int iYpos, int iWidth, int iHeight);
typedef void  (*DisplayToPrint2Ex)(int iXpos, int iYpos);
DisplayToPrint1Ex      m_pfDisplayToPrint1;
DisplayToPrint2Ex     m_pfDisplayToPrint2;
HINSTANCE            hInst;
hInst = LoadLibrary(L"SmartPrint.dll");
if(hInst == NULL)
{
  return;
}
m_pfDisplayToPrint1 = (DisplayToPrint1Ex)GetProcAddress(m_hInst, _T("DisplayToPrint1"));
m_pfDisplayToPrint2 = (DisplayToPrint2Ex)GetProcAddress(m_hInst, _T("DisplayToPrint2"));
m_pfDisplayToPrint1(300, 1400, 10, 30);  // 프린터로 출력
m_pfDisplayToPrint2(300, 1400);  // 프린터로 출력
FreeLibrary(hInst);
```

함수	EllipseEx

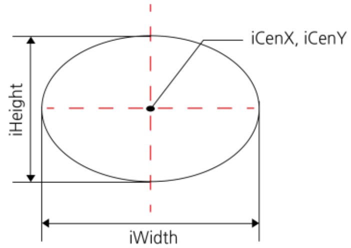
프린터 버퍼에 타원을 그립니다. 선의 색상 및 두께는 SetPenStyleEx() 함수로 설정하며 내부 채움은 SetBrush StyleEx() 함수로 설정합니다.

◉ typedef void (*EllipsesEx)(int iCenX, int iCenY, int iWidth, int iHeight)

[인자]

- int iCenX : 원 중심의 X좌표

- int iCenY : 원 중심의 Y좌표
- int iWidth : 타원의 수직 길이
- int iHeight : 타원의 수평 길이



C++ 사용법

```
typedef void  (*EllipsesEx)(int iCenX, int iCenY, int iWidth, int iHeight);
EllipsesEx          m_pfEllipses;
HINSTANCE           hInst;
hInst = LoadLibrary(L"SmartPrint.dll");
if(hInst == NULL)
{
  return;
}
m_pfEllipses = (EllipsesEx)GetProcAddress(m_hInst, _T("Ellipses"));
m_pfEllipses(300, 1400, 170, 30);  // 타원을 그린다.
FreeLibrary(hInst);
```

함수	EraseEx

프린터 버퍼에 그려진 데이터를 모두 지웁니다. 화면이 지워지면 배경 색상은 흰색으로 됩니다.

◉ typedef void (*EraseEx)()

C++ 사용법

```
typedef void  (*EraseEx)();
EraseEx             m_pfErase;
HINSTANCE           hInst;
hInst = LoadLibrary(L"SmartPrint.dll");
if(hInst == NULL)
{
  return;
}
m_pfErase = (EraseEx)GetProcAddress(m_hInst, _T("Erase"));
m_pfErase(); // 프린터 버퍼에 그려진 모든 그림을 지운다.
FreeLibrary(hInst);
```

함수	GetPaperPixelEx

현재 용지의 가로, 세로 크기를 Pixel 단위의 값으로 얻습니다. 프린터마다 Pixel값이 실제와 약간 다를수 있습니다.

◉ typedef void (*GetPaperPixelEx)(int* iWidthPixel, int* iHeightPixel)

[인자]

- int* iWidthPixel : 용지의 가로 크기(Pixel)
- int* iHeightPixel : 용지의 세로 크기(Pixel)

C++ 사용법

```
typedef void (*GetPaperPixelEx)(int* iWidthPixel, int* iHeightPixel);
GetPaperPixelEx      m_pfGetPaperPixel;
HINSTANCE            hInst;
hInst = LoadLibrary(L"SmartPrint.dll");
if(hInst == NULL)
{
  return;
}
m_pfGetPaperPixel = (GetPaperPixelEx)GetProcAddress(m_hInst, _T("GetPaperPixel"));
int iWidth, iHeight;
m_pfGetPaperPixel(iWidth, iHeight); // 용지의 크기를 읽는다.
FreeLibrary(hInst);
```

함수	GradientFillsEx

프린터 버퍼에 사각형 영역을 지정된 색으로 그라데이션 효과를 주어 그립니다.

◉ typedef void (*GradientFillsEx)(int iLeftX, int iTopY, int iWidth, int iHeight, int startColor, int endColor, int dwMode);

[인자]

- int iLeftX : 사각형 좌측 상단의 X좌표
- int iTopY : 사각형 좌측 상단의 Y좌표
- int iWidth : 사각형의 수평 길이
- int iHeight : 사각형의 수직 길이
- int startColor : 그라데이션의 시작 색상
- int endColor : 그라데이션의 끝 색상
- int dwMode : 그라데이션 방향
  - GRADIENT_FILL_RECT_H : 그라데이션 방향을 수평으로 설정
  - GRADIENT_FILL_RECT_V : 그라데이션 방향을 수직으로 설정

[표] GRADIENTFILLMODE 인자값에 따른 표현 예시

GRADIENT_FILL_RECT_H	GRADIENT_FILL_RECT_V
 수평그라데이션	 수직그라데이션

C++ 사용법

```
typedef void  (*GradientFillsEx)(int iLeftX, int iTopY, int iWidth, int iHeight, int startColor, 
int endColor, int dwMode);
GradientFillsEx      m_pfGradientFills;
```

```
HINSTANCE          hInst;
hInst = LoadLibrary(L"SmartPrint.dll");
if(hInst == NULL)
{
  return;
}
m_pfGradientFills = (GradientFillsEx)GetProcAddress(m_hInst, _T("GradientFills"));
// 그라데이션이 적용된 사각형을 그린다.
m_pfGradientFills(1010, 350, 400, 480, RGB(170, 0, 0), RGB(0, 0, 170), GRADIENT_FILL_RECT_H);
FreeLibrary(hInst);
```

함수 ImageDraw1Ex, ImageDraw2Ex, ImageDraw3Ex, ImageDraw4Ex Image

리소스로 포함된 이미지 또는 이미지 파일을 프린터 버퍼에 그립니다. (지원 이미지 파일 형식 : BMP, JPG, GIF, PNG)

주의 이미지 파일 출력 시 이미지 파일의 용량 관련 주의사항

이미지 파일의 용량이 클 경우 IEC- Series의 프로그램 메모리 용량에 따라 이미지 출력이 되지 않을 수 있습니다. 따라서 이미지 사용 시 되도록이면 적은 용량의 이미지를 사용하여 프린터 버퍼를 절약하시기 바랍니다. 다만, IEC667/1000 Series 제품을 사용하시면 상대적으로 프린터 버퍼를 확장하여 사용할 수 있습니다. 또한, ImageDraw1Ex() ~ ImageDraw4Ex() 이외의 함수는 프린터 버퍼를 적게 사용하는 벡터 출력 방식이므로 이미지 출력 이외의 기능은 메모리의 제한이 거의 없습니다.

Image 이미지 사용 시 주의사항

사용되는 이미지가 "이미지 제작 가이드"에 맞게 제작되지 않은 경우, 다양한 오류가 발생할 수 있습니다.
※ 반드시 "이미지 제작 가이드"를 참고하여 이미지를 제작하시기 바랍니다.

- ⦿ typedef void (*ImageDraw1Ex)(int iResourceID, int iXpos, int iYpos, int iWidth, int iHeight)
- ⦿ typedef void (*ImageDraw2Ex)(int iResourceID, int iXpos, int iYpos)
- ⦿ typedef void (*ImageDraw3Ex)(LPCTSTR strFilePath, int iXpos, int iYpos, int iWidth, int iHeight)
- ⦿ typedef void (*ImageDraw4Ex)(LPCTSTR strFilePath, int iXpos, int iYpos)

[인자]

- int iResourceID : 이미지 파일이 리소스로 포함된 경우 리소스명 (필수)
- LPCTSTR strFilePath : 이미지 파일의 경로와 파일명 (필수)
- int iXpos : 출력될 이미지 좌측 상단의 X좌표 (필수)
- int iYpos : 출력될 이미지 좌측 상단의 Y좌표 (필수)
- int iWidth : 출력될 이미지의 수평 길이 (옵션 : Scale 조정 시 사용)
- int iHeight : 출력될 이미지의 수직 길이 (옵션 : Scale 조정 시 사용)

C++ 사용법

```
typedef void  (*ImageDraw1Ex)(int iResourceID, int iXpos, int iYpos, int iWidth, int iHeight);
typedef void  (*ImageDraw2Ex)(int iResourceID, int iXpos, int iYpos);
typedef void  (*ImageDraw3Ex)(LPCTSTR strFilePath, int iXpos, int iYpos, int iWidth, int
iHeight);
typedef void  (*ImageDraw4Ex)(LPCTSTR  strFilePath, int iXpos, int iYpos);
ImageDraw1Ex         m_pfImageDraw1;
ImageDraw2Ex         m_pfImageDraw2;
ImageDraw3Ex         m_pfImageDraw3;
ImageDraw4Ex         m_pfImageDraw4;
```

```
HINSTANCE           hInst;
hInst = LoadLibrary(L"SmartPrint.dll");
if(hInst == NULL)
{
  return;
}
m_pfImageDraw1 = (ImageDraw1Ex)GetProcAddress(m_hInst, _T("ImageDraw1"));
m_pfImageDraw2 = (ImageDraw2Ex)GetProcAddress(m_hInst, _T("ImageDraw2"));
m_pfImageDraw3 = (ImageDraw3Ex)GetProcAddress(m_hInst, _T("ImageDraw3"));
m_pfImageDraw4 = (ImageDraw4Ex)GetProcAddress(m_hInst, _T("ImageDraw4"));
// 리소스 이미지 프린터 버퍼에 그리기
m_pfImageDraw2(IDB_BITMAP1, 1520, 350);
FreeLibrary(hInst);
```

함수	IsPrintingEx

프린터의 인쇄 상태를 확인합니다.

◉ typedef int (*IsPrintingEx)()

[리턴값]

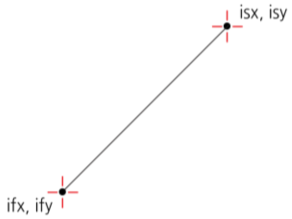
- int : 프린터의 인쇄 상태
  - 0 : 인쇄 성공
  - 1 : 인쇄 중
  - 2 : 인쇄 실패

C++ 사용법

```
typedef int (*IsPrintingEx)();
IsPrintingEx        m_pfIsPrinting;
HINSTANCE           hInst;
hInst = LoadLibrary(L"SmartPrint.dll");
if(hInst == NULL)
{
  return;
}
m_pfIsPrinting = (IsPrintingEx)GetProcAddress(m_hInst, _T("IsPrinting"));
// 현재 인쇄 중인지 확인
switch(m_pfIsPrinting())
{
  case 2:
    // 인쇄 실패
    break;
  case 1:
    // 인쇄 중
    break;
  case 0:
    // 인쇄 완료
    break;
}
FreeLibrary(hInst);
```

함수	LineEx

프린터 버퍼에 선을 그립니다. 선의 색상 및 두께는 SetPenStyleEx() 함수로 설정합니다.



◉ typedef void (*LineEx)(int ifx, int ify, int isx, int isy)

[인자]

- int ifx : 처음 점의 X좌표
- int ify : 처음 점의 Y좌표
- int isx : 두번째 점의 X좌표
- int isy : 두번째 점의 Y좌표

C++ 사용법

```
typedef void  (*LineEx)(int ifx, int ify, int isx, int isy);
LineEx              m_pfLine;
HINSTANCE           hInst;
hInst = LoadLibrary(L"SmartPrint.dll");
if(hInst == NULL)
{
  return;
}
m_pfLine = (LineEx)GetProcAddress(m_hInst, _T("Line"));
// 프린터 버퍼에 선 그리기
m_pfLine(1010, 350, 400, 480);
FreeLibrary(hInst);
```

함수	NewPageEx

여러 페이지를 출력하는 경우, 각각의 새로운 페이지를 생성할 때 호출합니다. 즉, 새로운 페이지를 만듭니다.

◉ typedef void (*NewPageEx)()

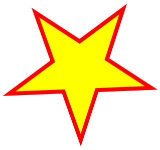
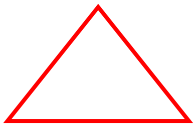
C++ 사용법

```
typedef void (*NewPageEx)();
NewPageEx           m_pfNewPage;
HINSTANCE           hInst;
hInst = LoadLibrary(L"SmartPrint.dll");
if(hInst == NULL)
{
  return;
}
m_pfNewPage = (NewPageEx)GetProcAddress(m_hInst, _T("NewPage"));
// 새로운 페이지를 만듬
m_pfNewPage();
FreeLibrary(hInst);
```

함수	PolygonsEx, PolylinesEx

PolygonsEx() 함수와 PolylinesEx() 함수는 연속된 여러 개의 선을 프린터 버퍼에 그립니다. 선의 색상 및 두께는 SetPenStyleEx() 함수로 설정하며, PolygonsEx() 함수의 경우 SetBrushStyleEx() 함수로 설정된 색상으로 안쪽이 채워집니다.

[표] PolygonsEx() 함수와 PolylinesEx() 함수의 차이점

함수	PolygonsEx	PolylinesEx
차이점	1. 시작과 끝점이 연결됨 2. 연결된 선의 안쪽이 채워짐	1. 시작과 끝점이 연결되지 않음 2. 연결된 선의 안쪽이 채워지지 않음
출력 결과		

◉ typedef void (*PolygonsEx)(POINT* pPoint, int iNum);

◉ typedef void (*PolylinesEx)(POINT* pPoint, int iNum);

[인자]

- POINT* pPoint : 연속된 선을 그리기 위한 여러 점의 좌표 배열
- int iNum : pPoint 배열의 멤버 수

C++ 사용법

```
typedef void  (*PolylinesEx)(POINT* pPoint, int iNum);
typedef void  (*PolygonsEx)(POINT* pPoint, int iNum);
PolylinesEx          m_pfPolylines;
PolygonsEx           m_pfPolygons;
HINSTANCE            hInst;
hInst = LoadLibrary(L"SmartPrint.dll");
if(hInst == NULL)
{
  return;
}
m_pfPolylines = (PolylinesEx)GetProcAddress(m_hInst, _T("Polylines"));
m_pfPolygons = (PolygonsEx)GetProcAddress(m_hInst, _T("Polygons"));
// 별을 그리기위한 좌표 데이터 변수 선언 및 정의
POINT points[10];
int iSize, iXOffset, iYOffset;
iXOffset = 1600;
iYOffset = 1150;
iSize = 10;
points[0].x = (16 * iSize) + iXOffset;
points[0].y = (28 * iSize) + iYOffset;
points[1].x = (19 * iSize) + iXOffset;
points[1].y = (17 * iSize) + iYOffset;
points[2].x = (30 * iSize) + iXOffset;
points[2].y = (17 * iSize) + iYOffset;
points[3].x = (21 * iSize) + iXOffset;
points[3].y = (11 * iSize) + iYOffset;
```

```
points[4].x = (25 * iSize) + iXOffset;
points[4].y = (1 * iSize) + iYOffset;
points[5].x = (16 * iSize) + iXOffset;
points[5].y = (7 * iSize) + iYOffset;
points[6].x = (7 * iSize) + iXOffset;
points[6].y = (1 * iSize) + iYOffset;
points[7].x = (11 * iSize) + iXOffset;
points[7].y = (11 * iSize) + iYOffset;
points[8].x = (2 * iSize) + iXOffset;
points[8].y = (17 * iSize) + iYOffset;
points[9].x = (13 * iSize) + iXOffset;
points[9].y = (17 * iSize) + iYOffset;
// 내부가 채워진 별을 프린터 버퍼에 그린다.
m_pfPolygons.Polygon(points, 10);

POINT points1[4];
points1[0].X = 400;
points1[0].Y = 200;
points1[1].X = 300;
points1[1].Y = 300;
points1[2].X = 500;
points1[2].Y = 300;
points1[3].X = 400;
points1[3].Y = 200;
// 설정된 데이터를 프린터 버퍼에 출력(내부가 투명한 삼각형)
m_pfPolylines(points1, 4);
FreeLibrary(hInst);
```

함수	PrintOutEx

프린터 버퍼에 출력된 데이터를 프린터로 전송하여 인쇄를 시작합니다. 1번 호출 시 1페이지 단위로 출력됩니다.

◉ `typedef void (*PrintOutEx)()`

C++ 사용법

```
typedef void (*PrintOutEx)()
PrintOutEx          m_pfPrintOut;
HINSTANCE           hInst;
hInst = LoadLibrary(L"SmartPrint.dll");
if(hInst == NULL)
{
  return;
}
m_pfPrintOut = (PrintOutEx)GetProcAddress(m_hInst, _T("PrintOut"));
// 프린터 버퍼에 출력된 데이터를 인쇄
m_pfPrintOut();
FreeLibrary(hInst);
```

함수 PrintSetupEx

프린터 출력을 위한 프린터 관련 환경 설정 및 초기화를 진행합니다.

주의 프린터를 사용하기 전 반드시 PrintSetupEx() 함수를 호출하시기 바랍니다.

◉ typedef bool (*PrintSetupEx)(LPTSTR strPrinterID, int iQuality, int iColorMode, int iPaperSize, int iOrientation, int iCopies)

[인자]

- LPTSTR strPrinterID : 사용자 지정 프린터 이름
- int iQuality : 해상도(DPI)
  - 0 : 300DPI
  - 1 : 600DPI
- int iColorMode : 컬러 모드
  - 0 : 흑백
  - 1 : 컬러
- int iPaperSize : 용지 크기
  - 0 : A4 210 x 297 mm
  - 2 : B5 (JIS) 182 x 257 mm
  - 3 : Letter 8 1/2 x 11 inch
  - 4 : Legal 8 1/2 x 14 inch
- int iOrientation : 용지 방향
  - 0 : 가로 방향
  - 1 : 세로 방향
- int iCopies : 인쇄 할 페이지 수

[리턴값]

- bool : 설정 성공 여부를 리턴
  - true : 설정 완료
  - false : 설정 실패

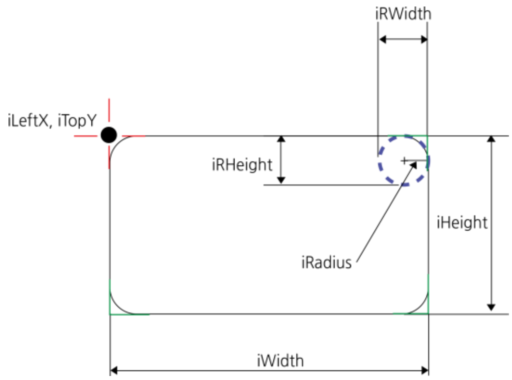
C++ 사용법

```
typedef bool (*PrintSetupEx)(LPTSTR strPrinterID, int iQuality, int iColorMode, int iPaperSize,
int iOrientation, int iCopies);
PrintSetupEx         m_pfPrintSetup;
HINSTANCE            hInst;
hInst = LoadLibrary(L"SmartPrint.dll");
if(hInst == NULL)
{
  return;
}
m_pfPrintSetup = (PrintSetupEx)GetProcAddress(m_hInst, _T("PrintSetup"));
m_pfPrintSetup(L"PCL Inkjet", 0, 0, 0, 1, 1);
FreeLibrary(hInst);
```

함수	RectanglesEx, RoundRect1Ex, RoundRect2Ex

내부가 채워진 직사각형을 프린터 버퍼에 그립니다. 선의 색상 및 두께는 SetPenStyleEx() 함수로 설정하며 내부 채움은 SetBrushStyleEx() 함수로 설정합니다.

- RectanglesEx() : 직사각형을 그립니다.
- RoundRect1Ex(), RoundRect2Ex() : 모서리가 둥근 직사각형을 그립니다.



◉ typedef void (*RectanglesEx)(int iLeftX, int iTopY, int iWidth, int iHeight)

◉ typedef void (*RoundRect1Ex)(int iLeftX, int iTopY, int iWidth, int iHeight, int iRWidth, int iRHeight)

◉ typedef void (*RoundRect2Ex)(int iLeftX, int iTopY, int iWidth, int iHeight, int iRadius)

[인자]

- int iLeftX : 출력될 사각형 좌측 상단의 X좌표,
- int iTopY : 출력될 사각형 좌측 상단의 Y좌표
- int iWidth : 출력될 사각형의 수평 길이
- int iHeight : 출력될 사각형의 수직 길이
- int iRadius : RoundRect에서 모서리의 라운드의 반지름
- int iRWidth : RoundRect에서 모서리의 라운드의 수평 길이
- int iRHeight : RoundRect에서 모서리의 라운드의 수직 길이

C++ 사용법

```
typedef void  (*RectanglesEx)(int iLeftX, int iTopY, int iWidth, int iHeight);
typedef void  (*RoundRect1Ex)(int iLeftX, int iTopY, int iWidth, int iHeight, int iRWidth, int
iRHeight);
typedef void  (*RoundRect2Ex)(int iLeftX, int iTopY, int iWidth, int iHeight, int iRadius);
RectanglesEx         m_pfRectangles;
RoundRect1Ex         m_pfRoundRect1;
RoundRect2Ex         m_pfRoundRect2;
HINSTANCE            hInst;
hInst = LoadLibrary(L"SmartPrint.dll");
if(hInst == NULL)
{
  return;
}
m_pfRectangles = (RectanglesEx)GetProcAddress(m_hInst, _T("Rectangles"));
m_pfRoundRect1 = (RoundRect1Ex)GetProcAddress(m_hInst, _T("RoundRect1"));
```

```
m_pfRoundRect2 = (RoundRect2Ex)GetProcAddress(m_hInst, _T("RoundRect2"));
// 프린터 버퍼에 내부가 채워진 사각형을 그린다.
m_pfRectangles(3, 3, 2390, 3300);
// 프린터 버퍼에 내부가 채워진 모서리가 둥근 사각형을 그린다.
m_pfRectangles2(100, 2900, 2220, 330, 40);
FreeLibrary(hInst);
```

함수	SetPenStyleEx, SetBrushStyleEx

도형의 선 및 내부 채움 색상을 설정합니다. 설정 이후 그려지는 모든 도형에 적용됩니다.

- SetPenStyleEx() : 도형의 선 색상 및 두께를 설정합니다.
- SetBrushStyleEx() : 도형의 내부 채움 색상을 설정합니다.

◉ typedef void (*SetPenStyleEx)(int penColor, int iWidth)

◉ typedef void (*SetBrushStyleEx)(int brushColor, bool bTransparent)

[인자]

- int penColor : 선의 색상
- int iWidth : 선의 두께
- int brushColor : 도형 내부를 채울 색상
- bool bTransparent : 도형 내부 투명처리 여부
  - true : 도형 내부를 투명하게 처리
  - false : 도형 내부를 SetBrushStyleEx()로 설정한 색으로 채움

C++ 사용법

```
typedef void (*SetBrushStyleEx)(int brushColor, bool bTransparent);
typedef void  (*SetPenStyleEx)(int penColor, int iWidth);
SetBrushStyleEx      m_pfSetBrushStyle;
SetPenStyleEx        m_pfSetPenStyle;
HINSTANCE            hInst;
hInst = LoadLibrary(L"SmartPrint.dll");
if(hInst == NULL)
{
  return;
}
m_pfSetBrushStyle = (SetBrushStyleEx)GetProcAddress(m_hInst, _T("SetBrushStyle"));
m_pfSetPenStyle = (SetPenStyleEx)GetProcAddress(m_hInst, _T("SetPenStyle"));
// 도형의 선 색상 및 두께 설정
m_pfSetPenStyle(RGB(200,0,0), 5);
// 도형의 내부를 투명하게 채움
m_pfSetBrushStyle(RGB(255, 255, 255), true);
FreeLibrary(hInst);
```

함수	SetFontCfg1Ex, SetFontCfg2Ex, SetFontCfg3Ex

TextOut() 함수로 프린터 버퍼에 텍스트를 출력할 경우 폰트 관련 속성값을 설정합니다.

참조	다양한 폰트를 추가하는 방법은 "홈페이지 → 자료실 → Tech Note → 20. IEC-Series에서 다양한 폰트(Fonts폴더, 트루타입)사용 방법안내" 내용을 참조하시기 바랍니다.

⦿ typedef void (*SetFontCfg1Ex)(LPTSTR strFontName, int iFontSize, int cFontColor, bool bBold, bool bItalic, bool bAntiAlased)

⦿ typedef void (*SetFontCfg2Ex)(LPTSTR strFontName, int iFontSize, int cFontColor)

⦿ typedef void (*SetFontCfg3Ex)(int iFontSize, int cFontColor)

[인자]

- LPTSTR strFontName : 출력 되는 폰트
- int iFontSize : 출력되는 폰트의 크기
- int cFontColor : 출력되는 폰트의 색상
- bool bBold : 출력되는 폰트의 굵기
  - true : 굵게
  - false : 표준
- bool bItalic : 출력되는 폰트의 기울임
  - true : 기울임
  - false : 표준
- bool bAntiAlased : 출력되는 폰트에 배경과 글자의 안티에일리어싱(Anti-Aliasing)효과 사용 여부
  - true : 안티에일리어싱 사용
  - false : 안티에일리어싱 사용 안함

C++ 사용법

```
typedef void  (*SetFontCfg1Ex)(LPTSTR strFontName, int iFontSize, int cFontColor, bool bBold,
bool bItalic, bool bAntiAlased);
typedef void  (*SetFontCfg2Ex)(LPTSTR strFontName, int iFontSize, int cFontColor);
typedef void  (*SetFontCfg3Ex)(int iFontSize, int cFontColor);
SetFontCfg1Ex         m_pfSetFontCfg1;
SetFontCfg2Ex         m_pfSetFontCfg2;
SetFontCfg3Ex         m_pfSetFontCfg3;
HINSTANCE             hInst;
hInst = LoadLibrary(L"SmartPrint.dll");
if(hInst == NULL)
{
  return;
}
m_pfSetFontCfg1 = (SetFontCfg1Ex)GetProcAddress(m_hInst, _T("SetFontCfg1"));
m_pfSetFontCfg2 = (SetFontCfg2Ex)GetProcAddress(m_hInst, _T("SetFontCfg2"));
m_pfSetFontCfg3 = (SetFontCfg3Ex)GetProcAddress(m_hInst, _T("SetFontCfg3"));
// 도형의 선 색상 및 두께 설정
m_pfSetFontCfg1(L"굴림체", 47, RGB(255, 255, 255), true, false, false);
m_pfSetFontCfg2(L"굴림체", 47, RGB(255, 255, 255));
m_pfSetFontCfg3(47, RGB(255, 255, 255));
FreeLibrary(hInst);
```

함수	SetPixelsEx

지정된 좌표에 색상을 지정하여 프린터 버퍼에 점을 그립니다.

⦿ typedef void (*SetPixelsEx)(int iXPos, int iYPos, int crColor)

[인자]

- int iXPos : 점의 X좌표
- int iYPos : 점의 Y좌표
- int crColor : 그려질 점의 색상

C++ 사용법

```
typedef void  (*SetPixelsEx)(int iXPos, int iYPos, int crColor);
SetPixelsEx          m_pfSetPixels;
HINSTANCE            hInst;
hInst = LoadLibrary(L"SmartPrint.dll");
if(hInst == NULL)
{
  return;
}
m_pfSetPixels = (SetPixelsEx)GetProcAddress(m_hInst, _T("SetPixels"));
// 특정 위치에 점을 찍음
m_pfSetPixels(10,100, RGB(200,0,0));
FreeLibrary(hInst);
```

함수	TextOutEx

텍스트를 프린터 버퍼에 출력합니다. 폰트 관련 속성 설정은 SetFontCfgEx() 함수로 설정합니다.

⦿ typedef void (*TextOutEx)(int iXpos, int iYpos, LPTSTR strFormat)

[인자]

- int iXpos : 출력되는 텍스트 좌측 상단의 X좌표
- int iYpos : 출력되는 텍스트 좌측 상단의 Y좌표
- LPTSTR strFormat : 출력될 문자열

C++ 사용법

```
typedef void  (*TextOutEx)(int iXpos, int iYpos, LPTSTR strFormat);
TextOutEx            m_pfTextOut;
HINSTANCE            hInst;
hInst = LoadLibrary(L"SmartPrint.dll");
if(hInst == NULL)
{
  return;
}
m_pfTextOut = (TextOutEx)GetProcAddress(m_hInst, _T("TextOut"));
// SetFontCfg1 설정값대로 특정 위치에 텍스트를 출력
m_pfTextOut(120, 2950, L"Test!!");
FreeLibrary(hInst);
```

4) CSmartPrintWrap Class 설명

SmartPrintWrap Class는 사용할 인터페이스를 각각 선언 및 얻어오는 복잡한 과정을 거칠 필요 없이 한 번의 선언과 인스턴싱만으로 편리하게 SmartPrint의 기능을 C++(C++, Visual Studio 2005, 2008)에서 지원하도록 Wrapping 한 Class입니다. 본 Class를 사용하기 위해서는 “SmartPrintWrap.h” 파일과 “SmartPrintWrap.cpp” 파

일을 대상 프로젝트에 복사한 후 #include "SmartPrintWrap.h"을 호출하여 사용하시기 바랍니다.

인터페이스 및 사용 방식은 위에서 설명한 C++ 인터페이스와는 약간의 차이가 있으므로 다음의 내용을 참고하시어 C++에서 적용하시기 바랍니다.

4-1) SmartPrintWrap Class 사용하기

C++에서 SmartPrintWrap Class를 사용하기 위해서는 아래와 STEP을 진행해야 합니다.

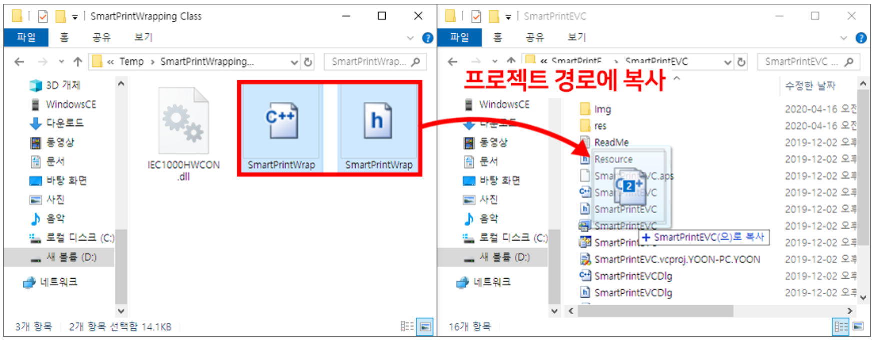
STEP-1 Wrap Class 파일 복사하기

HNS 홈페이지에서 사용하는 IEC-Series에 맞는 예제파일을 다운받은 후 압축을 풉니다.
"SmartPrintEVC" 폴더에서 아래 표를 확인하여 Wrap Class 파일을 프로젝트 경로에 복사합니다.

[표] IEC-Series별 Wrap Class 파일

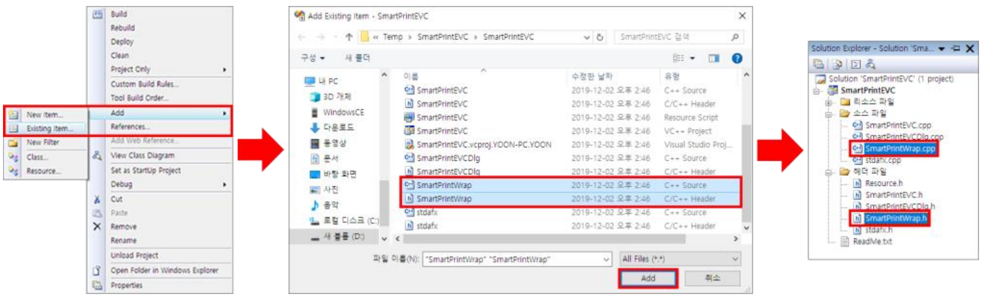
IEC-Series	IEC266-Series	IEC667-Series	IEC1000-Series
Wrap Class 파일	SmartPrintWrap.h, SmartPrintWrap.cpp		
다운로드 위치	홈페이지 → 자료실 → SmartX Framework 관련 → 6. SmartX Framework 예제파일 → Visual C++ 예제파일 → "SmartPrintEVC"		

Wrap Class 파일을 프로젝트 경로에 복사합니다.



STEP-2 Wrap Class 추가하기

복사한 Wrap Class 파일을 프로젝트에 추가합니다.



STEP-3 Wrap Class 사용하기

사용할 소스 파일에 #include<SmartPrintWrap.h> 추가한 후 선언 및 인스턴스하여 사용합니다.

```
#include "SmartPrintWrap.h"
// SmartPrint를 사용하기 위해 변수 선언 및 인스턴싱
CSmartPrintWrap m_SmartPrint;
```

```
// 프린터 설정
m_SmartPrint.(L"PCL Inkjet", 0, 0, 0, 1, 1);
// 인쇄 폰트 설정
m_SmartPrint.SetFontCfg(L"굴림체", 70, RGB(255, 255, 255), true, false, false);
// 펜 및 내부 채움 속성 설정
m_SmartPrint.SetPenStyle(RGB(200,0,0), 5);
m_SmartPrint.SetBrushStyle(RGB(255, 255, 255), true);
// 페이지 외곽선 프린터 버퍼에 그리기
m_SmartPrint.Rectangle(3, 3, 2390, 3300);
// 텍스트를 프린터 버퍼에 그리기
m_SmartPrint.TextOut(123, 160, L"SmartPrint Test!!");
// 현재 화면 버퍼에 그리기
m_SmartPrint.DisplayToPrint(100, 350);
// 리소스 이미지 프린터 버퍼에 그리기
m_SmartPrint.ImageDraw(IDB_BITMAP1, 1520, 350);
// 여러가지 도형을 프린터 버퍼에 그리기
m_SmartPrint.Rectangle(100, 1000, 2220, 100);
m_SmartPrint.Circle(500, 1400, 270);
// 프린터 버퍼에 그려진 데이터를 인쇄
m_SmartPrint.PrintOut();
```

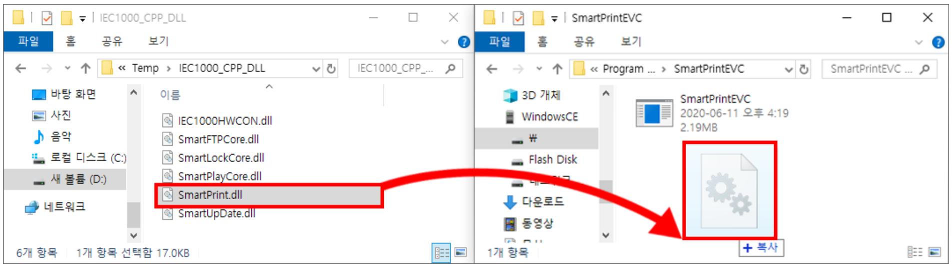
STEP-4 DLL 파일 복사하기

HNS 홈페이지에서 사용하는 IEC-Series에 맞는 예제파일을 다운받은 후 압축을 풉니다. "참조 DLL" 폴더에서 아래 표를 확인하여 사용하는 기능에 맞는 DLL 파일을 IEC-Series에 실행파일과 함께 복사합니다.

[표] IEC-Series별 DLL 파일과 다운로드 경로

IEC-Series	IEC266-Series	IEC667-Series	IEC1000-Series
DLL 파일	SmartPrint.dll		
다운로드 위치	홈페이지 → 자료실 → SmartX Framework 관련 → 5. SmartX C++ For DLL 파일		

DLL 파일을 실행 파일 위치에 복사합니다.

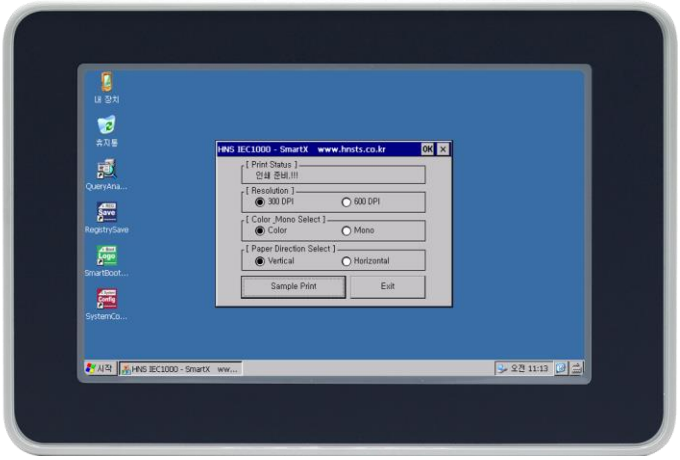


참고 자세한 DLL 파일의 복사 경로는 "DLL 파일 및 복사 경로" 내용을 참고하시기 바랍니다.

4-2) SmartPrintWrap Class 예제 사용하기

SmartPrintWrap Class를 활용한 예제를 제공하고 있으며, 예제파일은 다음 경로에서 다운로드하여 참고하시기 바랍니다.

[예제 파일 다운로드 위치]
홈페이지(www.hnsts.co.kr) → 자료실 → SmartX Framework 예제파일 다운로드 → "SmartPrintEVC"



9. SmartVideo

SmartX Framework는 기본적으로 .Net Compact Framework에서만 지원되도록 설계되어 있지만, 일부 기능을 C++에서 편리하게 사용할 수 있도록 외부 DLL 파일을 제공하고 있습니다. 따라서 SmartVideo를 사용하기 위해서는 반드시 외부 DLL 파일을 실행 파일과 같이 IEC-Series에 배포해야 하며, "1. 인터페이스 함수 선언 → 2. 라이브러리 메모리 로드 처리 → 3. 인터페이스 함수 얻기" 과정을 거쳐야 합니다. 이때 IEC-Series 및 기능별로 로드해야 할 외부 DLL 파일이 다르므로 다음 내용을 확인하시기 바랍니다.

주의 SmartVideo 사용 시 주의사항

1. IEC667非Lite-Series에서만 지원되며, SmartVideo Board가 반드시 필요합니다.
2. 인터페이스 가능한 카메라는 NTSC(National Television System Committee)의 신호를 출력하는 카메라를 장착하실 수 있습니다.
3. SmartVideo를 사용 시 PortB 6, 7번 Pin을 사용하게 됩니다. SmartIIC도 동일한 Pin을 사용하므로 SmartVideo와 SmartIIC는 동시에 사용할 수 없습니다.
4. SmartVideo에서 동영상 Encoding 중 동시에 화면 캡쳐는 권장하지 않습니다. 스레드가 충돌할 수 있는 위험성이 존재합니다.
5. 개발된 Program에서 CPU Performance를 많이 사용하는 경우, 상/하/좌/우 화면 떨림이 발생할 수 있습니다. (화면의 크기와 비례하여 증가합니다.)

참조 SmartVideo의 자세한 설명은 "홈페이지(www.hnsts.co.kr) → 자료실 → SmartX Framework 관련 → 프로그래밍 가이드 → SmartVideo"를 참조하시기 바랍니다.

[SmartVideo Board]

[SmartVideo Board 사진 및 시스템 연결사진]

1) 인터페이스 사양

아래 표에서 SmartVideo의 인터페이스 사양을 확인하여 적용하시기 바랍니다.

[표] SmartVideo 인터페이스 사양

입력 영상 신호 방식	NTSC(National Television System Committee)
입력 영상 채널 수	4 채널 / Capture & Preview (한 번에 한 채널만 선택 출력)
영상 속성 제어 기능	밝기(Brightness), 색조(Hue), 명암(Contrast), 채도(Color Saturation)
Preview 최대 해상도	800 x 600 → 해상도 가변
동영상 & 정지영상 저장 최대 해상도	720 x 480 (640 x 480) → 해상도 가변
동영상 저장 Codec	H.264(10fps, 15fps, 20fps)
정지 영상 저장 Format	BMP, GIF, PNG, JPG

연결 가능한 IEC제품	※ IEC667非Lite-Series만 연결 가능합니다! ※ IEC667-07 [B1 or B2], IEC667-08[B1 or B2], IEC667-102[B1 or B2], IEC667-104[B1 or B2],
기타 부가 기능	1. Preview(동영상), Capture(정지 영상) 각각 별도 해상도 제어 2. 정지 영상 메모리 Capture 기능
지원 OS	Core – Standard , Core – Professional 지원

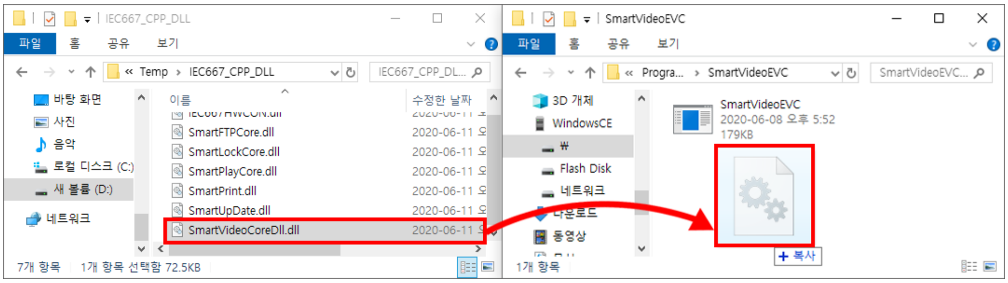
2) DLL 파일 및 복사 경로

HNS 홈페이지에서 사용하는 IEC-Series에 맞는 C++ For DLL 파일을 다운로드 받은 후 압축을 풉니다. 아래 표를 확인하여 DLL 파일을 IEC-Series에 실행 파일과 함께 복사합니다.

[표] IEC-Series별 DLL 파일과 다운로드 경로

IEC-Series	IEC266-Series	IEC667-Series	IEC1000-Series
DLL 파일	SmartVideoCoreDll.dll		
다운로드 위치	홈페이지 → 자료실 → SmartX Framework 관련 → 5. SmartX C++ For DLL 파일		

DLL 파일을 IEC-Series에 복사합니다. DLL 파일을 사용하는 실행 파일이 저장된 폴더에 저장하는 것을 권장하며, 다른 경로에 저장하는 경우 "인터페이스 선언 및 얻기"의 참고 내용을 확인하시기 바랍니다.
※ 개발 PC에서 프로그램을 배포하는 경우 실행 파일은 "Program File" 폴더에 배포됩니다.



3) 인터페이스 선언 및 얻기

인터페이스 사용을 위해 사용할 인터페이스의 선언 후 외부 DLL 파일을 메모리에 로드하여 인터페이스를 얻는 과정을 거쳐야 합니다. 이때 메모리 로드 처리를 위해 LoadLibrary() 함수 호출 시 상단 표의 DLL 파일의 이름을 인자로 설정합니다. 아래 코드는 예시이며, 필요로 하는 인터페이스 함수를 사용하기 위해서는 다음 내용을 참고하시기 바랍니다.

참고 DLL 파일 저장 경로에 따른 LoadLibrary() 함수 인자값

DLL 파일이 실행 파일과 같은 경로에 저장된 경우, LoadLibrary() 함수 인자값을 DLL 파일 이름만으로 설정해 호출할 수 있습니다. 만약 DLL 파일이 실행 파일과 다른 경로에 저장된 경우, LoadLibray() 함수의 인자값에는 저장된 DLL 파일의 전체 경로를 인자값으로 설정해야 합니다.

[C++] 인터페이스 얻기 및 사용하기 예시 코드

```
// 1. 인터페이스 함수 선언
typedef void (*SetDefaultControlEx)();
SetDefaultControlEx  m_pfSetDefaultControl;
// 2. 라이브러리 메모리 로드 처리
```

```
HINSTANCE  m_hInst;
// SmartVideo의 인터페이스를 사용하기 위해 SmartVideoCoreDll.dll 파일을 메모리에 로드
// IEC-Series 및 사용하는 기능별로 외부 DLL 파일이 다르며, 상단 표를 참고하시기 바랍니다.
m_hInst = LoadLibrary(L"SmartVideoCoreDll.dll");
// 외부 DLL 파일이 다른 경로(Flash Disk\\extdlls)에 저장된 경우
// m_hInst = LoadLibrary(L"Flash Disk\\extdlls\\SmartVideoCoreDll.dll");
if(m_hInst == NULL)
{
  // 메모리 로드 실패
  return FALSE;
}
// 3. 인터페이스 함수 얻기
m_pfSetDefaultControl = (SetDefaultControlEx)GetProcAddress(m_hInst, _T("SetDefaultControl"));
// 4. 함수 사용
m_pfSetDefaultControl();
```

참고 Wrapping Class 제공안내

사용할 인터페이스를 각각 선언 및 얻어오는 복잡한 과정을 거칠 필요 없이 한 번의 선언과 인스턴싱만으로 편리하게 SmartVideo의 인터페이스를 사용할 수 있도록 Wrapping Class를 제공하고 있습니다.
자세한 내용은 "CSmartVideoWrap Class 설명" 내용을 참고하시기 바랍니다.

4) C++ 인터페이스 설명

함수	CaptureStillImageEx

현재 영상의 정지 영상을 이미지 파일로 저장합니다. 저장 형식은 BMP, GIF, PNG, JPG 등으로 저장 가능합니다. 저장 시 적용되는 해상도는 SetCaptureResolution() 메소드에서 설정된 해상도로 저장됩니다.

◉ typedef bool (*CaptureStillImageEx)(LPTSTR strImgPathName)

[인자]

• LPTSTR strImgPathName : 저장될 정지 영상의 경로. 파일 형식은 파일 이름의 확장명에 따라서 적용됨

[리턴값]

• bool : 이미지 저장 성공 여부
  - true : 저장 성공
  - false : 저장 실패

C++ 사용법

```
typedef bool (*CaptureStillImageEx)(LPTSTR strImgPathName);
CaptureStillImageEx m_pfCaptureStillImage;
HINSTANCE          hInst;
hInst = LoadLibrary(L"SmartVideoCoreDll.dll");
if(hInst == NULL)
{
  return;
}
m_pfCaptureStillImage = (CaptureStillImageEx)GetProcAddress(m_hInst, _T("CaptureStillImage"));
// 정지 영상을 지정된 PNG 이미지 파일로 저장
m_pfCaptureStillImage(L"SD Card\\SmartX.png");
FreeLibrary(hInst);
```

함수 **EndEncodingEx, InitEncodingEx, RunEncodingEx, StopEncodingEx**

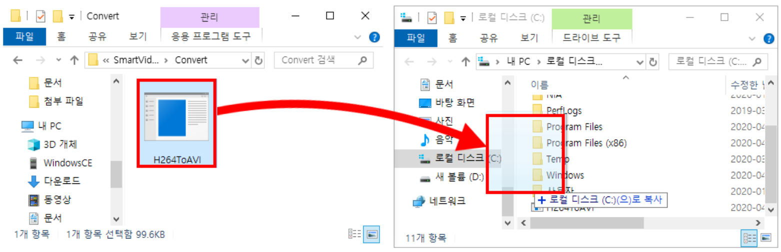
영상(동영상)을 저장하기 위한 초기화 및 시작, 일시 중지, 완료 등의 동작을 제어합니다. 저장된 동영상 파일은 단순히 H.264로 압축된 데이터입니다. 따라서 PC에서 파일을 Play 하기 위해서는 아래와 같이 변환하여 사용하시기 바랍니다. ※ 동영상 저장 시 약간의 시간이 발생할 수 있습니다.

- EndEncodingEx() : 동영상 저장을 완료(종료)합니다.
- InitEncodingEx() : 동영상을 저장을 위해 설정을 초기화합니다. 파라미터 값 등이 변경될 경우 호출합니다.
- RunEncodingEx() : 동영상 저장을 시작합니다.
- StopEncodingEx() : 동영상 저장을 잠시 중단합니다.

참고 H.264 압축 영상 파일을 AVI 확장자로 변경하는 방법

H.264 압축 영상 파일을 AVI 확장자로 변환하기 위해 변환 툴을 제공하고 있습니다. 변환 툴은 "SmartVideo 예제 폴더 → Convert 폴더 → H264ToAVI.exe" 이며 CMD를 사용하여 영상 파일을 변환할 수 있습니다.

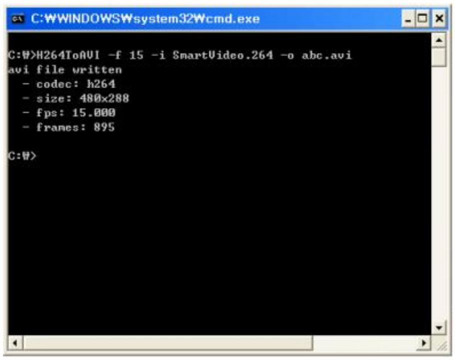
[STEP-1] "SmartVideo 예제 폴더 → Convert 폴더 → H264ToAVI.exe" 파일을 C드라이브에 이동



[STEP-2] CMD 실행 후 실행 경로 설정

CMD 실행 후 cd C:₩ 를 입력합니다.

[STEP-3] H264ToAVI [-f options] [-i input.h264] [-o output.avi] 명령어를 사용하여 영상을 변환



H264ToAVI [-f options] [-i input.h264] [-o output.avi]

- -f : Frame Rate(저장시 사용했던 값으로 설정. Ex : -f 10, -f 15, -f 20)
- -i : H.264 압축 영상 파일
- -o : AVI 형식으로 변환될 파일

※ 160X96 크기로 저장된 동영상 파일은 PC에서 Play되지 않습니다.

◉ typedef bool (*EndEncodingEx)()

◉ typedef bool (*InitEncodingEx)()

◉ typedef bool (*RunEncodingEx)()

◉ typedef bool (*StopEncodingEx)()

[리턴값]

- bool : 작업 성공 여부
  - true : 성공
  - false : 실패

C++ 사용법

```
typedef bool (*EndEncodingEx)()
typedef bool (*InitEncodingEx)()
typedef bool (*RunEncodingEx)()
typedef bool (*StopEncodingEx)()
EndEncodingEx                m_pfEndEncoding;
InitEncodingEx               m_pfInitEncoding;
RunEncodingEx                m_pfRunEncoding;
StopEncodingEx               m_pfStopEncoding;
HINSTANCE                    hInst;
hInst = LoadLibrary(L"SmartVideoCoreDll.dll");
if(hInst == NULL)
{
  return;
}
m_pfEndEncoding = (EndEncodingEx)GetProcAddress(m_hInst, _T("EndEncoding"));
m_pfInitEncoding = (InitEncodingEx)GetProcAddress(m_hInst, _T("InitEncoding"));
m_pfRunEncoding = (RunEncodingEx)GetProcAddress(m_hInst, _T("RunEncoding"));
m_pfStopEncoding = (StopEncodingEx)GetProcAddress(m_hInst, T("StopEncoding"));

m_pfEndEncoding();   // 동영상 저장 잠시 중지
m_pfInitEncoding();  // 동영상 저장 초기화
m_pfRunEncoding();   // 동영상 저장 시작
m_pfStopEncoding();  // 동영상 저장 종료
FreeLibrary(hInst);
```

함수	GetCaptureResolutionEx

SetCaptureResolutionEx() 함수에서 설정된 영상(동영상, 정지 영상) 저장(Capture) 시 사용되는 가로/세로 해상도를 얻습니다.

◉ typedef void (*GetCaptureResolutionEx)(int* pWidth, int* pHeight)

[인자]

- int* pWidth : 가로 해상도
- int* pHeight : 세로 해상도

C++ 사용법

```
typedef void (*GetCaptureResolutionEx)(int* pWidth, int* pHeight);
GetCaptureResolutionEx       m_pfGetCaptureResolution;
```

```
HINSTANCE                      hInst;
hInst = LoadLibrary(L"SmartVideoCoreDll.dll");
if(hInst == NULL)
{
  return;
}
m_pfGetCaptureResolution = (GetCaptureResolutionEx)GetProcAddress(m_hInst,
                                  _T("GetCaptureResolution"));
int iWidth, iHeight;
// 설정된 Capture 해상도 얻기
m_pfGetCaptureResolution(&iWidth, &iHeight);
FreeLibrary(hInst);
```

함수	GetPreviewResolutionEx

SetPreviewResolutionEx() 함수에서 설정된 영상 출력(Preview) 화면의 가로/세로 해상도를 얻습니다.

◉ `typedef void (*GetPreviewResolutionEx)(int* pWidth, int* pHeight)`

[인자]

- `int* pWidth` : 가로 해상도
- `int* pHeight` : 세로 해상도

C++ 사용법

```
typedef void (*GetPreviewResolutionEx)(int* pWidth, int* pHeight);
GetPreviewResolutionEx         m_pfGetPreviewResolution;
HINSTANCE                      hInst;
hInst = LoadLibrary(L"SmartVideoCoreDll.dll");
if(hInst == NULL)
{
  return;
}
m_pfGetPreviewResolution = (GetPreviewResolutionEx)GetProcAddress(m_hInst,
                                  _T("GetPreviewResolution"));
int iWidth, iHeight;
// 설정된 영상 출력 해상도 얻기
m_pfGetCaptureResolution(&iWidth, &iHeight);
FreeLibrary(hInst);
```

함수	ImageMemoryCaptureEx

현재 영상의 정지 영상을 메모리로 Capture 합니다. 즉, 영상을 Capture 하여 이미지 Processing을 할 수 있습니다. 입력 인자를 통하여 정지 영상 이미지의 데이터를 Raw Bmp의 바이트 스트림(stream)으로 얻으며, Capture 시 적용되는 해상도는 SetCaptureResolutionEx() 함수에서 설정된 해상도로 저장됩니다.

[표] 해상도별 Capture 소요 시간

해상도	소요 시간	해상도	소요 시간
160 X 96	17.2ms	480 X 288	87.7ms
190 X 144	31.8ms	560 X 336	109.3ms
240 X 144	34.7ms	576 X 432	162.3ms

256 X 192	36.2ms	640 X 384	155.7ms
320 X 192	44.5ms	640 X 480	185.8ms
320 X 240	55.5ms	720 X 480	220.5ms
448 X 336	91.9ms	-	-

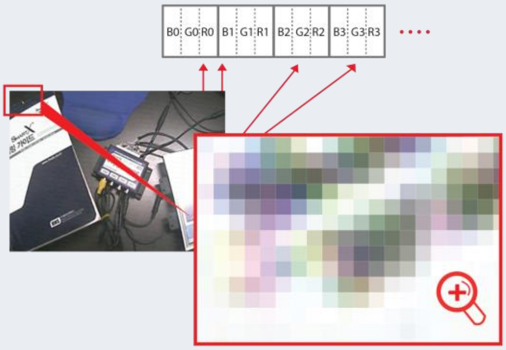
※ 위 측정 시간은 약간의 오차가 있을 수 있습니다.

◉ typedef bool (*ImageMemoryCaptureEx)(unsigned char* pRawByte)

[인자]

- unsigned char* pRawByte : 정지 영상을 Capture하여 얻어올 배열

중요 바이트 배열의 크기 및 멤버 순서



1. 바이트 배열의 크기

한 픽셀은 BGR 3개의 컬러로 이루어져 있습니다. 즉, 1 픽셀은 3Byte(24Bit)가 됩니다. 따라서 바이트 배열 크기 할당 시 전체 픽셀 수의 3배 크기를 할당해야 합니다.

※ 바이트 배열 크기 = 전체 픽셀 수 X 3

2. 바이트 배열의 멤버 순서

한 픽셀은 BGR 3개의 컬러로 이루어져 있으므로, 바이트 배열의 멤버는 B0(Blue), G0(Green), R0(Red), B1(Blue), G1(Green), R1(Red), …, B-n(Blue), G-n(Green), R-n(Red) 순서로 저장됩니다.

[리턴값]

- bool : 작업 성공 여부
  - true : 성공
  - false : 실패

C++ 사용법

```
typedef bool (*ImageMemoryCaptureEx)(unsigned char* pRawByte);
ImageMemoryCaptureEx          m_pfImageMemoryCapture;
HINSTANCE                     hInst;
hInst = LoadLibrary(L"SmartVideoCoreDll.dll");
if(hInst == NULL)
{
  return;
}
m_pfImageMemoryCapture = (ImageMemoryCaptureEx)GetProcAddress(m_hInst,
                              _T("ImageMemoryCapture"));
int iWidth = 640, iHeight = 480;
// 정지 영상의 크기 만큼 메모리 할당
unsigned char* Raw = new unsigned char[(iWidth * iHeight) * 3];
```

```
// 메모리로 정지 영상 얻기
m_pfImageMemoryCapture(Raw);
FreeLibrary(hInst);
```

함수 InitializationEx

SmartVideo를 사용하기 위하여 초기화 및 초기 영상 출력(Preview) 화면의 해상도를 설정합니다.

주의 초기값 설정은 처음 한번만 하시기 바랍니다.

◉ typedef bool (*InitializationEx)(int iPreviewResolution)

[인자]

- int iPreviewResolution: 초기 Preview의 해상도를 설정합니다.

참고 iPreviewResolution 인자값은 "SetPreviewResolutionEx() 함수" 내용을 참고하시기 바랍니다.

[리턴값]

- bool : SmartVideo 연결 여부
  - true : IEC-Series에 SmartVideo가 연결됨
  - false : IEC-Series에 SmartVideo 가 연결 안됨

C++ 사용법

```
typedef bool (*InitializationEx)(int iPreviewResolution);
InitializationEx      m_pfInitialization;
HINSTANCE             hInst;
hInst = LoadLibrary(L"SmartVideoCoreDll.dll");
if(hInst == NULL)
{
  return;
}
m_pfInitialization = (InitializationEx)GetProcAddress(m_hInst, _T("Initialization"));
// 영상 출력 화면 해상도 576 X 432
m_pfInitialization(11);
FreeLibrary(hInst);
```

함수 PlayEx, StopEx

SmartVideo의 초기화 후 비디오 영상 출력(Preview)을 시작 및 종료하도록 합니다.

◉ typedef bool (*PlayEx)()

◉ typedef bool (*StopEx)()

[리턴값]

- bool : 영상 출력 시작 및 종료 성공 여부
  - true : 성공
  - false : 실패

C++ 사용법

```
typedef bool (*PlayEx)();
typedef bool (*StopEx)();
PlayEx     m_pfPlay;
```

```
StopEx     m_pfStop;
HINSTANCE hInst;
hInst = LoadLibrary(L"SmartVideoCoreDll.dll");
if(hInst == NULL)
{
  return;
}
m_pfPlay = (PlayEx)GetProcAddress(m_hInst, _T("Play"));
m_pfStop = (StopEx)GetProcAddress(m_hInst, _T("Stop"));
m_pfPlay();  // 영상 출력 시작
m_pfStop();  // SmartVideo 종료
FreeLibrary(hInst);
```

함수 SetBrightnessControlEx, SetHueControlEx, SetContrastControlEx, SetColorSaturationControlEx

출력 영상의 속성(밝기, 색조, 명암, 채도) 값을 변경합니다.

- SetBrightnessControlEx() : 영상의 밝기를 제어합니다. (기본값 : 128)
- SetColorSaturationControlEx() : 영상의 채도를 제어합니다. (기본값 : 128)
- SetContrastControlEx() : 영상의 명암을 제어합니다. (기본값 : 128)
- SetHueControlEx() : 영상의 색조를 제어합니다. (기본값 : 0)

◉ typedef void (*SetBrightnessControlEx)(int iBrightness)

◉ typedef void (*SetColorSaturationControlEx)(int iColorSaturation)

◉ typedef void (*SetContrastControlEx)(int iContrast)

◉ typedef void (*SetHueControlEx)(int iHue)

[인자]

- int iBrightness : 밝기 (범위 : 0 ~ 255)
- int iColorSaturation : 채도 (범위 : 0 ~ 255)
- int iContrast : 명암 (범위 : 0 ~ 255)
- int iHue : 색조 (범위 : 0 ~ 255)

C++ 사용법

```
typedef void (*SetBrightnessControlEx)(int iBrightness);
typedef void (*SetColorSaturationControlEx)(int iColorSaturation);
typedef void (*SetContrastControlEx)(int iContrast);
typedef void (*SetHueControlEx)(int iHue);
SetBrightnessControlEx         m_pfSetBrightnessControl;
SetColorSaturationControlEx    m_pfSetColorSaturationControl;
SetContrastControlEx           m_pfSetContrastControl;
SetHueControlEx                m_pfSetHueControl;
HINSTANCE                      hInst;
hInst = LoadLibrary(L"SmartVideoCoreDll.dll");
if(hInst == NULL)
{
  return;
}
m_pfSetBrightnessControl = (SetBrightnessControlEx)GetProcAddress(m_hInst,
```

```
_T("SetBrightnessControl"));
m_pfSetColorSaturationControl = (SetColorSaturationControlEx)GetProcAddress(m_hInst,
                                    _T("SetColorSaturationControl"));
m_pfSetContrastControl = (SetContrastControlEx)GetProcAddress(m_hInst,
                             _T("SetContrastControl"));
m_pfSetHueControl = (SetHueControlEx)GetProcAddress(m_hInst, _T("SetHueControl"));
m_pfSetBrightnessControl(100);       // 밝기
m_pfSetColorSaturationControl(200);  // 채도
m_pfSetContrastControl(77);          // 명암
m_pfSetHueControl(33);               // 색조
FreeLibrary(hInst);
```

함수 SetCaptureResolutionEx

동영상 및 정지 영상의 저장(Capture) 해상도를 설정합니다.

주의	Preview 해상도가 720 X 480 이상인 경우 Capture 해상도는 640 X 480까지 설정 가능합니다. 160 X 96 크기로 저장된 동영상 파일은 PC에서 Play 되지 않습니다.

◉ typedef bool (*SetCaptureResolutionEx)(int iCaptureResolution)

[인자]

- int iCaptureResolution : Capture 시 가로 X 세로 해상도 값

인자값	해상도	인자값	해상도
0	160 X 96	7	480 X 288
1	192 X 144	8	560 X 336
2	240 X 144	9	576 X 432
3	256 X 192	10	640 X 384
4	320 X 192	11	640 X 480
5	320 X 240	12	720 X 480
6	448 X 336	-	-

[리턴값]

- bool : 작업 성공 여부
  - true : 성공
  - false : 실패

C++ 사용법

```
typedef bool (*SetCaptureResolutionEx)(int iCaptureResolution);
SetCaptureResolutionEx          m_pfSetCaptureResolution;
HINSTANCE                       hInst;
hInst = LoadLibrary(L"SmartVideoCoreDll.dll");
if(hInst == NULL)
{
  return;
}
m_pfSetCaptureResolution = (SetCaptureResolutionEx)GetProcAddress(m_hInst,
                               _T("SetCaptureResolution"));
// Capture 해상도를 320 X 240으로 설정
m_pfSetCaptureResolution(5);
FreeLibrary(hInst);
```

함수	SetDefaultControlEx

출력 영상의 밝기, 색조, 명암, 채도 설정 값을 기본 값으로 설정합니다.

참고 기본 설정 값

- SetBrightnessControlEx() : 밝기 128 설정
- SetColorSaturationControlEx() : 채도 128 설정
- SetContrastControlEx() : 명암 128 설정
- SetHueControlEx() : 색조 0 설정

"SetBrightnessControlEx(),SetColorSaturationControlEx(),SetContrastControlEx(), SetHueControlEx() 함수" 내용을 참고하시기 바랍니다.

◉ typedef void (*SetDefaultControlEx)()

C++ 사용법

```
typedef void (*SetDefaultControlEx)();
SetDefaultControlEx m_pfSetDefaultControl;
HINSTANCE           hInst;
hInst = LoadLibrary(L"SmartVideoCoreDll.dll");
if(hInst == NULL)
{
  return;
}
m_pfSetDefaultControl = (SetDefaultControlEx)GetProcAddress(m_hInst, _T("SetDefaultControl"));
m_pfSetDefaultControl();  // 영상의 밝기, 색조, 명암, 채도 설정 값을 초기 기본 설정값으로 설정
FreeLibrary(hInst);
```

함수	SetEncodingParameter1Ex, SetEncodingParameter2Ex

동영상을 저장하기 위하여 관련된 파라미터 값을 설정합니다. 동영상 Codec은 H.264로 저장됩니다.

◉ typedef void (*SetEncodingParameter1Ex)(LPTSTR strOutFilePathName, unsigned int iframe_rate, unsigned int ibitrate)

◉ typedef void (*SetEncodingParameter2Ex)(unsigned int igop_num, unsigned int iIntraqp, unsigned int iqpmax, float fgamma)

[인자]

- LPTSTR strOutFilePathName : 저장될 동영상 파일의 이름 및 경로
- unsigned int iframe_rate : 초당 프레임 수(설정 값 : 10, 15, 20(fps))
- unsigned int ibitrate : 비트 레이트(Bit Rate) (입력 범위 : 1 ~ 32767(kbps))

참고 비트 레이트(Bit Rate)란?

비트 레이트는 1초에 해당하는 동영상에 얼마의 비트(Bit) 수를 사용하는지를 의미합니다. 따라서 비트 레이트가 높을수록 동영상은 더 많은 정보(Bit)를 가지게 되므로 화질은 더 좋아지게 됩니다. 하지만 비트의 수는 그만큼 커지게 되므로 용량 또한 더 커지게 됩니다.

- unsigned int igop_num : 프레임 묶음 길이(입력 범위 : 0 ~ 60)

참고 프레임 묶음 길이란?

프레임 간 압축 방식은 프레임 전체 정보가 다 들어 있는 I 프레임을 기준으로 그 이후의 프레임은 이전 프레임 대비 변화량만을 기록하는 방식으로 데이터를 압축합니다.

변화량만을 기록하므로 움직임이 적을 경우 아주 적은 데이터만으로 한 프레임을 기록할 수 있습니다.

하지만 그렇다고 무작정 앞 프레임의 변화량을 저장하다가는 만약 데이터 유실에 치명적이고, 특정 위치의 프레임에 임의 접근 시 해당 프레임을 계산해 내는데 시간이 오래 걸리게 되므로 적절한 시간 간격으로 I 프레임을 삽입해 기준점을 다시 갱신해 주어야 합니다. 이렇게 하나의 I 프레임이 있으면 그 이후의 프레임은 특정 I 프레임에 종속적으로 됩니다. 따라서 하나의 I 프레임을 기준으로 다음 I 프레임이 나타날 때까지의 프레임들은 하나의 묶음으로 볼 수 있습니다.

이 묶음은 GOP라고 부르며, 묶음의 길이가 길수록 데이터 압축 효율은 올라가지만, 반응성은 떨어지고 데이터가 손실될 확률이 올라갑니다. 묶음의 길이가 짧을수록 데이터 압축 효율은 줄어들지만 반응성은 좋아지고 데이터 손실 확률이 낮아집니다.

[표] igop_num 인자 설정값에 따른 프레임 묶음

설정 값	설　　명	설정 값	설　　명
0	I, P, P, P,…	3	I, P, P, I, P, P, I, …:
1	I, I, I, I,…	60	I, P, P, P, P, P,…
2	I, P, I, P,…		

– I(Intra Picture) : 전 후의 화면과는 관계없이 그 화면 내에서 독립적으로 압축하여 얻어지는 화면(Picture). 즉 시간 방향의 움직임 예측을 적용하지 않으면서 화면 내 정보만을 사용해서 압축 처리를 수행합니다.
– P(Predictive Picture) : 화면 간의 순방향 예측 압축에 의해 얻어지는 화면이다. 즉 I Picture 또는 P Picture를 화면(프레임)간 예측에 사용하여 압축 처리를 수행합니다.

• `unsigned int iIntraqp` : Intra Picture의 양자화(Quantization) 스텝. 즉, 영상의 압축 정도를 설정하며 값이 클수록 압축률이 증가 (입력범위 : 0 ~ 51)

• `unsigned int iqpmax` : 최대 양자화 계수. iIntraqp와 같은 값으로 설정

• `float fgamma` : 감마 보정 값. 영상 신호의 입력 레벨에 따라서 0%의 검정에서 100%의 흰색까지 밝기가 변화하도록 직선적으로 실시하는 보정이며 영상데이터의 출력 즉 브라운관이나 액정에서 각각 최적의 감마 보정을 하여 자연스런 명암을 재현 할 수 있도록 설정 (기본값 : 0.75)

[표] 파라미터와 해상도에 따른 동영상 파일 크기

파라미터 설정 값	Bitrate : 700(kbps), GOPNum : 60, InterQP : 51, QPMax : 51		
해상도	Frame Rate-10	Frame Rate-15	Frame Rate-20
192 X 144	4.95MB	4.98MB	4.99MB
320 X 240	4.97MB	4.99MB	4.99MB
640 X 480	4.97MB	5.00MB	5.01MB

C++ 사용법

```
typedef void (*SetEncodingParameter1Ex)(LPTSTR strOutFilePathName, unsigned int iframe_rate,
unsigned int ibitrate);
typedef void (*SetEncodingParameter2Ex)(unsigned int igop_num, unsigned int iIntraqp, unsigned
int iqpmax, float fgamma);
SetEncodingParameter1Ex        m_pfSetEncodingParameter1;
SetEncodingParameter2Ex        m_pfSetEncodingParameter2;
HINSTANCE                      hInst;
hInst = LoadLibrary(L"SmartVideoCoreDll.dll");
if(hInst == NULL)
{
  return;
}
m_pfSetEncodingParameter1 = (SetEncodingParameter1Ex)GetProcAddress(m_hInst,
```

```
                              _T("SetEncodingParameter1"));
m_pfSetEncodingParameter2 = (SetEncodingParameter2Ex)GetProcAddress(m_hInst,
                              _T("SetEncodingParameter2"));
// 동영상 저장을 위한 관련 파라미터 설정
m_pfSetEncodingParameter1(L"SD Card\\HNS.264", 30, 3000);
m_pfSetEncodingParameter2(60, 15, 20, 0.75F);
FreeLibrary(hInst);
```

함수	SetPreviewResolutionEx

영상 출력(Preview) 해상도를 설정합니다.

주의 Preview 해상도가 720 X 480 이상인 경우 Capture 해상도는 640 X 480까지 설정 가능합니다.

◉ typedef bool (*SetPreviewResolutionEx)(int iPreviewResolution)

[인자]

- int iPreviewResolution : 영상의 가로 X 세로 해상도 값 설정

인자값	해상도	인자값	해상도
0	160 X 96	9	512 X 384
1	192 X 144	10	560 X 336
2	240 X 144	11	576 X 432
3	256 X 192	12	640 X 384
4	320 X 192	13	640 X 480
5	320 X 240	14	720 X 480
6	380 X 228	15	800 X 480
7	448 X 336	16	800 X 600
8	480 X 288	-	-

[리턴값]

- bool : 작업 성공 여부
  - true : 성공
  - false : 실패

C++ 사용법

```
typedef bool (*SetPreviewResolutionEx)(int iPreviewResolution);
SetPreviewResolutionEx          m_pfSetPreviewResolution;
HINSTANCE                       hInst;
hInst = LoadLibrary(L"SmartVideoCoreDll.dll");
if(hInst == NULL)
{
  return;
}
m_pfSetPreviewResolution = (SetPreviewResolutionEx)GetProcAddress(m_hInst,
                               _T("SetPreviewResolution"));
// 영상 해상도를 576 X 432로 설정
m_pfSetPreviewResolution(11);
FreeLibrary(hInst);
```

함수	SetPreviewWindowHandleEx

비디오 영상이 출력되는 윈도우의 핸들을 지정하여 지정된 Window, Control 등에 Preview 화면을 출력 하도록 설정합니다.

◉ typedef void (*SetPreviewWindowHandleEx)(HWND hPreviewWindow)

[인자]

- HWND hPreviewWindow : 컨트롤의 핸들

C++ 사용법

```
typedef void (*SetPreviewWindowHandleEx)(HWND hPreviewWindow);
SetPreviewWindowHandleEx        m_pfSetPreviewWindowHandle;
HINSTANCE                       hInst;
hInst = LoadLibrary(L"SmartVideoCoreDll.dll");
if(hInst == NULL)
{
  return;
}

m_pfSetPreviewWindowHandle = (SetPreviewWindowHandleEx)GetProcAddress(m_hInst,
                             _T("SetPreviewWindowHandle"));
// Preview Control 선언
CStatic m_CamPreView;
// Preview 창으로 사용할 Window 또는 컨트롤의 핸들을 설정
m_pfSetPreviewWindowHandle(m_CamPreView.m_hWnd);
FreeLibrary(hInst);
```

함수	SetSourceChannelEx

입력 채널을 선택 합니다. SmartVideo Board에는 총 4개의 채널이 있습니다. 이중 출력을 원하는 채널을 선택합니다. (하나의 채널만 출력이 가능하며, 동시 출력을 지원하지 않습니다.)

◉ typedef void (*SetSourceChannelEx)(int iChannel)

[인자]

- int iChannel : 출력할 채널
  - 1 : 1번 채널
  - 2 : 2번 채널
  - 3 : 3번 채널
  - 4 : 4번 채널

C++ 사용법

```
typedef void (*SetSourceChannelEx)(int iChannel);
SetSourceChannelEx   m_pfSetSourceChannel;
HINSTANCE            hInst;
hInst = LoadLibrary(L"SmartVideoCoreDll.dll");
if(hInst == NULL)
{
  return;
}
```

```
m_pfSetSourceChannel = (SetSourceChannelEx)GetProcAddress(m_hInst, _T("SetSourceChannel"));
// 비디오 신호 입력 채널 선택 : 1
m_pfSetSourceChannel(1);
FreeLibrary(hInst);
```

5) CSmartVideoWrap Class 설명

SmartVideoWrap Class는 사용할 인터페이스를 각각 선언 및 얻어오는 복잡한 과정을 거칠 필요 없이 한 번의 선언과 인스턴싱만으로 편리하게 SmartVideo의 기능을 C++(C++, Visual Studio 2005, 2008)에서 지원하도록 Wrapping 한 Class입니다. 본 Class를 사용하기 위해서는 "SmartVideoCoreWrap.h" 파일과 "SmartVideoCoreWrap.cpp" 파일을 대상 프로젝트에 복사한 후 #include "SmartVideoCoreWrap.h"을 호출하여 사용하시기 바랍니다.

인터페이스 및 사용 방식은 위에서 설명한 C++ 인터페이스와는 약간의 차이가 있으므로 다음의 내용을 참고하시어 C++에서 적용하시기 바랍니다.

5-1) SmartVideoWrap Class 사용하기

C++에서 SmartVideoWrap Class를 사용하기 위해서는 아래와 STEP을 진행해야 합니다.

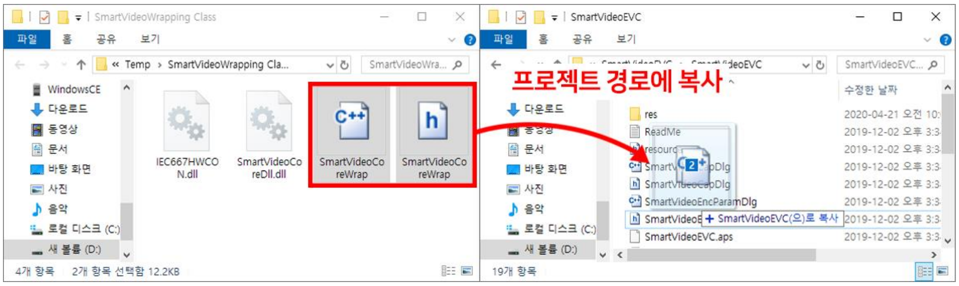
STEP-1	Wrap Class 파일 복사하기

HNS 홈페이지에서 사용하는 IEC-Series에 맞는 예제파일을 다운받은 후 압축을 풉니다.
"SmartVideoEVC" 폴더에서 아래 표를 확인하여 Wrap Class 파일을 프로젝트 경로에 복사합니다.

[표] IEC-Series별 Wrap Class 파일

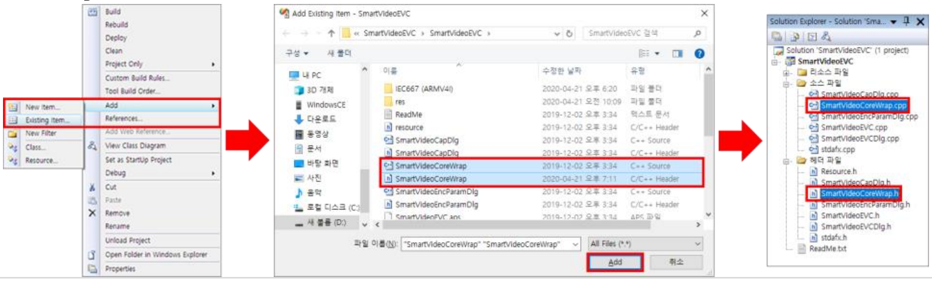
IEC-Series	IEC266-Series	IEC667-Series	IEC1000-Series
Wrap Class 파일	SmartVideoCoreWrap.h, SmartVideoCoreWrap.cpp		
다운로드 위치	홈페이지 → 자료실 → SmartX Framework 관련 → 6. SmartX Framework 예제파일 → Visual C++ 예제파일 → "SmartVideoEVC"		

Wrap Class 파일을 프로젝트 경로에 복사합니다.



STEP-2 Wrap Class 추가하기

복사한 Wrap Class 파일을 프로젝트에 추가합니다.



STEP-3 Wrap Class 사용하기

사용할 소스 파일에 #include〈SmartVideoCoreWrap.h〉 추가한 후 선언 및 인스턴스하여 사용합니다.

```
#include "SmartVideoCoreWrap.h"
// SmartVideo를 사용하기 위해 변수 선언 및 인스턴싱
CSmartVideoCoreWrap m_SmartVideo;
// Preview Control 선언
CStatic m_CamPreView;
// Preview 창으로 사용할 Window 또는 컨트롤의 핸들을 설정
m_pfSetPreviewWindowHandle(m_CamPreView.m_hWnd);
// 밝기
m_SmartVideo.SetBrightnessControl(100);
// 색조
m_SmartVideo.SetHueControl(33);
// 명암
m_SmartVideo.SetContrastControl(77);
// 채도
m_SmartVideo.SetColorSaturationControl(200);
// 미리보기 해상도 576 X 432
m_SmartVideo.Initialization(11);
// 동영상 저장을 위한 관련 파라미터 설정
m_SmartVideo.SetEncodingParameter1(L"SD Card\\HNS.264", 30, 3000);
m_SmartVideo.SetEncodingParameter2(60, 15, 20, 0.75F);
// 비디오 신호 입력 채널 선택 - 1
m_SmartVideo.SetSourceChannel(1);
// 영상 출력 시작
m_SmartVideo.Play();
// SmartVideo 종료
m_SmartVideo.Stop();
int iWidth, iHeight;
// 설정된 미리보기 해상도 얻기
m_SmartVideo.GetCaptureResolution(&iWidth, &iHeight);
// 정지 영상의 크기 만큼 메모리 할당
unsigned char* Raw = new unsigned char[(iWidth * iHeight) * 3];
// 메모리로 정지 영상 얻기
m_SmartVideo.ImageMemoryCapture(Raw);
```

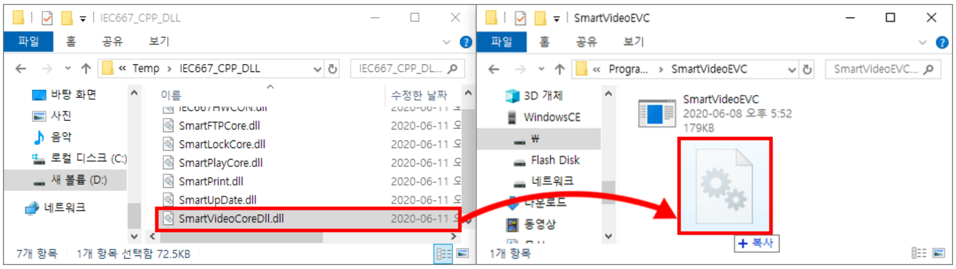
STEP-4	DLL 파일 복사하기

HNS 홈페이지에서 사용하는 IEC-Series에 맞는 예제파일을 다운받은 후 압축을 풉니다. "참조 DLL" 폴더에서 아래 표를 확인하여 사용하는 기능에 맞는 DLL 파일을 IEC-Series에 실행파일과 함께 복사합니다.

[표] IEC-Series별 DLL 파일과 다운로드 경로

IEC-Series	IEC266-Series	IEC667-Series	IEC1000-Series
DLL 파일	SmartVideoCoreDll.dll		
다운로드 위치	홈페이지 → 자료실 → SmartX Framework 관련 → 5. SmartX C++ For DLL 파일		

DLL 파일을 실행 파일 위치에 복사합니다.



참고	자세한 DLL 파일의 복사 경로는 "DLL 파일 및 복사 경로" 내용을 참고하시기 바랍니다.

5-2) SmartVideoWrap Class 예제 사용하기

SmartVideoWrap Class를 활용한 예제를 제공하고 있으며, 예제파일은 다음 경로에서 다운로드하여 참고하시기 바랍니다.

[예제 파일 다운로드 위치]

홈페이지(www.hnsts.co.kr) → 자료실 → SmartX Framework 예제파일 다운로드 → “SmartVideoEVC”

10. SmartUpdate

SmartX Framework는 기본적으로 .Net Compact Framework에서만 지원되도록 설계되어 있지만, 일부 기능을 C++에서 편리하게 사용할 수 있도록 외부 DLL 파일을 제공하고 있습니다. 따라서 SmartUpdate를 사용하기 위해서는 반드시 외부 DLL 파일을 실행 파일과 같이 IEC-Series에 배포해야 하며, "1. 인터페이스 함수 선언 → 2. 라이브러리 메모리 로드 처리 → 3. 인터페이스 함수 얻기" 과정을 거쳐야 합니다. 이때 IEC-Series 및 기능별로 로드해야 할 외부 DLL 파일이 다르므로 다음 내용을 확인하시기 바랍니다.

참조 SmartUpdate의 자세한 설명은 "홈페이지 → 자료실 → SmartX Framework 관련 → 프로그래밍 가이드 → SmartUpdate"를 참조하시기 바랍니다.

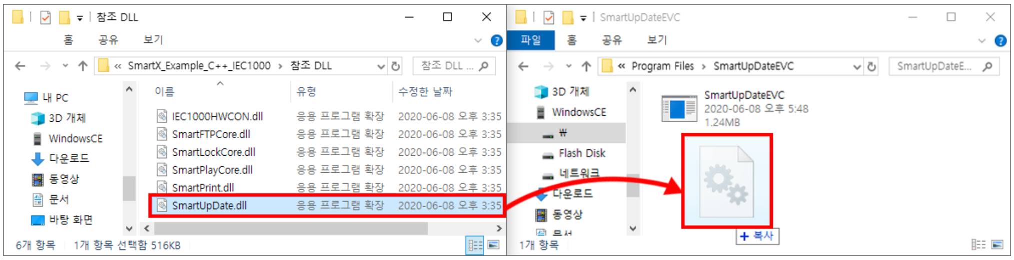
1) DLL 파일 및 복사 경로

HNS 홈페이지에서 사용하는 IEC-Series에 맞는 C++ For DLL 파일을 다운로드 받은 후 압축을 풉니다. 아래 표를 확인하여 DLL 파일을 IEC-Series에 실행 파일과 함께 복사합니다.

[표] IEC-Series별 DLL 파일과 다운로드 경로

IEC-Series	IEC266-Series	IEC667-Series	IEC1000-Series
DLL 파일	SmartUpDate.dll		
다운로드 위치	홈페이지 → 자료실 → SmartX Framework 관련 → 5. SmartX C++ For DLL 파일		

DLL 파일을 IEC-Series에 복사합니다. DLL 파일을 사용하는 실행 파일이 저장된 폴더에 저장하는 것을 권장하며, 다른 경로에 저장하는 경우 "인터페이스 선언 및 얻기"의 참고 내용을 확인하시기 바랍니다.
※ 개발 PC에서 프로그램을 배포하는 경우 실행 파일은 "Program File" 폴더에 배포됩니다.



2) 인터페이스 선언 및 얻기

인터페이스 사용을 위해 사용할 인터페이스의 선언 후 외부 DLL 파일을 메모리에 로드하여 인터페이스를 얻는 과정을 거쳐야 합니다. 이때 메모리 로드 처리를 위해 LoadLibrary() 함수 호출 시 상단 표의 DLL 파일의 이름을 인자로 설정합니다. 아래 코드는 예시이며, 필요로 하는 인터페이스 함수를 사용하기 위해서는 다음 내용을 참고하시기 바랍니다.

참고 DLL 파일 저장 경로에 따른 LoadLibrary() 함수 인자값

DLL 파일이 실행 파일과 같은 경로에 저장된 경우, LoadLibrary() 함수 인자값을 DLL 파일 이름만으로 설정해 호출할 수 있습니다. 만약 DLL 파일이 실행 파일과 다른 경로에 저장된 경우, LoadLibray() 함수의 인자값에는 저장된 DLL 파일의 전체 경로를 인자값으로 설정해야 합니다.

[C++] 인터페이스 얻기 및 사용하기 예시 코드

```
// 1. 인터페이스 함수 선언
typedef BOOL (*StartEx)();
StartEx  m_pfStart;
// 2. 라이브러리 메모리 로드 처리
HINSTANCE  m_hInst;
// SmartUpdate의 인터페이스를 사용하기 위해 SmartUpdate.dll 파일을 메모리에 로드
// IEC-Series 및 사용하는 기능별로 외부 DLL 파일이 다르며, 상단 표를 참고하시기 바랍니다.
m_hInst = LoadLibrary(L"SmartUpdate.dll");
// 외부 DLL 파일이 다른 경로(Flash Disk\\extdlls)에 저장된 경우
// m_hInst = LoadLibrary(L"Flash Disk\\extdlls\\SmartUpdate.dll");
if(m_hInst == NULL)
{
  // 메모리 로드 실패
  return FALSE;
}
// 3. 인터페이스 함수 얻기
m_pfStart = (StartEx)GetProcAddress(m_hInst, _T("Start"));
// 4. 함수 사용
m_pfStart();
```

참고 Wrapping Class 제공안내

사용할 인터페이스를 각각 선언 및 얻어오는 복잡한 과정을 거칠 필요 없이 한 번의 선언과 인스턴싱만으로 편리하게 SmartUpdate의 인터페이스를 사용할 수 있도록 Wrapping Class를 제공하고 있습니다.
자세한 내용은 "CSmartUpdateWrap Class 설명" 내용을 참고하시기 바랍니다.

3) C++ 인터페이스 설명

함수	SetPathEx

업데이트 경로를 설정합니다.

◉ typedef void (*SetPathEx)(LPWSTR strFromFile, LPWSTR strToFile)

[인자]

- LPWSTR strFromFile : 업데이트 파일이 있는 경로를 설정
- LPWSTR strToFile : 업데이트 될 대상 파일이 있는 경로를 설정

참고 strFromFile 경로의 파일이 기준이며, strToFile에 있는 모든 파일을 Checksum으로 비교하여 다른 경우 업데이트합니다. 만약 strFromFile의 파일이 strToFile에 없는 경우 그대로 복사합니다.

C++ 사용법

```
typedef void (*SetPathEx)(LPWSTR strFromFile, LPWSTR strToFile);
HINSTANCE                    m_hInst;
SetPathEx                    m_pfSetPath;
m_hInst = LoadLibrary(L"SmartUpdate.dll");
if(m_hInst == NULL)
{
    return FALSE;
}
m_pfSetPath = (SetPathEx)GetProcAddress(m_hInst, _T("SetPath"));
// 업데이트 매체 및 경로를 설정
m_pfSetPath("하드 디스크\\UpdateFiles", "Flash Disk\\Run");
```

함수 SetBackImagePathEx Image

업데이트 진행 시 다이얼로그 창 배경 이미지 파일의 경로 및 이름을 문자열로 설정 합니다. (이미지 확장자는 BMP만 가능)

◉ typedef void (*SetBackImagePathEx)(LPWSTR strImgFilePath)

[인자]

- LPWSTR strImgFilePath : 배경 이미지 파일의 경로 및 이름의 문자열

참고 SetBackImagePathEx() 함수에 이미지를 사용하지 않을 경우 기본 이미지로 출력됩니다.

배경이미지 설정(기본이미지 일 경우)

Image 이미지 사용 시 주의사항

사용되는 이미지가 "이미지 제작 가이드"에 맞게 제작되지 않은 경우, 다양한 오류가 발생할 수 있습니다.

※ 반드시 "이미지 제작 가이드"를 참고하여 이미지를 제작하시기 바랍니다.

C++ 사용법

```
typedef void (*SetBackImagePathEx)(LPWSTR strImgFilePath);
HINSTANCE                    m_hInst;
SetBackImagePathEx           m_pfSetBackImagePath;
m_hInst = LoadLibrary(L"SmartUpdate.dll");
if(m_hInst == NULL)
{
    return FALSE;
}
m_pfSetBackImagePath = (SetBackImagePathEx)GetProcAddress(m_hInst, _T("SetBackImagePath"));
// 업데이트 진행 시 화면에 출력될 배경 이미지의 경로를 설정
m_pfSetBackImagePath(L"Flash Disk\\UpBKImg.bmp");
```

함수 SetDisplayTextEx, SetDisplayTextColorEx

- SetDisplayTextEx() : 업데이트 진행 시 화면에 표시될 메시지의 텍스트를 설정합니다.
- SetDisplayTextColorEx() : 업데이트 진행 시 화면에 표시될 메시지 텍스트의 색상을 설정합니다.

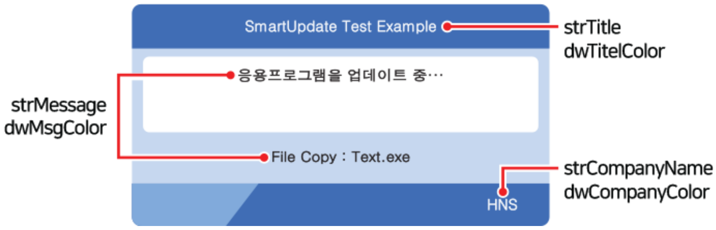
◉ typedef void (*SetDisplayTextEx)(LPWSTR strTitle, LPWSTR strMessage, LPWSTR strCompanyName)

◉ typedef void (*SetDisplayTextColorEx)(DWORD dwTitelColor, DWORD dwMsgColor,
DWORD dwCompanyColor)

[인자]

- LPWSTR strTitle : 상단 가운데 표시할 메시지의 텍스트를 설정
- LPWSTR strMessage : 중앙에 표시할 설명 메시지의 텍스트를 설정
- LPWSTR strCompanyName : 하단 우측에 표시할 메시지의 텍스트를 설정
- DWORD dwTitelColor : 상단 가운데 표시할 메시지 텍스트의 색상을 설정

- DWORD dwMsgColor : 중앙에 표시할 설명 메시지 텍스트의 색상을 설정
- DWORD dwCompanyColor : 하단 우측에 표시할 메시지 텍스트의 색상을 설정



C++ 사용법

```
typedef void (*SetDisplayTextEx)(LPWSTR strTitle, LPWSTR strMessage, LPWSTR strCompanyName);
typedef void (*SetDisplayTextColorEx)(DWORD dwTitelColor, DWORD dwMsgColor, DWORD
dwCompanyColor);
HINSTANCE                    m_hInst;
SetDisplayTextEx             m_pfSetDisplayText;
SetDisplayTextColorEx        m_pfSetDisplayTextColor;
m_hInst = LoadLibrary(L"SmartUpdate.dll");
if(m_hInst == NULL)
{
    return FALSE;
}
m_pfSetDisplayText = (SetDisplayTextEx)GetProcAddress(m_hInst, _T("SetDisplayText"));
m_pfSetDisplayTextColor = (SetDisplayTextColorEx)GetProcAddress(m_hInst,
_T("SetDisplayTextColor"));
// 업데이트 진행 시 화면에 출력되는 메시지의 텍스트를 설정
m_pfSetDisplayText(L"SmartUpdate Test Example", L"응용프로그램을 업데이트 중…", L"HNS");
// 메시지 텍스트의 색상을 설정
m_pfSetDisplayTextColor(RGB(0, 0, 0, RGB(0, 0, 0), RGB(0, 0, 0));
```

함수 SetStartIntervalEx

업데이트 시작의 지연 시간을 설정합니다.

업데이트하기 위해서 프로그램이 종료될 경우 프로그램의 용량 및 종료 관련 처리 코드로 인하여 바로 종료되지 않고 약간의 시간이 지난 후 프로그램이 종료됩니다. 업데이트를 정상적으로 시작할 수 없을 경우 설정합니다.

◉ typedef void (*SetStartIntervalEx)(DWORD dwInterval)

[인자]

- DWORD dwInterval : 업데이트 시작 지연 시간 (단위 : ms)

C++ 사용법

```
typedef void (*SetStartIntervalEx)(DWORD dwInterval);
HINSTANCE                    m_hInst;
SetStartIntervalEx           m_pfSetStartInterval;
m_hInst = LoadLibrary(L"SmartUpdate.dll");
if(m_hInst == NULL)
{
    return FALSE;
```

```
}
m_pfSetStartInterval = (SetStartIntervalEx)GetProcAddress(m_hInst, _T("SetStartInterval"));
// 업데이트 시작 지연 시간을 설정 (3000ms = 3초)
m_pfSetStartInterval(3000);
```

함수 SetRebootEx

업데이트 완료 후 IEC-Series를 재부팅 또는 업데이트된 새로운 장치 응용프로그램을 실행할지를 설정합니다. (단, 업데이트 내용이 없을 경우 해당 사항 없습니다.)

◉ typedef void (*SetRebootEx)(int iReboot)

[인자]

- int iReboot : 업데이트 완료 후 IEC-Series 동작
  - 0 : Run 폴더의 파일(EXE 파일)을 실행
  - 1 : IEC-Series을 재부팅

C++ 사용법

```
typedef void (*SetRebootEx)(int iReboot);
HINSTANCE                    m_hInst;
SetRebootEx                  m_pfSetReboot;
m_hInst = LoadLibrary(L"SmartUpdate.dll");
if(m_hInst == NULL)
{
    return FALSE;
}
m_pfSetReboot = (SetRebootEx)GetProcAddress(m_hInst, _T("SetReboot"));
// 업데이트 완료 후 IEC-Series 재부팅
m_pfSetReboot(1);
```

함수 StartEx

설정된 값에 따라 업데이트를 시작합니다. 업데이트할 파일이 있는 경우에 Update 모듈이 실행되어 본격적인 업데이트 작업이 시작됩니다.

◉ typedef BOOL (*StartEx)()

주의 업데이트할 파일이 있는 경우 반드시 현재 응용프로그램을 종료해야 합니다.

[리턴값]

- BOOL : 업데이트할 파일 유무
  - TURE : 있음
  - FALSE : 없음

C++ 사용법

```
typedef BOOL (*StartEx)();
HINSTANCE                    m_hInst;
StartEx                      m_pfStart;
m_hInst = LoadLibrary(L"SmartUpdate.dll");
if(m_hInst == NULL)
{
```

```
    return FALSE;
}
m_pfStart = (StartEx)GetProcAddress(m_hInst, _T("Start"));
// Update File을 확인하여 업데이트할 파일이 있을 경우 업데이트 시작
if(m_pfStart() == true)
{
  // 현재 응용프로그램을 종료
  CSmartUpDateEVCDlg::OnOK();
}
```

4) CSmartUpdateWrap Class 설명

SmartUpdateWrap Class는 사용할 인터페이스를 각각 선언 및 얻어오는 복잡한 과정을 거칠 필요 없이 한 번의 선언과 인스턴싱만으로 편리하게 SmartUpdate의 기능을 C++(C++, Visual Studio 2005, 2008)에서 지원하도록 Wrapping 한 Class입니다. 본 Class를 사용하기 위해서는 "SmartUpdateWrap.h" 파일과 "SmartUpdateWrap.cpp" 파일을 대상 프로젝트에 복사한 후 #include "SmartUpdateWrap.h"을 호출하여 사용하시기 바랍니다.

인터페이스 및 사용 방식은 위에서 설명한 C++ 인터페이스와는 약간의 차이가 있으므로 다음의 내용을 참고하시어 C++에서 적용하시기 바랍니다.

4-1) SmartUpdateWrap Class 사용하기

C++에서 SmartUpdateWrap Class를 사용하기 위해서는 아래와 STEP을 진행해야 합니다.

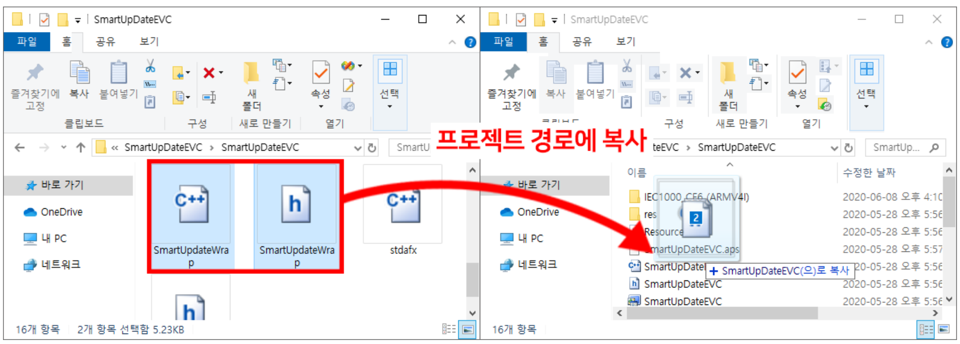
STEP-1	Wrap Class 파일 복사하기

HNS 홈페이지에서 사용하는 IEC-Series에 맞는 예제파일을 다운받은 후 압축을 풉니다.
"SmartUpDateEVC" 폴더에서 아래 표를 확인하여 Wrap Class 파일을 프로젝트 경로에 복사합니다.

[표] IEC-Series별 Wrap Class 파일

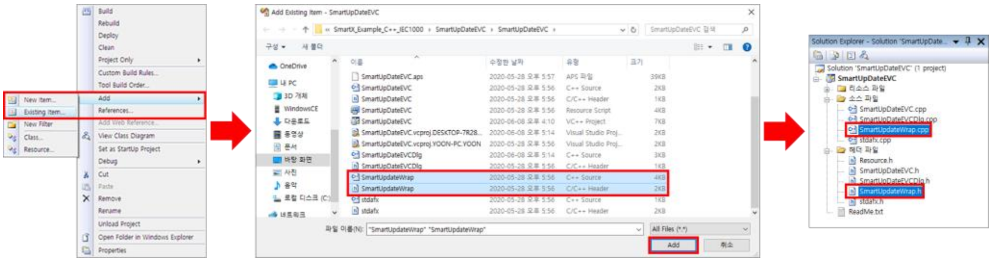
IEC-Series	IEC266-Series	IEC667-Series	IEC1000-Series
Wrap Class 파일	SmartUpdateWrap.h, SmartUpdateWrap.cpp		
다운로드 위치	홈페이지 → 자료실 → SmartX Framework 관련 → 6. SmartX Framework 예제파일 → Visual C++ 예제파일 → "SmartUpDateEVC"		

Wrap Class 파일을 프로젝트 경로에 복사합니다.



STEP-2 Wrap Class 추가하기

복사한 Wrap Class 파일을 프로젝트에 추가합니다.



STEP-3 Wrap Class 사용하기

사용할 소스 파일에 #include〈SmartUpdateWrap.h〉 추가한 후 선언 및 인스턴스하여 사용합니다.

```
#include "SmartUpdate.h"
// SmartUpdate를 사용하기 위해 변수 선언 및 인스턴싱
CSmartUpdateWrap      m_SmartUpdate;
// 업데이트 진행 시 화면에 출력되는 배경 이미지
m_SmartUpdate.SetBackImagePath(L"Flash Disk\\test1.bmp");
// 업데이트 매체 및 경로 설정
m_SmartUpdate.SetPath(L"Flash Disk\\UpdateFiles", L"Flash Disk\\Run");
// 업데이트 진행 시 화면에 출력되는 텍스트 설정
m_SmartUpdate.SetDisplayText(L"SmartUpdate Test Example", L"응용프로그램 업데이트 중…", L"HNS");
// 텍스트 색상 설정
m_SmartUpdate.SetDisplayTextColor(RGB(192, 0, 0), RGB(0, 192, 0), RGB(0, 0, 192));
// 업데이투 후 제품을 재부팅
m_SmartUpdate.SetReboot(1);
// 응용프로그램이 종료된 후 업데이트 시작까지의 지연 시간을 3초로 설정
m_SmartUpdate.SetStartInterval(3000);
if(m_SmartUpdate.Start() == true) // 업데이트할 파일이 있는 경우 업데이트를 시작
{
  CSmartUpDateEVCDlg::OnOK();// 현재 응용프로그램을 종료
}
```

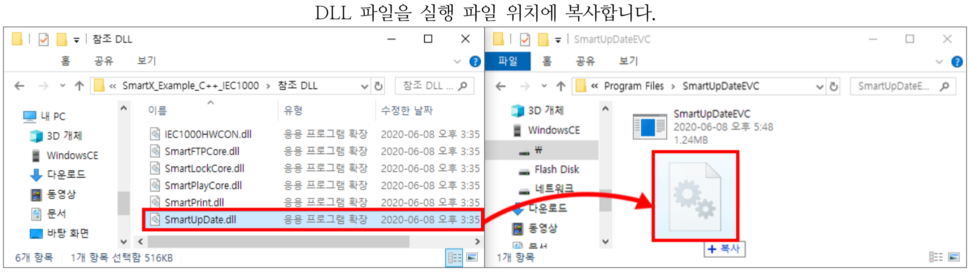
STEP-4 DLL 파일 복사하기

HNS 홈페이지에서 사용하는 IEC-Series에 맞는 예제파일을 다운받은 후 압축을 풉니다. "참조 DLL" 폴더에서 아래 표를 확인하여 사용하는 기능에 맞는 DLL 파일을 IEC-Series에 실행파일과 함께 복사합니다.

[표] IEC-Series별 DLL 파일과 다운로드 경로

IEC-Series	IEC266-Series	IEC667-Series	IEC1000-Series
DLL 파일	SmartUpDate.dll		
다운로드 위치	홈페이지 → 자료실 → SmartX Framework 관련 → 5. SmartX C++ For DLL 파일		

DLL 파일을 실행 파일 위치에 복사합니다.



참고	자세한 DLL 파일의 복사 경로는 "DLL 파일 및 복사 경로" 내용을 참고하시기 바랍니다.

4-2) SmartUpdateWrap Class 예제 사용하기

SmartUpdateWrap Class를 활용한 예제를 제공하고 있으며, 예제파일은 다음 경로에서 다운로드하여 참고하시기 바랍니다.

[예제 파일 다운로드 위치]
홈페이지(www.hnsts.co.kr) → 자료실 → SmartX Framework 예제파일 다운로드 → “SmartUpDateEVC”

11. SmartFTP

SmartX Framework는 기본적으로 .Net Compact Framework에서만 지원되도록 설계되어 있지만, 일부 기능을 C++에서 편리하게 사용할 수 있도록 외부 DLL 파일을 제공하고 있습니다. 따라서 SmartFTP를 사용하기 위해서는 반드시 외부 DLL 파일을 실행 파일과 같이 IEC-Series에 배포해야 하며, "1. 인터페이스 함수 선언 → 2. 라이브러리 메모리 로드 처리 → 3. 인터페이스 함수 얻기" 과정을 거쳐야 합니다. 이때 IEC-Series 및 기능별로 로드해야 할 외부 DLL 파일이 다르므로 다음 내용을 확인하시기 바랍니다.

참조 SmartFTP의 자세한 설명은 "홈페이지 → 자료실 → SmartX Framework 관련 → 프로그래밍 가이드 → SmartFTP"를 참조하시기 바랍니다.

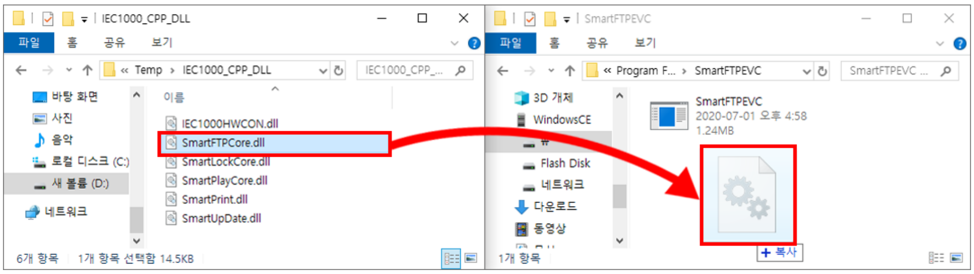
1) DLL 파일 및 복사 경로

HNS 홈페이지에서 사용하는 IEC-Series에 맞는 예제파일을 다운받은 후 압축을 풉니다. "참조 DLL" 폴더에서 아래 표를 확인하여 DLL 파일을 IEC-Series에 실행파일과 함께 복사합니다.

[표] IEC-Series별 DLL 파일과 다운로드 경로

IEC-Series	IEC266-Series	IEC667-Series	IEC1000-Series
DLL 파일	SmartFTPCore.dll		
다운로드 위치	홈페이지 → 자료실 → SmartX Framework 관련 → 6. SmartX Framework 예제파일 → Visual C++ 예제파일 → 참조 DLL		

DLL 파일을 IEC-Series에 복사합니다. DLL 파일을 사용하는 실행 파일이 저장된 폴더에 저장하는 것을 권장하며, 다른 경로에 저장하는 경우 "인터페이스 선언 및 얻기"의 참고 내용을 확인하시기 바랍니다.
※ 개발 PC에서 프로그램을 배포하는 경우 실행파일은 "Program File" 폴더에 배포됩니다.



2) 인터페이스 선언 및 얻기

인터페이스 사용을 위해 사용할 인터페이스의 선언 후 외부 DLL 파일을 메모리에 로드하여 인터페이스를 얻는 과정을 거쳐야 합니다. 이때 메모리 로드 처리를 위해 LoadLibrary() 함수 호출 시 상단 표의 DLL 파일의 이름을 인자로 설정합니다. 아래 코드는 예시이며, 필요로 하는 인터페이스 함수를 사용하기 위해서는 다음 내용을 참고하시기 바랍니다.

참고 DLL 파일 저장 경로에 따른 LoadLibrary() 함수 인자값

DLL 파일이 실행 파일과 같은 경로에 저장된 경우, LoadLibrary() 함수 인자값을 DLL 파일 이름만으로 설정해 호출할 수 있습니다. 만약 DLL 파일이 실행 파일과 다른 경로에 저장된 경우, LoadLibray() 함수의 인자값에는 저장된 DLL 파일의 전체 경로를 인자값으로 설정해야 합니다.

[C++] 인터페이스 얻기 및 사용하기 예시 코드

```
// 1. 인터페이스 함수 선언
typedef BOOL (*ConnectEx)();
ConnectEx  m_pfConnect;
// 2. 라이브러리 메모리 로드 처리
HINSTANCE  m_hInst;
// SmartFTP의 인터페이스를 사용하기 위해 SmartFTPCore.dll 파일을 메모리에 로드
// IEC-Series 및 사용하는 기능별로 외부 DLL 파일이 다르며, 상단 표를 참고하시기 바랍니다.
m_hInst = LoadLibrary(L"SmartFTPCore.dll");
// 외부 DLL 파일이 다른 경로(Flash Disk\\extdlls)에 저장된 경우
// m_hInst = LoadLibrary(L"Flash Disk\\extdlls\\SmartFTPCore.dll");
if(m_hInst == NULL)
{
  // 메모리 로드 실패
  return FALSE;
}
// 3. 인터페이스 함수 얻기
m_pfConnect = (ConnectEx)GetProcAddress(m_hInst, _T("Connect"));
// 4. 함수 사용
m_pfConnect();
```

참고 Wrapping Class 제공안내

사용할 인터페이스를 각각 선언 및 얻어오는 복잡한 과정을 거칠 필요없이 한 번의 선언과 인스턴싱만으로 편리하게 SmartFTP의 인터페이스를 사용할 수 있도록 Wrapping Class를 제공하고 있습니다.
자세한 내용은 “CSmartFTPWrap Class 설명” 내용을 참고하시기 바랍니다.

3) C++ 인터페이스 설명

함수	SetConnectInfoEx

FTP 서버 접속에 필요한 정보들을 설정합니다.

◉ `typedef void (*SetConnectInfoEx) (LPTSTR strFtpAddress, unsigned int iPortNo, LPTSTR strUserID, LPTSTR strPassword, BOOL bPassiveMode)`

[인자]

- `LPTSTR strFtpAddress` : FTP 서버 주소(URL or IP)
- `unsigned int iPortNo` : 포트 번호
- `LPTSTR strUserID` : 사용자 계정
- `LPTSTR strPassword` : 비밀번호
- `BOOL bPassiveMode` : PassiveMode 사용 여부
  - TRUE : 사용
  - FALSE : 사용 안함

C++ 사용법

```
typedef void (*SetConnectInfoEx)(LPTSTR strFtpAddress, unsigned int iPortNo, LPTSTR strUserID,
LPTSTR strPassword, BOOL bPassiveMode);
HINSTANCE           m_hInst;
SetConnectInfoEx    m_pfSetConnectInfo;
m_hInst = LoadLibrary(L"SmartFTPCore.dll");
if(m_hInst == NULL)
{
  return FALSE;
```

```
}
m_pfSetConnectInfo = (SetConnectInfoEx)GetProcAddress(m_hInst, _T("SetConnectInfo"));
m_pfSetConnectInfo(L"192.168.1.14", 21, L"hns", L"hns", TRUE);
```

함수 ConnectEx, DisConnectEx

- ConnectEx() : FTP 서버에 연결합니다.
- DisConnectEx() : FTP 서버와 연결을 해제합니다.

◉ typedef BOOL (*ConnectEx)()

◉ typedef BOOL (*DisConnectEx)()

[리턴값]

- BOOL : 연결 및 연결 해제 성공 여부
  - TRUE : 성공
  - FALSE : 실패

C++ 사용법

```
typedef BOOL (*ConnectEx)();
typedef BOOL (*DisConnectEx)();
HINSTANCE          m_hInst;
ConnectEx          m_pfConnect;
DisConnectEx       m_pfDisConnect;
m_hInst = LoadLibrary(L"SmartFTPCore.DLL");
if(m_hInst == NULL)
{
  return FALSE;
}
m_pfConnect = (ConnectEx)GetProcAddress(m_hInst, _T("Connect"));
m_pfDisConnect = (DisConnectEx)GetProcAddress(m_hInst, _T("DisConnect"));
// FTP 서버 연결
if(m_pfConnect() == TRUE)
{
  // 연결 성공 시 연결 해제
  m_pfDisConnect()
}
```

함수 SetRootDirectoryEx

SetCurrentDirectoryEx() 함수에서 설정된 경로를 Root Directory로 설정합니다.

◉ typedef BOOL (*SetRootDirectoryEx)()

[리턴값]

- BOOL : 설정 성공 여부
  - TRUE : 성공
  - FALSE : 실패

C++ 사용법

```
typedef BOOL (*SetRootDirectoryEx)();
HINSTANCE          m_hInst;
```

```
SetRootDirectoryEx   m_pfSetRootDirectory;
m_hInst = LoadLibrary(L"SmartFTPCore.DLL");
if(m_hInst == NULL)
{
  return FALSE;
}
m_pfSetRootDirectory = (SetRootDirectoryEx)GetProcAddress(m_hInst, _T("SetRootDirectory"));
if(m_pfSetRootDirectory("FTP//FTPTEST") == TRUE)
{
  // 설정 성공
}
```

함수	SetCurrentDirectoryEx, GetCurrentDirectoryEx

- SetCurrentDirectoryEx() : 업/다운로드 경로를 설정합니다.
- GetCurrentDirectoryEx() : 설정된 업/다운로드 경로를 얻습니다.

◉ typedef BOOL (*SetCurrentDirectoryEx)(LPTSTR strPath)

◉ typedef BOOL (*GetCurrentDirectoryEx)(LPTSTR strPath)

[인자]

- LPTSTR strPath : 설정할 경로 또는 얻어온 경로

[리턴값]

- BOOL : 설정 및 얻기 성공 여부
  - TRUE : 성공
  - FALSE : 실패

C++ 사용법

```
typedef BOOL (*SetCurrentDirectoryEx)(LPTSTR strPath)
typedef BOOL (*GetCurrentDirectoryEx)(LPTSTR strPath)
HINSTANCE                      m_hInst;
SetCurrentDirectoryEx          m_pfSetCurrentDirectory;
GetCurrentDirectoryEx          m_pfGetCurrentDirectory;
m_hInst = LoadLibrary(L"SmartFTPCore.DLL");
if(m_hInst == NULL)
{
  return FALSE;
}
m_pfSetCurrentDirectory = (SetCurrentDirectoryEx)GetProcAddress(m_hInst,
                          _T("SetCurrentDirectory"));
m_pfGetCurrentDirectory = (GetCurrentDirectoryEx)GetProcAddress(m_hInst,
                          _T("GetCurrentDirectory"));
m_pfSetCurrentDirectory("FTP//FTPTEST");
TCHAR strDirectory[500];
m_pfGetCurrentDirectory(strDirectory);
```

함수	CreateFolderEx, DeleteFolderEx, DeleteFilesEx

- CreateFolderEx() : SetCurrentDirectory() 함수에서 설정된 경로에 폴더를 생성합니다.
- DeleteFolderEx() : SetCurrentDirectory() 함수에서 설정된 경로의 폴더를 삭제합니다.
- DeleteFilesEx() : SetCurrentDirectory() 함수에서 설정된 경로의 파일을 삭제합니다.

◉ typedef BOOL (*CreateFolderEx)(LPTSTR strFolderName)

◉ typedef BOOL (*DeleteFolderEx)(LPTSTR strFolderName)

◉ typedef BOOL (*DeleteFilesEx)(LPTSTR strFileName)

[인자]

- LPTSTR strFolderName : 생성 및 삭제할 폴더 이름
- LPTSTR strFileName : 삭제할 파일 이름

[리턴값]

- BOOL : 생성 및 삭제 성공 여부
  - TRUE : 성공
  - FALSE : 실패

C++ 사용법

```
typedef BOOL (*CreateFolderEx)(LPTSTR strFolderName);
typedef BOOL (*DeleteFolderEx)(LPTSTR strFolderName);
typedef BOOL (*DeleteFilesEx)(LPTSTR strFileName);
HINSTANCE           m_hInst;
CreateFolderEx      m_pfCreateFolder;
DeleteFolderEx      m_pfDeleteFolder;
DeleteFilesEx       m_pfDeleteFiles;
m_hInst = LoadLibrary(L"SmartFTPCore.DLL");
if(m_hInst == NULL)
{
  return FALSE;
}
m_pfCreateFolder = (CreateFolderEx)GetProcAddress(m_hInst, _T("CreateFolder"));
m_pfDeleteFolder = (DeleteFolderEx)GetProcAddress(m_hInst, _T("DeleteFolder"));
m_pfDeleteFiles = (DeleteFilesEx)GetProcAddress(m_hInst, _T("DeleteFiles"));
// 설정된 경로에 하위 폴더를 만듬
if(m_pfCreateFolder("TestFoder") = TRUE)
{
  // 만들어진 하위 폴더를 지움
  m_pfDeleteFolder("TestFoder");
}
// 설정된 경로의 특정 파일을 지움
m_pfDeleteFiles("Test.dat");
```

함수	ReNameEx

특정 경로의 파일 및 폴더의 이름을 변경 합니다.

◉ typedef BOOL (*ReNameEx)(LPTSTR strOldName, LPTSTR strNewName)

[인자]

• LPTSTR strOldName : 변경할 현재 파일 및 폴더의 이름
• LPTSTR strNewName : 새로 변경할 이름

[리턴값]

• BOOL : 이름 변경 성공 여부
  - TRUE : 성공
  - FALSE : 실패

C++ 사용법

```
typedef BOOL (*ReNameEx)(LPTSTR strOldName, LPTSTR strNewName);
HINSTANCE           m_hInst;
ReNameEx            m_pfReName;
m_hInst = LoadLibrary(L"SmartFTPCore.DLL");
if(m_hInst == NULL)
{
  return FALSE;
}
m_pfReName = (ReNameEx)GetProcAddress(m_hInst, _T("ReName"));
m_pfReName("FTP//FTPTEST", "FTP//SMARTFTP");  // 폴더 이름 변경
m_pfReName("FTP//SMARTFTP//123.DAT", "FTP//SMARTFTP//ABC.DAT"); // 파일 이름 변경
```

함수	FileDownloadEx, FileUploadEx

• FileDownloadEx() : FTP 서버로부터 지정된 경로의 파일을 다운로드합니다.
• FileUploadEx() : IEC-Series의 파일을 FTP 서버의 지정된 경로에 업로드합니다.

주의 FileDownloadEx(), FileUploadEx() 함수 사용 시 주의사항

1. strFTPFilePathName 인자는 "//"로 경로를 구분합니다.
2. strLocalFilePathName 인자는 "₩₩"로 경로를 구분합니다.
3. 대소문자 및 서버측의 방화벽이 체크되었는지 확인합니다.
4. 업로드 진행 중 다운로드 실행 시 진행 중이던 업로드 작업이 취소됩니다. (반대의 경우도 동일)

중요 FileDownload와 FileUpload는 Non-Blocking 함수입니다. 함수가 호출즉시 리턴하는 구조로 되어 있습니다. 따라서 업/다운로드 진행 상태를 확인하는 방법이 필요하게 됩니다.
타이머에서 주기적으로 GetPercentageEx() 함수의 리턴 값을 확인하면 현재 업/다운로드 상태를 확인할 수 있습니다.

참고 “GetPercentageEx() 함수” 내용을 참고하시기 바랍니다.

◉ typedef BOOL (*FileDownloadEx)(LPTSTR strFTPFilePathName, LPTSTR strLocalFilePathName)
◉ typedef BOOL (*FileUploadEx)(LPTSTR strLocalFilePathName, LPTSTR strFTPFilePathName)

[인자]

• LPTSTR strFTPFilePathName : FTP 서버의 경로 및 파일 이름
• LPTSTR strLocalFilePathName : IEC-Series의 Local 경로 및 파일 이름

[리턴값]

• BOOL : 다운 및 업로드 성공 여부
  - TRUE : 성공
  - FALSE : 실패

C++ 사용법

```
typedef BOOL (*FileDownloadEx)(LPTSTR strFTPFilePathName, LPTSTR strLocalFilePathName);
typedef BOOL (*FileUploadEx)(LPTSTR strLocalFilePathName, LPTSTR strFTPFilePathName);
HINSTANCE           m_hInst;
FileDownloadEx      m_pfFileDownload;
FileUploadEx        m_pfFileUpload;
m_hInst = LoadLibrary(L"SmartFTPCore.DLL");
if(m_hInst == NULL)
{
  return FALSE;
}
m_pfFileDownload = (FileDownloadEx)GetProcAddress(m_hInst, _T("FileDownload"));
m_pfFileUpload = (FileUploadEx)GetProcAddress(m_hInst, _T("FileUpload"));
m_pfFileDownload("//123.DAT", "Flash Disk\\123.DAT");
m_pfFileUpload("SD Card\\123\\ABC.DAT", "FTP\\SMARTFTP\\ABC.DAT");
```

함수	FileUpNDownloadCancelEx

파일의 업/다운로드 작업을 취소합니다.

◉ typedef void (*FileUpNDownloadCancelEx)()

C++ 사용법

```
typedef void (*FileUpNDownloadCancelEx)();
HINSTANCE                   m_hInst;
FileUpNDownloadCancelEx     m_pfFileUpNDownloadCancel;
m_hInst = LoadLibrary(L"SmartFTPCore.DLL");
if(m_hInst == NULL)
{
  return FALSE;
}
m_pfFileUpNDownloadCancel = (FileUpNDownloadCancelEx)GetProcAddress(m_hInst,
                            _T("FileUpNDownloadCancel"));
m_pfFileUpNDownloadCancel();
```

함수	GetPercentageEx

파일의 업/다운로드 시 진행 상태를 백분율 값으로 얻습니다. 또는 실패 유무를 확인합니다.

참고	FileDownloadEx() 또는 FileUploadEx() 함수 호출 후 타이머를 사용해 진행 상황을 확인할 수 있습니다. 자세한 내용은 “FileDownloadEx(), FileUploadEx() 함수”를 참고하시기 바랍니다.

◉ typedef int (*GetPercentageEx)()

[리턴값]

- int : 다운 및 업로드 진행 비율 (-1 : 실패)

C++ 사용법

```
typedef int (*GetPercentageEx)();
HINSTANCE           m_hInst;
GetPercentageEx     m_pfGetPercentage;
```

```
m_hInst = LoadLibrary(L"SmartFTPCore.DLL");
if(m_hInst == NULL)
{
  return FALSE;
}
m_pfGetPercentage = (GetPercentageEx)GetProcAddress(m_hInst, _T("GetPercentage"));
int iPercent = m_pfGetPercentage();
if(iPercent == -1)
{
  // 업, 다운로드 실패
}
```

함수	GetFileListEx

SetCurrentDirectoryEx() 함수에서 설정된 경로의 모든 파일과 폴더 리스트를 얻습니다.

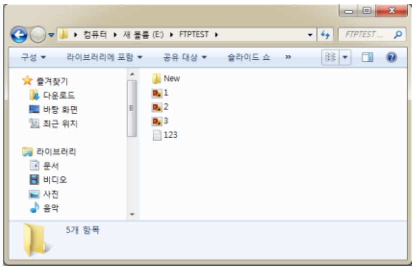
◉ typedef BOOL (*GetFileListEx)(LPTSTR strRetList)

[인자]

- LPTSTR strRetList : 현재 설정된 경로의 모든 파일과 폴더명

중요 strRetList 리턴 형식

GetFileListEx() 메소드는 인자값으로 설정한 strRetList에 다음과 같은 형식의 문자열을 리턴합니다.
또한, strRetList 인자의 배열 크기는 3000으로 설정하시기 바랍니다.
[저장 형식]
파일이름.확장자 : … 중략 … 폴더이름W:파일이름.확장자: …

<table>
<tr><th>FTP 서버의 경로</th><th>GetFileListEx() 리턴값</th></tr>
<tr><td>
</td><td>
</td></tr>
</table>

[리턴값]

- BOOL : 생성 및 삭제 성공 여부
  - TRUE : 성공
  - FALSE : 실패

C++ 사용법

```
typedef BOOL (*GetFileListEx)(LPTSTR strRetList);
HINSTANCE          m_hInst;
GetFileListEx      m_pfGetFileList;
m_hInst = LoadLibrary(L"SmartFTPCore.DLL");
if(m_hInst == NULL)
{
  return FALSE;
```

```
}
m_pfGetFileList = (GetFileListEx)GetProcAddress(m_hInst, _T("GetFileList"));
TCHAR     strListFileList[3000];
if(m_pfGetFileList(strListFileList) == TRUE)
{
  // 파일 및 폴더 리스트 얻기 성공
}
```

4) CSmartFTPWrap Class 설명

SmartFTPWrap Class는 사용할 인터페이스를 각각 선언 및 얻어오는 복잡한 과정을 거칠 필요없이 한 번의 선언과 인스턴싱만으로 편리하게 SmartFTP의 기능을 C++(C++, Visual Studio 2005, 2008)에서 지원하도록 Wrapping 한 Class입니다. 본 Class를 사용하기 위해서는 “SmartFTPWrap.h” 파일과 “SmartFTPWrap.cpp” 파일을 대상 프로젝트에 복사한 후 #include "SmartFTPWrap.h"을 호출하여 사용하시기 바랍니다.

인터페이스 및 사용 방식은 위에서 설명한 C++ 인터페이스와는 약간의 차이가 있으므로 다음의 내용을 참고하시어 C++에서 적용하시기 바랍니다.

4-1) SmartFTPWrap Class 사용하기

C++에서 SmartFTPWrap Class를 사용하기 위해서는 아래와 STEP을 진행해야 합니다.

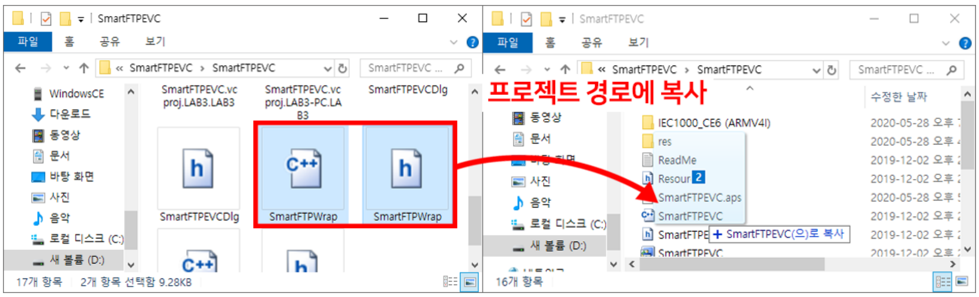
STEP-1	Wrap Class 파일 복사하기

HNS 홈페이지에서 사용하는 IEC-Series에 맞는 예제파일을 다운받은 후 압축을 풉니다.
“SmartFTPEVC” 폴더에서 아래 표를 확인하여 Wrap Class 파일을 프로젝트 경로에 복사합니다.

[표] IEC-Series별 Wrap Class 파일

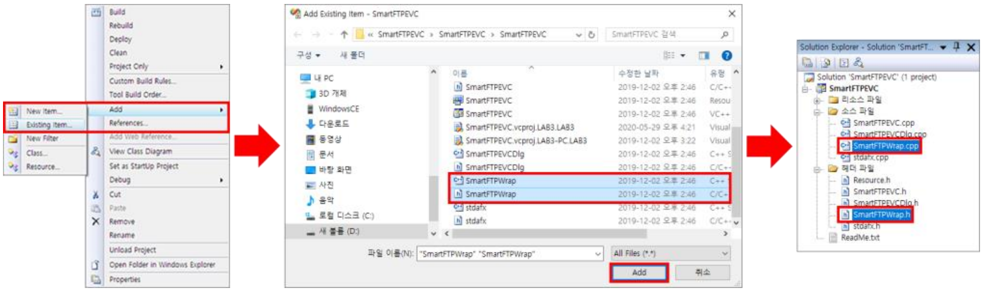
IEC-Series	IEC266-Series	IEC667-Series	IEC1000-Series
Wrap Class 파일	SmartFTPWrap.h, SmartFTPWrap.cpp		
다운로드 위치	홈페이지 → 자료실 → SmartX Framework 관련 → 6. SmartX Framework 예제파일 → Visual C++ 예제파일 → "SmartFTPEVC"		

Wrap Class 파일을 프로젝트 경로에 복사합니다.



STEP-2 Wrap Class 추가하기

복사한 Wrap Class 파일을 프로젝트에 추가합니다.



STEP-3 Wrap Class 사용하기

사용할 소스 파일에 #include〈SmartFTPWrap.h〉 추가한 후 선언 및 인스턴스하여 사용합니다.

```
#include "SmartFTPWrap.h"
// SmartFTP를 사용하기 위해 변수 선언 및 인스턴싱
CSmartFTPWrap m_SmartFTP;
// FTP 서버에 접속하기 위해 관련 정보를 설정
m_SmartFTP.SetConnectInfo(L"192.168.1.14", 21, L"hns", L"hns", TRUE);
// FTP 서버 연결
m_SmartFTP.Connect();
// 기본 경로 설정
m_SmartFTP.SetRootDirectory("FTP//FTPTEST");
// 설정된 경로에 하위 폴더를 만듦
m_SmartFTP.CreateFolder("TestFoder");
// 설정된 경로의 특정 하위 폴더를 지움
m_SmartFTP.DeleteFolder("TestFoder");
// 설정된 경로의 특정 파일을 지움
m_SmartFTP.DeleteFiles("Test.dat");
// 입력된 경로의 파일을 다운로드 시작
m_SmartFTP.FileDownload("//123.DAT", "Flash Disk\\123.DAT");
// 입력된 경로의 파일의 업로드를 시작
m_SmartFTP.FileUpload("SD Card\\123\\ABC.DAT", "FTP\\SMARTFTP\\ABC.DAT");
// 다운로드 및 업로드 진행을 취소
m_SmartFTP.FileUpNDownloadCancel();
// 현재 다운 및 업로드 진행 상황을 확인
int iPercent = m_SmartFTP.GetPercentage();
```

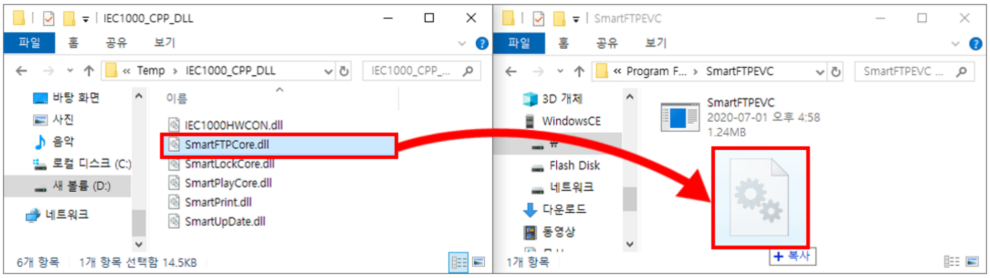
STEP-4 DLL 파일 복사하기

HNS 홈페이지에서 사용하는 IEC-Series에 맞는 예제파일을 다운받은 후 압축을 풉니다. "참조 DLL" 폴더에서 아래 표를 확인하여 사용하는 기능에 맞는 DLL 파일을 IEC-Series에 실행파일과 함께 복사합니다.

[표] IEC-Series별 DLL 파일과 다운로드 경로

IEC-Series	IEC266-Series	IEC667-Series	IEC1000-Series
DLL 파일	SmartFTPCore.dll		
다운로드 위치	홈페이지 → 자료실 → SmartX Framework 관련 → 6. SmartX Framework 예제파일 → Visual C++ 예제파일 → 참조 DLL		

DLL 파일을 실행 파일 위치에 복사합니다.



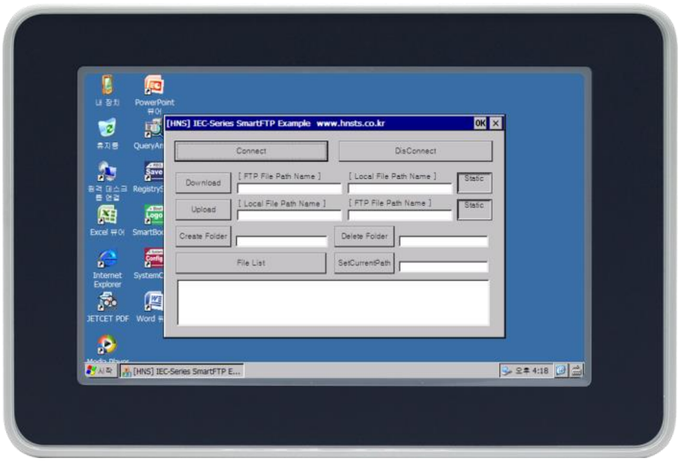
참고 자세한 DLL 파일의 복사 경로는 "DLL 파일 및 복사 경로" 내용을 참고하시기 바랍니다.

4-2) SmartFTPWrap Class 예제 사용하기

SmartFTPWrap Class를 활용한 예제를 제공하고 있으며, 예제파일은 다음 경로에서 다운로드하여 참고하시기 바랍니다.

[예제 파일 다운로드 위치]

홈페이지(www.hnsts.co.kr) → 자료실 → SmartX Framework 예제파일 다운로드 → “SmartFTPEVC”



12. SmartPlayer

SmartX Framework는 기본적으로 .Net Compact Framework에서만 지원되도록 설계되어 있지만, 일부 기능을 C++에서 편리하게 사용할 수 있도록 외부 DLL 파일을 제공하고 있습니다. 따라서 SmartPlayer를 사용하기 위해서는 반드시 외부 DLL 파일을 실행 파일과 같이 IEC-Series에 배포해야 하며, "1. 인터페이스 함수 선언 → 2. 라이브러리 메모리 로드 처리 → 3. 인터페이스 함수 얻기" 과정을 거쳐야 합니다. 이때 IEC-Series 및 기능별로 로드해야 할 외부 DLL 파일이 다르므로 다음 내용을 확인하시기 바랍니다.

참조 SmartPlayer의 자세한 설명은 "홈페이지 → 자료실 → SmartX Framework 관련 → 프로그래밍 가이드 → SmartPlayer"를 참조하시기 바랍니다.

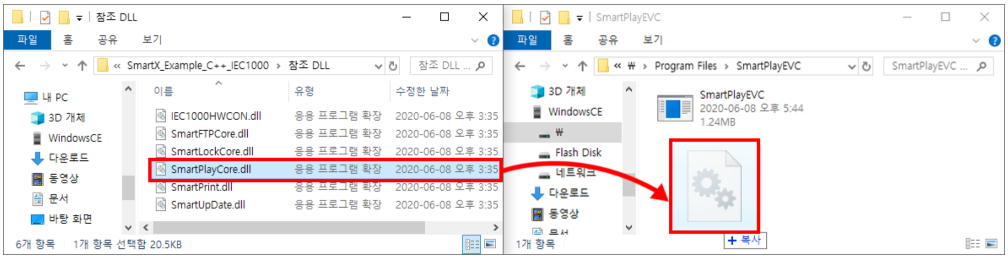
1) DLL 파일 및 복사 경로

HNS 홈페이지에서 사용하는 IEC-Series에 맞는 C++ For DLL 파일을 다운로드 받은 후 압축을 풉니다. 아래 표를 확인하여 DLL 파일을 IEC-Series에 실행 파일과 함께 복사합니다.

[표] IEC-Series별 DLL 파일과 다운로드 경로

IEC-Series	IEC266-Series	IEC667-Series	IEC1000-Series
DLL 파일	SmartPlayCore.dll		
다운로드 위치	홈페이지 → 자료실 → SmartX Framework 관련 → 5. SmartX C++ For DLL 파일		

DLL 파일을 IEC-Series에 복사합니다. DLL 파일을 사용하는 실행 파일이 저장된 폴더에 저장하는 것을 권장하며, 다른 경로에 저장하는 경우 "인터페이스 선언 및 얻기"의 참고 내용을 확인하시기 바랍니다.
※ 개발 PC에서 프로그램을 배포하는 경우 실행 파일은 "Program File" 폴더에 배포됩니다.



2) 인터페이스 선언 및 얻기

인터페이스 사용을 위해 사용할 인터페이스의 선언 후 외부 DLL 파일을 메모리에 로드하여 인터페이스를 얻는 과정을 거쳐야 합니다. 이때 메모리 로드 처리를 위해 LoadLibrary() 함수 호출 시 상단 표의 DLL 파일의 이름을 인자로 설정합니다. 아래 코드는 예시이며, 필요로 하는 인터페이스 함수를 사용하기 위해서는 다음 내용을 참고하시기 바랍니다.

참고 DLL 파일 저장 경로에 따른 LoadLibrary() 함수 인자값

DLL 파일이 실행 파일과 같은 경로에 저장된 경우, LoadLibrary() 함수 인자값을 DLL 파일 이름만으로 설정해 호출할 수 있습니다. 만약 DLL 파일이 실행 파일과 다른 경로에 저장된 경우, LoadLibray() 함수의 인자값에는 저장된 DLL 파일의 전체 경로를 인자값으로 설정해야 합니다.

[C++] 인터페이스 얻기 및 사용하기 예시 코드

```
// 1. 인터페이스 함수 선언
typedef void (*InitializationEx)();
InitializationEx  m_pfInitialization;
// 2. 라이브러리 메모리 로드 처리
HINSTANCE  m_hInst;
// SmartPlayer의 인터페이스를 사용하기 위해 SmartPlayCore.dll 파일을 메모리에 로드
// IEC-Series 및 사용하는 기능별로 외부 DLL 파일이 다르며, 상단 표를 참고하시기 바랍니다.
m_hInst = LoadLibrary(L"SmartPlayCore.dll");
// 외부 DLL 파일이 다른 경로(Flash Disk\\extdlls)에 저장된 경우
// m_hInst = LoadLibrary(L"Flash Disk\\extdlls\\SmartPlayCore.dll");
if(m_hInst == NULL)
{
  // 메모리 로드 실패
  return FALSE;
}
// 3. 인터페이스 함수 얻기
m_pfInitialization = (InitializationEx)GetProcAddress(m_hInst, _T("Initialization"));
// 4. 함수 사용
m_pfInitialization();
```

참고 Wrapping Class 제공안내

사용할 인터페이스를 각각 선언 및 얻어오는 복잡한 과정을 거칠 필요 없이 한 번의 선언과 인스턴싱만으로 편리하게 SmartPlay의 인터페이스를 사용할 수 있도록 Wrapping Class를 제공하고 있습니다.
자세한 내용은 "CSmartPlayWrap Class 설명" 내용을 참고하시기 바랍니다.

3) C++ 인터페이스 설명

함수	InitializationEx

SmartPlayer가 사용되기 전에 호출하여 초기화합니다.

중요 최초 1회 실행되어야 합니다.

◉ typedef void (*InitializationEx)()

C++ 사용법

```
typedef void (*InitializationEx)();
HINSTANCE                    m_hInst;
InitializationEx             m_pfInitialization;
m_hInst = LoadLibrary(L"SmartPlayCore.dll");
if(m_hInst == NULL)
{
     return FALSE;
}
m_pfInitialization = (InitializationEx)GetProcAddress(m_hInst, _T("Initialization"));
m_pfInitialization();  // SmartPlayer를 사용하기 위한 초기화 처리
```

함수	SetPreviewWindowHandleEx

Play View 창으로 사용할 Window 또는 컨트롤의 핸들을 설정합니다.

중요 최초 1회 실행되어야 합니다.

◉ typedef void (*SetPreviewWindowHandleEx)(HWND hPreviewWindow)

[인자]

- HWND hPreviewWindow : Play View 창 Handle

C++ 사용법

```
typedef void (*SetPreviewWindowHandleEx)(HWND hPreviewWindow);
HINSTANCE                       m_hInst;
SetPreviewWindowHandleEx        m_pfSetPreviewWindowHandle;
m_hInst = LoadLibrary(L"SmartPlayCore.dll");
if(m_hInst == NULL)
{
     return FALSE;
}
m_pfSetPreviewWindowHandle = (SetPreviewWindowHandleEx)GetProcAddress(m_hInst,
                              _T("SetPreviewWindowHandle"));
m_pfSetPreviewWindowHandle(m_View.m_hWnd);  // 동영상 재생 화면 Handle 설정
```

함수	OpenEx, OpenEx1

- OpenEx() : 재생할 동영상 파일의 경로를 설정합니다.
- OpenEx1() : 재생할 동영상 파일의 경로 및 크기를 설정합니다.

◉ typedef bool (*OpenEx)(LPTSTR strFilePathName)

◉ typedef bool (*OpenEx1)(LPTSTR strFilePathName, long lWidth, long lHeight)

[인자]

- LPTSTR strFilePathName : 재생할 동영상 파일의 경로와 파일명을 설정
- long lWidth : 화면에 보여질 동영상의 폭 설정
- long lHeight : 화면에 보여질 동영상의 높이 설정

주의 lWidth, lHeight 설정 시 주의사항

동영상 파일을 Open하면서 재생할 화면의 크기를 설정하며, 재생 중에는 화면의 크기를 변경할 수 없습니다. (단, 전체 화면 기능은 사용 가능합니다.) 만약 lWidth, lHeight값을 0으로 설정할 경우 화면의 크기는 SmartPlayer 컨트롤의 크기로 재생됩니다.

[리턴값]

- bool : 동영상 파일의 Open 여부
  - true : Open 성공
  - false : Open 실패

C++ 사용법

```
typedef bool (*OpenEx)(LPTSTR strFilePathName);
typedef bool (*OpenEx1)(LPTSTR strFilePathName, long lWidth, long lHeight);
HINSTANCE             m_hInst;
OpenEx                m_pfOpen;
OpenEx1               m_pfOpen1;
m_hInst = LoadLibrary(L"SmartPlayCore.dll");
if(m_hInst == NULL)
{
    return FALSE;
```

```
}
m_pfOpen = (OpenEx)GetProcAddress(m_hInst, _T("Open"));
m_pfOpen1 = (OpenEx1)GetProcAddress(m_hInst, _T("Open1"));
// 동영상 파일 Open하며, View 크기를 지정한 값으로 적용
m_pfOpen1(L"Flash Disk\\TestSmartPlayer.avi", 680, 320);
```

함수	PlayEx, PauseEx, StopEx

현재 Open된 동영상 파일을 제어합니다.

- PlayEx() : 동영상을 Play합니다.
- PauseEx() : 현재 Play 중인 동영상을 일시 정지합니다.
- StopEx() : 현재 Play 중인 동영상을 Stop합니다.

◉ typedef bool (*PlayEx)()

◉ typedef bool (*PauseEx)()

◉ typedef bool (*StopEx)()

[리턴값]

- bool : 동영상 파일의 Play, Pause, Stop 성공 여부
  - true : 성공
  - false : 실패

C++ 사용법

```
typedef bool (*PlayEx)();
typedef bool (*PauseEx)();
typedef bool (*StopEx)();
HINSTANCE           m_hInst;
PlayEx              m_pfPlay;
PauseEx             m_pfPause;
StopEx              m_pfStop;
m_hInst = LoadLibrary(L"SmartPlayCore.dll");
if(m_hInst == NULL)
{
     return FALSE;
}
m_pfPlay = (PlayEx)GetProcAddress(m_hInst, _T("Play"));
m_pfPause = (PauseEx)GetProcAddress(m_hInst, _T("Pause"));
m_pfStop = (StopEx)GetProcAddress(m_hInst, _T("Stop"));
m_pfPlay();  // Open된 파일을 Play
m_pfPause(); // 일시 정지
m_pfPlay();  // 정지한 곳부터 다시 Play
m_pfStop();  // 동영상 재생을 Stop
```

함수	IsCompleteEx

현재 재생 중인 동영상의 재생 완료 여부를 확인합니다.

◉ typedef bool (*IsCompleteEx)()

[리턴값]

- bool : 동영상의 재생 완료 여부
  - true : 재생 완료
  - false : 재생 중

C++ 사용법

```
typedef bool (*IsCompleteEx)();
HINSTANCE                     m_hInst;
IsCompleteEx                  m_pfIsComplete;
m_hInst = LoadLibrary(L"SmartPlayCore.dll");
if(m_hInst == NULL)
{
     return FALSE;
}
m_pfIsComplete = (IsCompleteEx)GetProcAddress(m_hInst, _T("IsComplete"));
CString strMsg;
// 동영상이 재생 중인지 확인
if(m_pfIsComplete() == false)
{
    strMsg.Format(L"재생 중 …");  // 동영상 재생 중
}
```

함수	GetCurrentPosEx, SetCurrentPosEx

- GetCurrentPosEx() : 현재 재생 중인 동영상의 재생 시점을 가져옵니다.
- SetCurrentPosEx() : 현재 동영상의 재생 시점을 설정하거나 읽습니다.

◉ typedef DWORD (*GetCurrentPosEx)()

◉ typedef bool (*SetCurrentPosEx)(DWORD dwPos)

[인자]

- DWORD dwPos : 설정할 현재 동영상의 재생 시점 (단위 : 초)

[리턴값]

- DWORD : 현재 동영상의 재생 시점 (단위 : 초)
- bool : 현재 동영상의 재생 시점 설정 또는 읽기 성공 여부
  - true : 성공
  - false : 실패

C++ 사용법

```
typedef unsigned long DWORD;
typedef DWORD (*GetCurrentPosEx)();
typedef bool (*SetCurrentPosEx)(DWORD dwPos);
HINSTANCE                     m_hInst;
GetCurrentPosEx               m_pfGetCurrentPos;
SetCurrentPosEx               m_pfSetCurrentPos;
m_hInst = LoadLibrary(L"SmartPlayCore.dll");
if(m_hInst == NULL)
{
     return FALSE;
}
m_pfGetCurrentPos = (GetCurrentPosEx)GetProcAddress(m_hInst, _T("GetCurrentPos"));
```

```
m_pfSetCurrentPos = (SetCurrentPosEx)GetProcAddress(m_hInst, _T("SetCurrentPos"));
// 현재 재생 시점을 얻음
DWORD dwPos = m_pfGetCurrentPos();
dwPos = 30;
// 재생 시점을 30초로 설정
m_pfSetCurrentPos(dwPos);
```

함수	GetMaxPosEx

현재 Open되어 있는 동영상 파일의 총 재생 시간을 가져옵니다.

◉ typedef DWORD (*GetMaxPosEx)()

[리턴값]

- DWORD : 동영상의 총 재생 시간 (단위 : 초)

C++ 사용법

```
typedef unsigned long DWORD;
typedef DWORD (*GetMaxPosEx)();
HINSTANCE                    m_hInst;
GetMaxPosEx                  m_pfGetMaxPos;
m_hInst = LoadLibrary(L"SmartPlayCore.dll");
if(m_hInst == NULL)
{
     return FALSE;
}
m_pfGetMaxPos = (GetMaxPosEx)GetProcAddress(m_hInst, _T("GetMaxPos"));
// 재생 중인 동영상의 총 재생 시간을 리턴
DWORD dwMax = m_pfGetMaxPos();
```

함수	GetVolumeEx, SetVolumeEx

- GetVolumeEx() : 현재 재생 중인 동영상의 소리 크기를 가져옵니다.
- SetVolumeEx() : 현재 동영상의 소리 크기를 설정하거나 읽습니다.

◉ typedef long (*GetVolumeEx)()

◉ typedef bool (*SetVolumeEx)(long lVol)

[인자]

- long lVol : 설정할 동영상의 소리 크기

[리턴값]

- long : 동영상의 소리 크기
- bool : 현재 재생 중인 동영상의 재생 시점 설정 성공 여부
  - true : 성공
  - false : 실패

C++ 사용법

```
typedef long (*GetVolumeEx)();
typedef bool (*SetVolumeEx)(long lVol);
HINSTANCE                    m_hInst;
```

```
GetVolumeEx                     m_pfGetVolume;
SetVolumeEx                     m_pfSetVolume;
m_hInst = LoadLibrary(L"SmartPlayCore.dll");
if(m_hInst == NULL)
{
     return FALSE;
}
m_pfGetVolume = (GetVolumeEx)GetProcAddress(m_hInst, _T("GetVolume"));
m_pfSetVolume = (SetVolumeEx)GetProcAddress(m_hInst, _T("SetVolume"));
// 현재 소리 크기를 얻음
long lVol = m_pfGetVolume();
lVol += 1000;
// 소리 크기가 1000 증가
m_pfSetVolume(lVol);
```

4) CSmartPlayWrap Class 설명

SmartPlayWrap Class는 사용할 인터페이스를 각각 선언 및 얻어오는 복잡한 과정을 거칠 필요 없이 한 번의 선언과 인스턴싱만으로 편리하게 SmartPlay의 기능을 C++(C++, Visual Studio 2005, 2008)에서 지원하도록 Wrapping 한 Class입니다. 본 Class를 사용하기 위해서는 "SmartPlayWrap.h" 파일과 "SmartPlayWrap.cpp" 파일을 대상 프로젝트에 복사한 후 #include "SmartPlayWrap.h"을 호출하여 사용하시기 바랍니다.

인터페이스 및 사용 방식은 위에서 설명한 C++ 인터페이스와는 약간의 차이가 있으므로 다음의 내용을 참고하시어 C++에서 적용하시기 바랍니다.

4-1) SmartPlayWrap Class 사용하기

C++에서 SmartPlayWrap Class를 사용하기 위해서는 아래와 STEP을 진행해야 합니다.

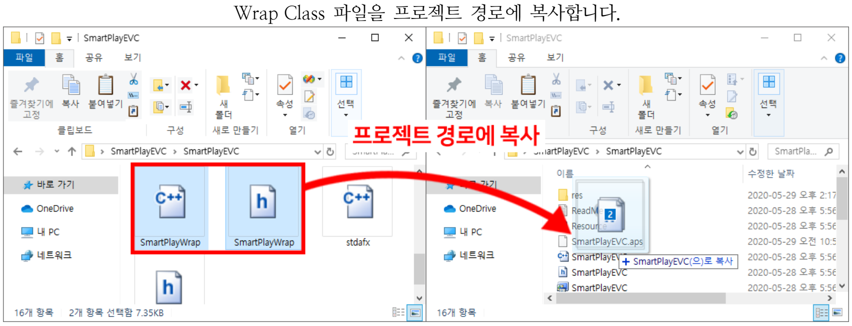
STEP-1	Wrap Class 파일 복사하기

HNS 홈페이지에서 사용하는 IEC-Series에 맞는 예제파일을 다운받은 후 압축을 풉니다.
"SmartPlayEVC" 폴더에서 아래 표를 확인하여 Wrap Class 파일을 프로젝트 경로에 복사합니다.

[표] IEC-Series별 Wrap Class 파일

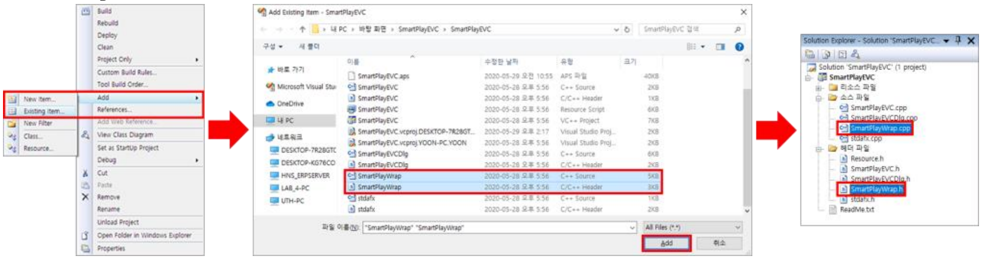
IEC-Series	IEC266-Series	IEC667-Series	IEC1000-Series
Wrap Class 파일	SmartPlayWrap.h, SmartPlayWrap.cpp		
다운로드 위치	홈페이지 → 자료실 → SmartX Framework 관련 → 6. SmartX Framework 예제파일 → Visual C++ 예제파일 → "SmartPlayEVC"		

Wrap Class 파일을 프로젝트 경로에 복사합니다.



STEP-2 Wrap Class 추가하기

복사한 Wrap Class 파일을 프로젝트에 추가합니다.



STEP-3 Wrap Class 사용하기

사용할 소스 파일에 #include〈SmartPlayWrap.h〉 추가한 후 선언 및 인스턴스하여 사용합니다.

```
#include "SmartPlayWrap.h"
// SmartPlayer를 사용하기 위해 변수 선언 및 인스턴싱
CSmartPlayWrap        m_SmartPlay;
// 동영상 재생을 위한 초기화 처리
m_SmartPlay.InitializationEx();
// 동영상 재생 화면 Handle 설정
m_SmartPlay.SetPreviewWindowHandle(hPreviewWindow);
// 동영상 소리 설정 : 1000
m_SmartPlay.SetVolume(m_SmartPlay.GetVolume(1000));
// 동영상 파일 Open 처리
m_SmartPlay.Open(strFilePathName, 680, 320);
// Open된 동영상 파일 Play
m_SmartPlay.Play();
// Play 중인 동영상 파일 일시 정지
m_SmartPlay.Pause();
// 정지한 곳부터 다시 Play
m_SmartPlay.Play();
// Play 중인 동영상 Stop
m_SmartPlay.Stop();
```

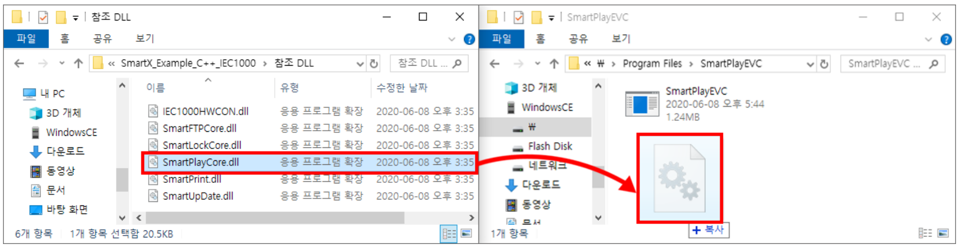
STEP-4 DLL 파일 복사하기

HNS 홈페이지에서 사용하는 IEC-Series에 맞는 예제파일을 다운받은 후 압축을 풉니다. "참조 DLL" 폴더에서 아래 표를 확인하여 사용하는 기능에 맞는 DLL 파일을 IEC-Series에 실행파일과 함께 복사합니다.

[표] IEC-Series별 DLL 파일과 다운로드 경로

IEC-Series	IEC266-Series	IEC667-Series	IEC1000-Series
DLL 파일	SmartPlayCore.dll		
다운로드 위치	홈페이지 → 자료실 → SmartX Framework 관련 → 5. SmartX C++ For DLL 파일		

DLL 파일을 실행 파일 위치에 복사합니다.



참고 자세한 DLL 파일의 복사 경로는 "DLL 파일 및 복사 경로" 내용을 참고하시기 바랍니다.

4-2) SmartPlayWrap Class 예제 사용하기

SmartPlayWrap Class를 활용한 예제를 제공하고 있으며, 예제파일은 다음 경로에서 다운로드하여 참고하시기 바랍니다.

[예제 파일 다운로드 위치]
홈페이지(www.hnsts.co.kr) → 자료실 → SmartX Framework 예제파일 다운로드 → “SmartPlayEVC”

13. SmartLock

SmartX Framework는 기본적으로 .Net Compact Framework에서만 지원되도록 설계되어 있지만, 일부 기능을 C++에서 편리하게 사용할 수 있도록 외부 DLL 파일을 제공하고 있습니다. 따라서 SmartLock를 사용하기 위해서는 반드시 외부 DLL 파일을 실행 파일과 같이 IEC-Series에 배포해야 하며, "1. 인터페이스 함수 선언 → 2. 라이브러리 메모리 로드 처리 → 3. 인터페이스 함수 얻기" 과정을 거쳐야 합니다. 이때 IEC-Series 및 기능별로 로드해야 할 외부 DLL 파일이 다르므로 다음 내용을 확인하시기 바랍니다.

참조	SmartLock의 자세한 설명은 "홈페이지 → 자료실 → SmartX Framework 관련 → 프로그래밍 가이드 → SmartLock"를 참조하시기 바랍니다.

1) 제품키 설정 및 형식 정의 시 주의사항

인증 키(제품키, Serial Number로도 표기함)는 프로그램의 최초 실행 시 프로그램을 등록하기 위해서 사용되는 키로, 일정 형식을 갖는 형태로 정의하여 사용할 수 있도록 관리하시기 바랍니다.
실행 프로그램 내부에 정의된 제품 키들과 등록 시 입력된 제품키의 동일 여부에 따라서 인증을 수행하며, 인증은 최초 실행 시 1회만 처리됩니다. 등록이 완료되면 다음 실행부터는 정상 실행되도록 처리합니다. 아래의 제품키의 형식 및 설정 시 주의사항을 참고하여 제품키의 형식 정의 및 설정을 하시기 바랍니다.

주의 제품키의 형식 및 설정 시 주의사항

1. 제품키는 고정 길이로만 설정

제품키를 여러 개 입력 시, 가변 길이가 아닌 고정 길이로만 설정해야 합니다. 즉, SetReferenceSNKeysEx() 함수에 제품키를 여러 개 설정하는 경우 각 제품키의 크기를 모두 같게 설정하시기 바랍니다.

옳은 예	`m_pfSetReferenceSNKeys(L"0000-0000-0000-0001;0000-0000-0000-0002;");`
잘못된 예	`// 첫번째 제품키와 두번째 제품키의 길이가 다름` `m_pfSetReferenceSNKeys(L"000-000-000-001;0000-0000-0000-0002;");`

2. 제품키 마지막에 세미콜론 추가

제품키 입력 시 세미콜론에 주의 하셔야 합니다. 즉, SetReferenceSNKeysEx() 함수에 제품키 설정 시, 제품키의 개수와 상관없이 각 제품키의 마지막에 ';'(세미콜론)을 추가하셔야 합니다.

옳은 예	`m_pfSetReferenceSNKeys(L"0000-0000-0000-0001;");` `m_pfSetReferenceSNKeys(L"0000-0000-0000-0001;0000-0000-0000-0002;");`

잘못된 예	`// 세미콜론이 없음`
	`m_pfSetReferenceSNKeys(L"0000-0000-0000-0001");`
	`m_pfSetReferenceSNKeys(L"0000-0000-0000-0001;0000-0000-0000-0002");`
	`// 세미콜론이 앞과 뒤에 사용됨`
	`m_pfSetReferenceSNKeys(L";0000-0000-0000-0001");`
	`m_pfSetReferenceSNKeys(L";0000-0000-0000-0001;0000-0000-0000-0002");`

3. 제품키 구성 시 주의사항

제품키 입력 시 문자 입력에 주의해야 합니다.

3-1. 사용 가능한 문자는 0~9(숫자), A~Z(대문자), -(기호) 이며, 각 제품키의 구분 문자는 ';'(세미콜론) 입니다. 또한, 공백 문자를 주의하시기 바랍니다.

3-2. 하나의 키가 20자를 넘을 수 없습니다. "-" 포함 5~19자리로 설정해야 하며, 200개 미만의 데이터를 사용하시기 바랍니다. 또한 각 제품키의 마지막에는 반드시 ';'(세미콜론)을 추가해야 합니다.

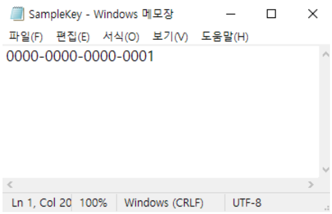
구분	예시
옳은 예	m_pfSetReferenceSNKeys(L"0000-0000-0000-0001;"); m_pfSetReferenceSNKeys(L"A000-B000-C000-D001;0000-0000-0000-0002;");

잘못된 예	// 제품키가 20자를 넘음
	m_pfSetReferenceSNKeys(L"0000-0000-0000-0001233");
	// 제품키에 한글이 포함됨
	m_pfSetReferenceSNKeys(L"가나다라-0000-0000-0001");
	// 제품키에 특수문자가 포함됨
	m_pfSetReferenceSNKeys(L"!@#$-0000-0000-0001;*&A4-0000-0000-0002");

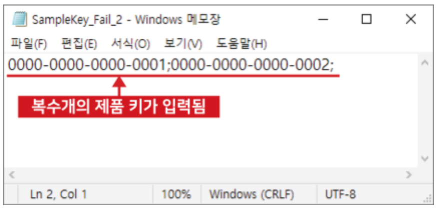
4. 자동 등록 파일 사용 시 주의사항
AutoRegistrationApplyEx() 함수를 사용해 제품키 자동 등록 기능을 사용할 때 자동 등록 파일 안에는 반드시 올바른 형식의 제품키 하나만 존재해야 합니다. 그렇지 않은 경우 제품키 등록이 되지 않습니다. 또한, ';'(세미콜론)을 제외한 제품키만이 저장되어 있어야 하며, 엔터키, 띄어쓰기, 잘못된 키 등이 입력되면 등록이 되지 않습니다.

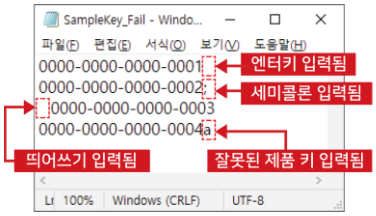
제품키	m_pfSetReferenceSNKeys(L"0000-0000-0000-0001;");

옳은 예



잘못된 예





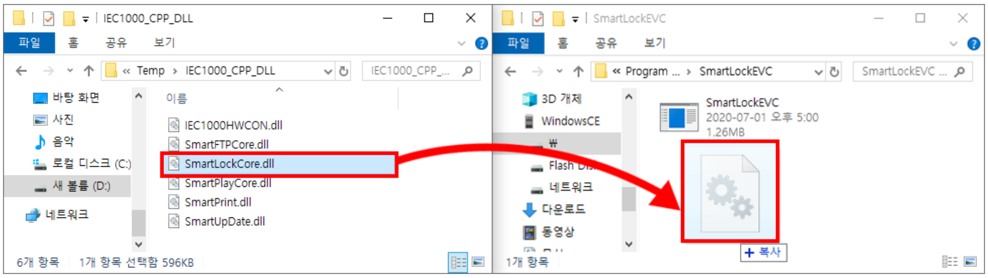
2) DLL 파일 및 복사 경로

HNS 홈페이지에서 사용하는 IEC-Series에 맞는 C++ For DLL 파일을 다운로드 받은 후 압축을 풉니다. 아래 표를 확인하여 DLL 파일을 IEC-Series에 실행 파일과 함께 복사합니다.

[표] IEC-Series별 DLL 파일과 다운로드 경로

IEC-Series	IEC266-Series	IEC667-Series	IEC1000-Series
DLL 파일	SmartLockCore.dll		
다운로드 위치	홈페이지 → 자료실 → SmartX Framework 관련 → 5. SmartX C++ For DLL 파일		

DLL 파일을 IEC-Series에 복사합니다. DLL 파일을 사용하는 실행 파일이 저장된 폴더에 저장하는 것을 권장하며, 다른 경로에 저장하는 경우 "인터페이스 선언 및 얻기"의 참고 내용을 확인하시기 바랍니다.
※ 개발 PC에서 프로그램을 배포하는 경우 실행 파일은 "Program File" 폴더에 배포됩니다.



3) 인터페이스 선언 및 얻기

인터페이스 사용을 위해 사용할 인터페이스의 선언 후 외부 DLL 파일을 메모리에 로드하여 인터페이스를 얻는 과정을 거쳐야 합니다. 이때 메모리 로드 처리를 위해 LoadLibrary() 함수 호출 시 상단 표의 DLL 파일의 이름을 인자로 설정합니다. 아래 코드는 예시이며, 필요로 하는 인터페이스 함수를 사용하기 위해서는 다음 내용을 참고하시기 바랍니다.

참고 DLL 파일 저장 경로에 따른 LoadLibrary() 함수 인자값

DLL 파일이 실행 파일과 같은 경로에 저장된 경우, LoadLibrary() 함수 인자값을 DLL 파일 이름만으로 설정해 호출할 수 있습니다. 만약 DLL 파일이 실행 파일과 다른 경로에 저장된 경우, LoadLibray() 함수의 인자값에는 저장된 DLL 파일의 전체 경로를 인자값으로 설정해야 합니다.

[C++] 인터페이스 얻기 및 사용하기 예시 코드

```
// 1. 인터페이스 함수 선언
typedef void (*SetTitleEx)(LPTSTR strTitle);
SetTitleEx  m_pfSetTitle;
// 2. 라이브러리 메모리 로드 처리
HINSTANCE   m_hInst;
// SmartLock의 인터페이스를 사용하기 위해 SmartLockCore.dll 파일을 메모리에 로드
// IEC-Series 및 사용하는 기능별로 외부 DLL 파일이 다르며, 상단 표를 참고하시기 바랍니다.
m_hInst = LoadLibrary(L"SmartLockCore.dll");
// 외부 DLL 파일이 다른 경로(Flash Disk\\extdlls)에 저장된 경우
// m_hInst = LoadLibrary(L"Flash Disk\\extdlls\\SmartLockCore.dll");
if(m_hInst == NULL)
{
  // 메모리 로드 실패
  return FALSE;
}
// 3. 인터페이스 함수 얻기
m_pfSetTitle = (SetTitleEx)GetProcAddress(m_hInst, _T("SetTitle"));
// 4. 함수 사용
m_pfSetTitle(L"SmartLock C++ Ver.");
```

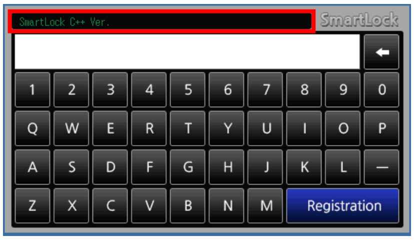
참고 Wrapping Class 제공안내

사용할 인터페이스를 각각 선언 및 얻어오는 복잡한 과정을 거칠 필요 없이 한 번의 선언과 인스턴싱만으로 편리하게 SmartLock의 인터페이스를 사용할 수 있도록 Wrapping Class를 제공하고 있습니다.
자세한 내용은 "CSmartLockWrap Class 설명" 내용을 참고하시기 바랍니다.

4) C++ 인터페이스 설명

함수	SetTitleEx

제품키를 수동으로 입력하기 위한 입력 창 상단의 메시지 내용을 설정합니다.



참고 "RegistrationFormShowEx() 함수" 내용을 참고하시기 바랍니다.

◉ typedef void (*SetTitleEx)(LPTSTR strTitle)

[인자]

- LPTSTR strTitle : 입력창 상단 메시지

C++ 사용법

```
typedef void (*SetTitleEx)(LPTSTR strTitle);
HINSTANCE           m_hInst;
SetTitleEx          m_pfSetTitle;
m_hInst = LoadLibrary(L"SmartLockCore.dll");
if(m_hInst == NULL)
{
  return FALSE;
}
m_pfSetTitle = (SetTitleEx)GetProcAddress(m_hInst, _T("SetTitle"));
m_pfSetTitle(L"SmartLock C++ Ver.");
```

함수	SetReferenceSNKeysEx

프로그램의 등록을 위해 사용할 제품키들을 설정합니다.

◉ typedef void (*SetReferenceSNKeysEx)(LPTSTR strRefSNKeys)

중요 SetReferenceSNKeysEx() 함수 호출 시 중요사항

1. 사용 가능한 문자는 0~9(숫자), A~Z(대문자), -(기호) 이며, 각 제품키의 구분 문자는 ;(세미콜론) 입니다. 또한, 공백 문자를 주의하시기 바랍니다.
2. 하나의 키가 20자를 넘을수 없습니다. "-" 포함 5~19자리로 설정해야 하며, 200개 미만의 데이터를 사용하시기 바랍니다. 또한 각 제품키의 마지막에는 반드시 세미콜론(';')을 추가해야 합니다.

참고 "제품키 설정 및 형식 정의 시 주의사항" 내용을 참고하시기 바랍니다.

[인자]

- LPTSTR strRefSNKeys : 제품 키

C++ 사용법

```
typedef void (*SetReferenceSNKeysEx)(LPTSTR strRefSNKeys);
```

```
HINSTANCE                      m_hInst;
SetReferenceSNKeysEx           m_pfSetReferenceSNKeys;
m_hInst = LoadLibrary(L"SmartLockCore.dll");
if(m_hInst == NULL)
{
  return FALSE;
}
m_pfSetReferenceSNKeys = (SetReferenceSNKeysEx)GetProcAddress(m_hInst,_T("SetReferenceSNKeys"));
// 제품키 한 개 등록, 마지막에 구분자 ;를 꼭 넣어주시기 바랍니다.
m_pfSetReferenceSNKeys(L"0000-0000-0000-0001;");
// 제품키 한 개 이상 등록, 마지막에 구분자 ;를 꼭 넣어주시기 바랍니다.
m_pfSetReferenceSNKeys(L"0000-0000-0000-0001;0000-0000-0000-0002;");
```

함수 SetAutoRegistrationFilePathNameEx

제품키의 자동 등록 처리를 위해 제품키가 입력된 파일의 경로와 파일명을 설정합니다.

참고	"AutoRegistrationApplyEx() 함수" 내용을 참고하시기 바랍니다.

◉ typedef void (*SetAutoRegistrationFilePathNameEx)(LPTSTR strFilePathName)

[인자]

- LPTSTR strFilePathName : 제품키가 입력되어있는 파일 경로

C++ 사용법

```
typedef void (*SetAutoRegistrationFilePathNameEx)(LPTSTR strFilePathName);
HINSTANCE                            m_hInst;
SetAutoRegistrationFilePathNameEx    m_pfSetAutoRegistrationFilePathName;
m_hInst = LoadLibrary(L"SmartLockCore.dll");
if(m_hInst == NULL)
{
  return FALSE;
}
m_pfSetAutoRegistrationFilePathName = (SetAutoRegistrationFilePathNameEx)GetProcAddress(m_hInst,
                                        _T("SetAutoRegistrationFilePathName"));
//제품키가 입력되어있는 파일 경로
m_pfSetAutoRegistrationFilePathName(L"SD Card\\Run\\lock.lic");
```

함수 RegistrationCheckEx

프로그램의 정상 등록 여부를 확인합니다.

◉ typedef bool (*RegistrationCheckEx)()

[리턴값]

- bool : 등록 여부
  - true : 등록됨
  - false : 등록되지 않음

C++ 사용법

```
typedef void (*RegistrationCheckEx);
```

```
HINSTANCE                    m_hInst;
RegistrationCheckEx          m_pfRegistrationCheck;
m_hInst = LoadLibrary(L"SmartLockCore.dll");
if(m_hInst == NULL)
{
  return FALSE;
}
m_pfRegistrationCheck = (RegistrationCheckEx)GetProcAddress(m_hInst, _T("RegistrationCheck"));
if (m_pfRegistrationCheck() == true)
{
  // 정상 등록된 프로그램
}
```

함수	AutoRegistrationApplyEx

SetAutoRegistrationFilePathNameEx() 함수에서 설정된 경로의 파일을 읽어 제품키를 등록합니다

참고 "SetAutoRegistrationFilePathNameEx() 함수" 내용을 참고하시기 바랍니다.

◉ typedef int (*AutoRegistrationApplyEx)()

[리턴값]

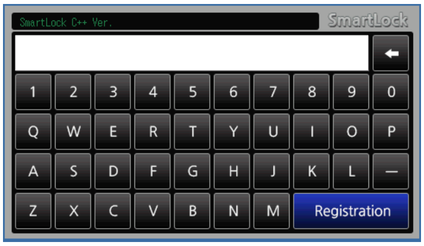
- int : 제품 키 등록 결과
  - 0 : 등록 완료(성공)
  - 1 : 제품키 파일이 지정된 경로에 없는 경우(실패)
  - 2 : 잘못된 제품키 입력(실패)
  - 3 : 제품키 등록 오류(실패)

C++ 사용법

```
typedef void (*AutoRegistrationApplyEx)();
HINSTANCE                    m_hInst;
AutoRegistrationApplyEx      m_pfAutoRegistrationApply;
m_hInst = LoadLibrary(L"SmartLockCore.dll");
if(m_hInst == NULL)
{
  return FALSE;
}
m_pfAutoRegistrationApply = (AutoRegistrationApplyEx)GetProcAddress(m_hInst,
                           _T("AutoRegistrationApply"));
if (m_pfAutoRegistrationApply() == 0)
{
  // 자동 등록 성공
}
else
{
  // 자동 등록 실패
}
```

함수	RegistrationFormShowEx

수동으로 프로그램을 등록하기 위해 제품키 입력 창을 출력합니다.



◉ typedef int (*RegistrationFormShowEx)()

[리턴값]

• int : 제품 키 등록 결과
  - 0 : 등록 완료(성공)
  - 1 : 등록 실패

C++ 사용법

```
typedef void (*RegistrationFormShowEx)();
HINSTANCE                       m_hInst;
RegistrationFormShowEx          m_pfRegistrationFormShow;
m_hInst = LoadLibrary(L"SmartLockCore.dll");
if(m_hInst == NULL)
{
  return FALSE;
}
m_pfRegistrationFormShow = (RegistrationFormShowEx)GetProcAddress(m_hInst,
                           _T("RegistrationFormShow"));
if (m_pfSetConnectInfo() == 0)
{
  // 수동 등록 성공
}
else
{
  // 수동 등록 실패
}
```

5) CSmartLockWrap Class 설명

SmartLockWrap Class는 사용할 인터페이스를 각각 선언 및 얻어오는 복잡한 과정을 거칠 필요 없이 한 번의 선언과 인스턴싱만으로 편리하게 SmartLock의 기능을 C++(C++, Visual Studio 2005, 2008)에서 지원하도록 Wrapping한 Class입니다. 본 Class를 사용하기 위해서는 “SmartLockWrap.h” 파일과 “SmartLockWrap.cpp” 파일을 대상 프로젝트에 복사한 후 #include "SmartLockWrap.h"을 호출하여 사용하시기 바랍니다.

인터페이스 및 사용 방식은 위에서 설명한 C++ 인터페이스와는 약간의 차이가 있으므로 다음의 내용을 참고하시어 C++에서 적용하시기 바랍니다.

5-1) SmartLockWrap Class 사용하기

C++에서 SmartLockWrap Class를 사용하기 위해서는 아래와 STEP을 진행해야 합니다.

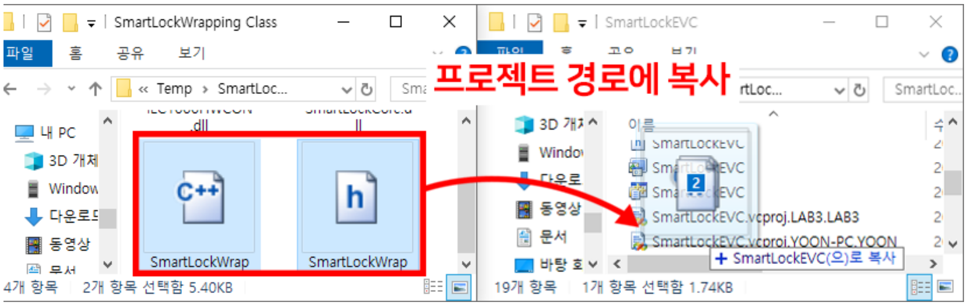
STEP-1 Wrap Class 파일 복사하기

HNS 홈페이지에서 사용하는 IEC-Series에 맞는 예제파일을 다운받은 후 압축을 풉니다.
“SmartLockEVC” 폴더에서 아래 표를 확인하여 Wrap Class 파일을 프로젝트 경로에 복사합니다.

[표] IEC-Series별 Wrap Class 파일

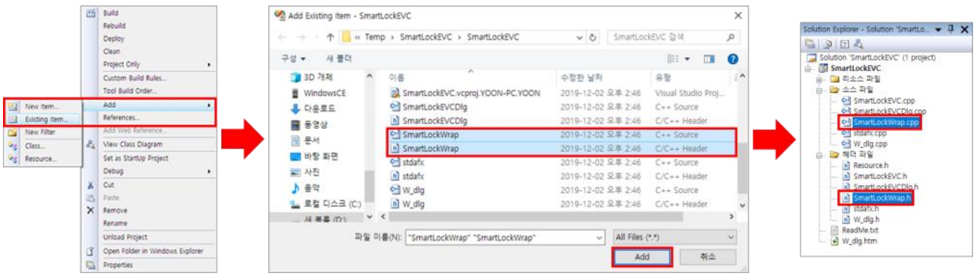
IEC-Series	IEC266-Series	IEC667-Series	IEC1000-Series
Wrap Class 파일	SmartLockWrap.h, SmartLockWrap.cpp		
다운로드 위치	홈페이지 → 자료실 → SmartX Framework 관련 → 6. SmartX Framework 예제파일 → Visual C++ 예제파일 → "SmartLockEVC"		

Wrap Class 파일을 프로젝트 경로에 복사합니다.



STEP-2 Wrap Class 추가하기

복사한 Wrap Class 파일을 프로젝트에 추가합니다.



STEP-3 Wrap Class 사용하기

사용할 소스 파일에 #include〈SmartLockWrap.h〉 추가한 후 선언 및 인스턴스하여 사용합니다.

```
#include "SmartLockWrap.h"
// SmartLock을 사용하기 위해 변수 선언 및 인스턴싱
CSmartLockWrap m_SmartLockCore;
// 프로그램의 등록을 위해 사용할 제품키들을 설정
m_SmartLockCore.SetReferenceSNKeys(L"0000-0000-0000-0001;");
// 제품키 자동 등록을 위한 경로설정
m_SmartLockCore.SetAutoRegistrationFilePathName(L"Flash Disk\\Run\\lock.lic");
// 현재 실행되는 프로그램이 정상 등록된 프로그램인지 확인한다.
if(m_SmartLockCore.RegistrationCheck() == false)
{
  // 등록이 안된 경우 프로그램을 자동 등록 처리한다.
```

```
  if(m_SmartLockCore.RegistrationCheck() != 0)
  {
    // 수동 설정창 상단 문구 설정
    m_SmartLockCore.SetTitle(L"SmartLock C++ Ver.");
    // 제품키 수동 등록창 출력
    if(m_SmartLockCore.RegistrationFormShow() != 0)
    {
      // 수동 등록에 실패함
    }
  }
}
```

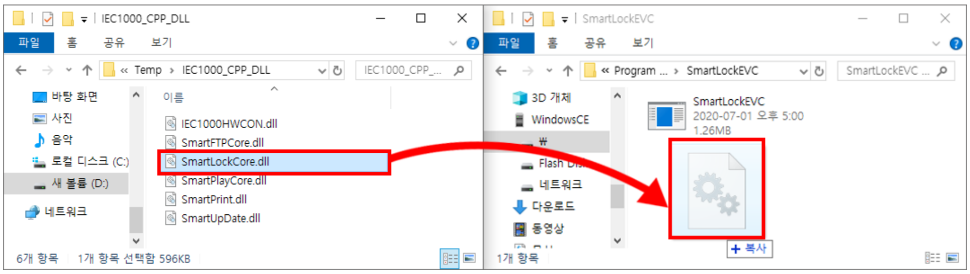
STEP-4	DLL 파일 복사하기

HNS 홈페이지에서 사용하는 IEC-Series에 맞는 예제파일을 다운받은 후 압축을 풉니다. "참조 DLL" 폴더에서 아래 표를 확인하여 사용하는 기능에 맞는 DLL 파일을 IEC-Series에 실행파일과 함께 복사합니다.

[표] IEC-Series별 DLL 파일과 다운로드 경로

IEC-Series	IEC266-Series	IEC667-Series	IEC1000-Series
DLL 파일	SmartLockCore.dll		
다운로드 위치	홈페이지 → 자료실 → SmartX Framework 관련 → 5. SmartX C++ For DLL 파일		

DLL 파일을 실행 파일 위치에 복사합니다.

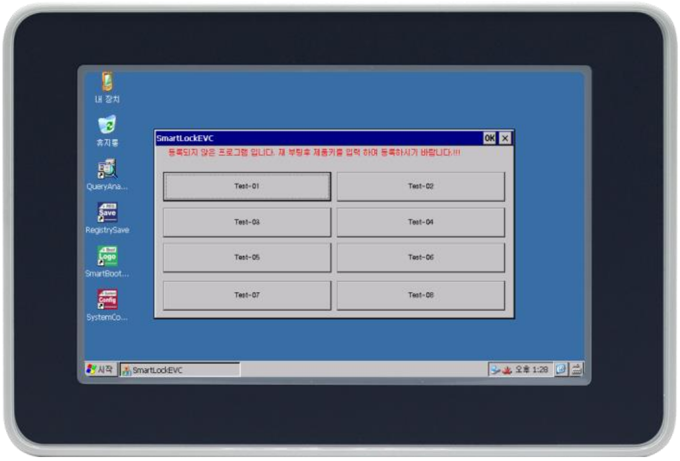


참고	자세한 DLL 파일의 복사 경로는 "DLL 파일 및 복사 경로" 내용을 참고하시기 바랍니다.

5-2) SmartLockWrap Class 예제 사용하기

SmartLockWrap Class를 활용한 예제를 제공하고 있으며, 예제파일은 다음 경로에서 다운로드하여 참고하시기 바랍니다.

[예제 파일 다운로드 위치]
홈페이지(www.hnsts.co.kr) → 자료실 → SmartX Framework 예제파일 다운로드 → "SmartLockEVC"



SmartX Framework
프로그래밍 가이드

제품 및 자세한 문의 사항은 아래의 연락처로 연락 및 메일 문의 주시기 바랍니다. 감사합니다.

㈜에이치앤에스
서울특별시 금천구 가산디지털1로 181, 1505호(가산 W센터)
대표전화 : 02-6402-8001 / 팩스 : 02-6442-9775

부서안내	연락처	직통 전화	이메일
제품 구매 및 견적문의	02-6402-8001(내선 1번)	070-7094-5770	sales@hnsts.co.kr
하드웨어 기술문의	02-6402-8001(내선 2번)	070-7094-5001	hns@hnsts.co.kr
소프트웨어 기술문의	02-6402-8001(내선 3번)	070-7094-5002	app@smartx.co.kr
제품서비스 기술문의	02-6402-8001(내선 4번)	070-7094-5003	tech@smartx.co.kr

홈페이지 : www.hnsts.co.kr / 쇼핑몰(제품구매) : www.hnsstore.co.kr