# Integration von IDS sherpa in eine IDS Software Suite MFC-Anwendung

Um eine bestehende MFC-Kameraanwendung der IDS Software Suite für die Verwendung von IDS peak umzustellen sind ein paar wenige Schritte notwendig:



## 1) Ordnerpfad der IDS peak SDK Header angeben

Mit der Installation von IDS peak ist der Pfad zu den **Entwicklungsdateien** (Bibliotheken, Header, usw.) als **Umgebungsvariable** in Windows eingerichtet. Fügen Sie damit die Pfade zur API (Application Programming Interface) und zur IPL (Image Processing Library) als zusätzliche Include-Verzeichnisse in allen Konfigurationen der IDE (32/64-Bit, Debug, Release) hinzu.

- $(IDS_PEAK_SDK_PATH)\api\include
- $(IDS_PEAK_SDK_PATH)\ipl\include

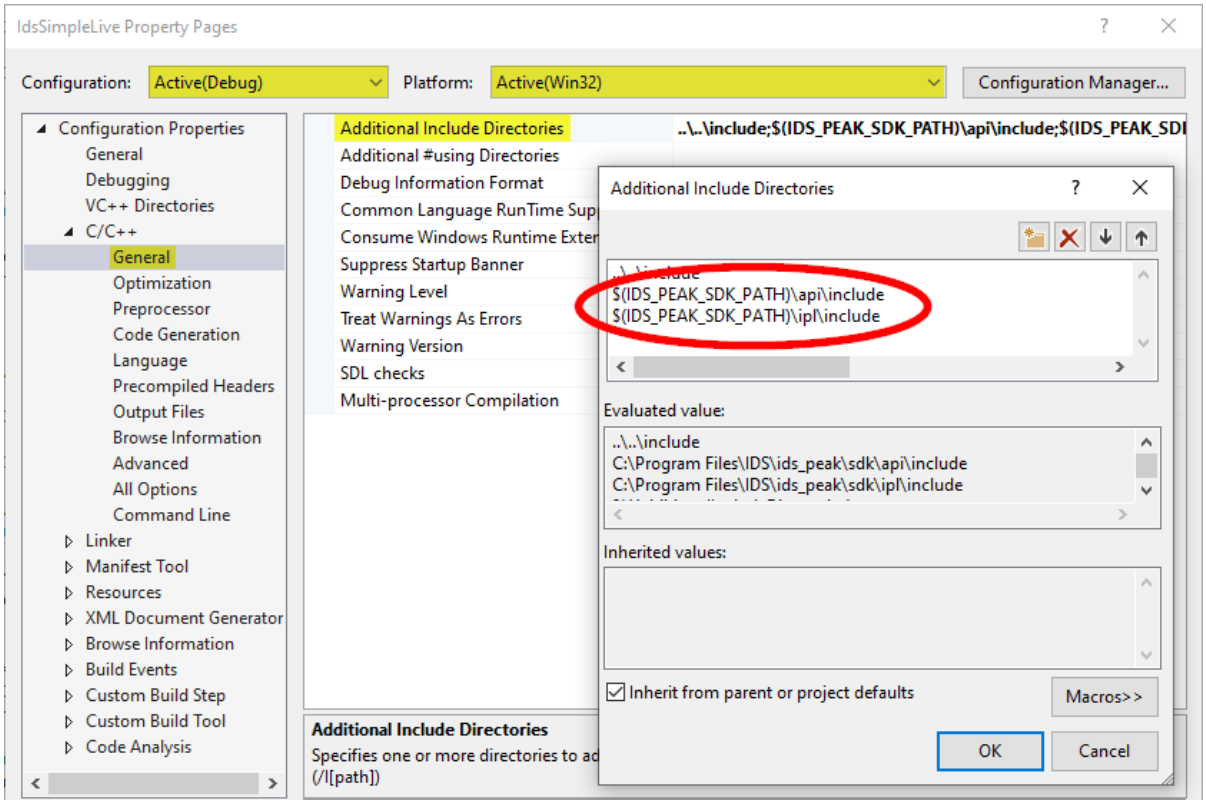


*Abbildung 1 IDS peak Include-Verzeichnisse*

## 2) Ordnerpfad der IDS peak SDK Bibliotheken angeben

Für den Linker müssen Sie zuerst die Pfade zu den Bilbliotheken der API und IPL vorgeben. Auch dies erledigen Sie in allen Konfigurationen mit Hilfe der IDS_PEAK_SDK_PATH Variablen.

| 32-bit | 64-bit |
|---|---|
| • $(IDS_PEAK_SDK_PATH)\api\lib\x86_32<br>• $(IDS_PEAK_SDK_PATH)\ipl\lib\x86_32 | • $(IDS_PEAK_SDK_PATH)\api\lib\x86_64<br>• $(IDS_PEAK_SDK_PATH)\ipl\lib\x86_64 |

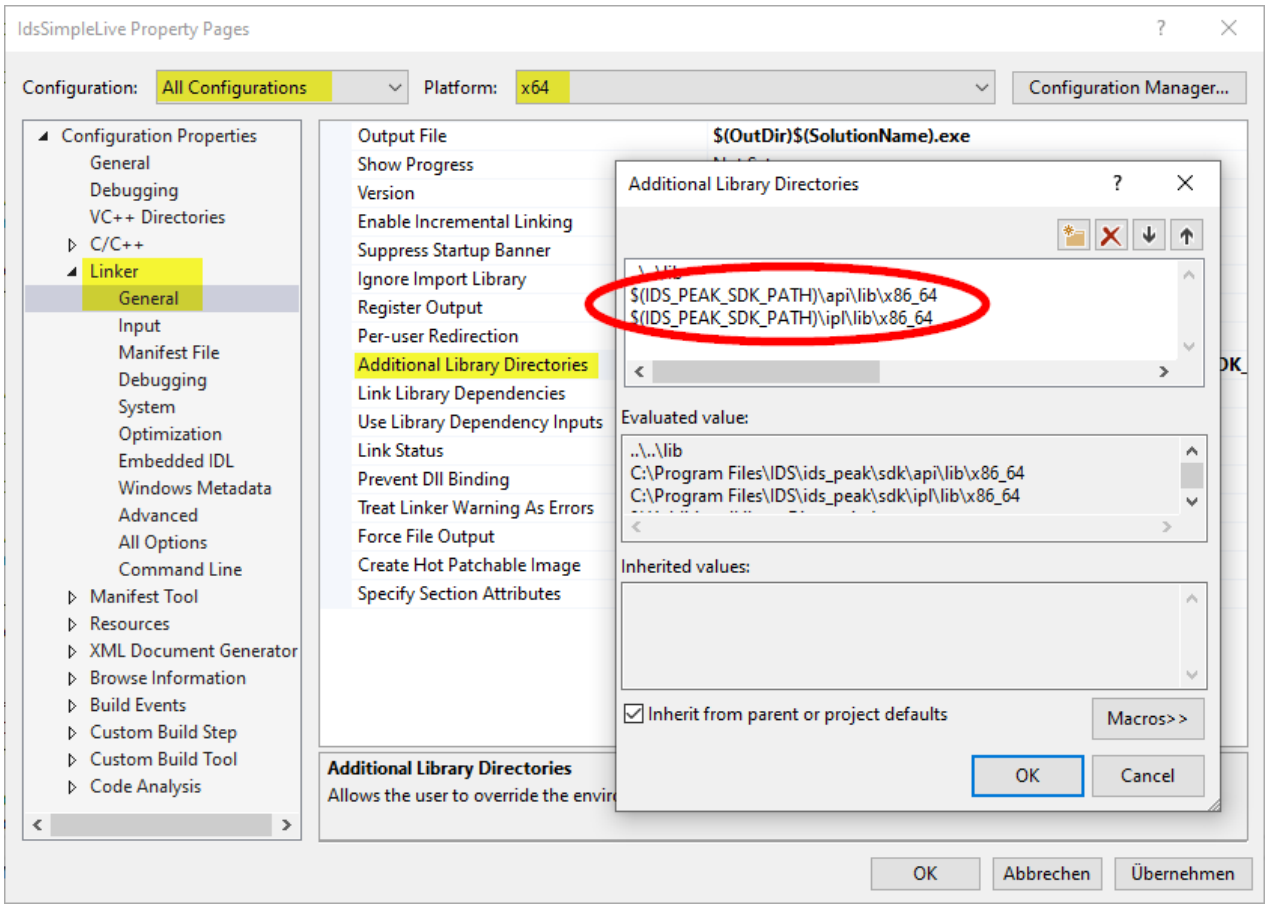


*Abbildung 2 IDS peak Bibliotheks-Verzeichnisse angeben*

Danach integrieren Sie die beiden Bibliotheken in allen Konfigurationen.

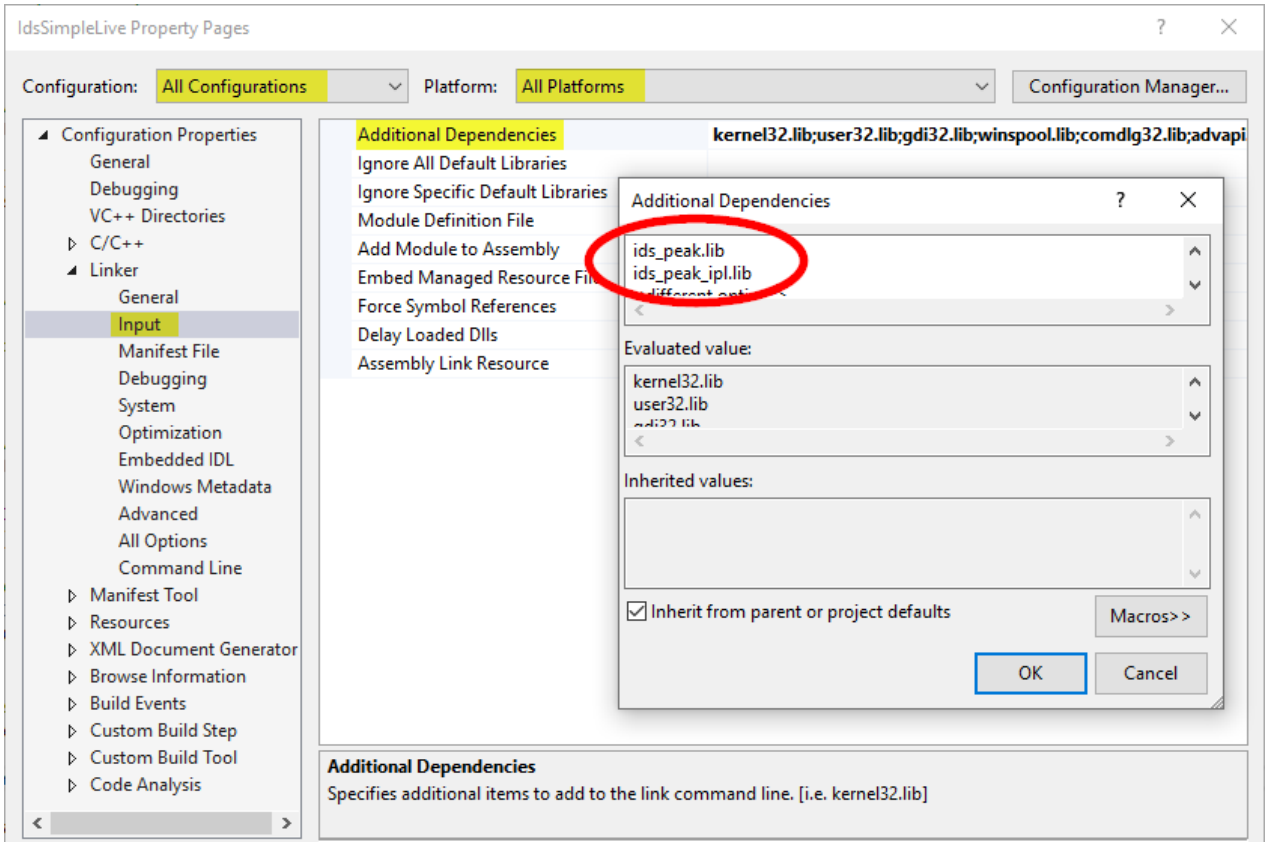


*Abbildung 3 IDS peak Bibliotheken integrieren*

## 3) Compiler Flags einrichten

In den Projekteigenschaften müssen noch ein paar wenige zusätzliche Compiler Flags eingerichtet werden, um C++14 zu kompilieren.

| Projekteigenschaft | Flags | Wert |
|---|---|---|
| C/C++ → Advanced | Compile As | Compile as C++ Code (/TP) |
| C/C++ → Command Line | Additional Options | /bigobj /std:c++14 |

## 4) Programmbibliotheken für die Ausführung kopieren (post-build step)

Für die Ausführung der fertig erstellten Anwendung (*.exe) außerhalb der Entwicklungsumgebung müssen Sie die dynamisch gelinkten Bibliotheken von IDS peak und GenICam ins Zielverzeichnis kopieren. Das erledigen wir in einem nachträglichen Kopiervorgang (Post-Build Event) in der jeweiligen Konfiguration. Die Dateien finden Sie ebenfalls alle im IDS peak Installationspfad.

**Benötige Bibliotheken:**

- ids_peak.dll
- ids_peak_ipl.dll
- FirmwareUpdate_MD_VC140_v3_1_IDS.dll
- GCBase_MD_VC140_v3_1_IDS.dll

- GenApi_MD_VC140_v3_1_IDS.dll
- Log_MD_VC140_v3_1_IDS.dll
- MathParser_MD_VC140_v3_1_IDS.dll
- NodeMapData_MD_VC140_v3_1_IDS.dll
- XmlParser_MD_VC140_v3_1_IDS.dll

**Kopierbefehl im Post-Build Event (64-bit Konfiguration)**

```
copy /Y "$(IDS_PEAK_SDK_PATH)\api\lib\x86_64\*.dll" "$(OutDir)\*.dll"
copy /Y "$(IDS_PEAK_SDK_PATH)\ipl\lib\x86_64\*.dll" "$(OutDir)\*.dll"
```

**Kopierbefehl im Post-Build Event (32-bit Konfiguration)**

```
copy /Y "$(IDS_PEAK_SDK_PATH)\api\lib\x86_32\*.dll" "$(OutDir)\*.dll"
copy /Y "$(IDS_PEAK_SDK_PATH)\ipl\lib\x86_32\*.dll" "$(OutDir)\*.dll"
```

## 5) IDS sherpa Dateien integrieren

Um IDS sherpa Sourcecode in Ihrem Projekt zu verwenden, kopieren Sie die drei Dateien der Klasse "IdsSherpa" in Ihr Projekt:

- IdsSherpa.h – Funktionsdeklarationen der IDS sherpa Klasse
- IdsSherpa_ueye.cpp – Funktionsdefinitionen "IDS sherpa → IDS Software Suite"
- IdsSherpa_peak.cpp – Funktionsdefinitionen "IDS sherpa → IDS peak"

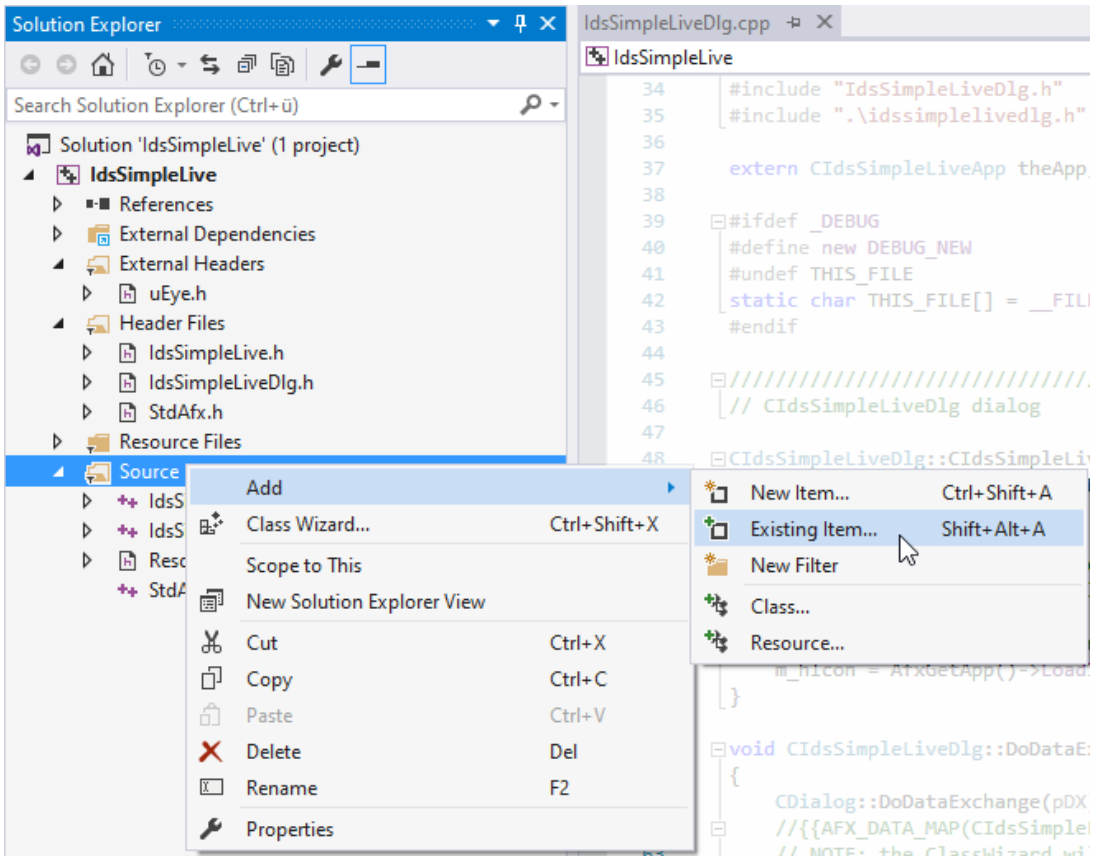


*Abbildung 4 IDS sherpa Dateien zum Projekt hinzufügen*

Anschließend fügen Sie "include" Anweisungen in die Anwendungsklasse ein:

- "IdsSherpa.h" in "IdsSimpleLiveDlg.h" integrieren:
  *#include "IdsSherpa.h"*
- "IdsSherpa.h" in "IdsSimpleLiveDlg.cpp" integrieren:
  *#include "IdsSherpa.h"*

# 6) Programmcode modifizieren

Um die IDS sherpa Adapter-Klasse in Ihrem Projekt zu verwenden, müssen alle C-Funktionen der uEye API (IDS Software Suite) durch Funktionen ersetzt werden, die im sherpa-Header deklariert sind. Im Idealfall gibt es für alle verwendeten uEye API Funktionen eine entsprechende sherpa-Funktion. Alle uEye API Funktionen beginnen mit **is**_ . Es macht Sinn, danach im Quellcode zu suchen, um alle Funktionen zu identifizieren. Mit Hilfer der IDS sherpa Funktionsliste (siehe IDS Sherpa - Anhang 2) können Sie nach äquivalenten Funktionen suchen.

## Beispiel: Funktionsaufruf ersetzen

Sie suchen die IDS sherpa Funktion für die uEye API Funktion "is_FreezeVideo". Die äquivalente Funktion "FreezeVideo" der IDS sherpa Klasse hat bis auf das "is_" den selben Namen. Die Funktions-Signatur kann sich geringfügig unterscheiden. Da wir hier C mit C++ Funktionen vergleichen, fehlt beispielsweise immer das Kamera Handle, dass bereits über den C++-Funktionsaufruf über die Basisklasse referenziert wird.

| Nr. | IDS Software Suite Funktion | IDS sherpa Funktion |
|---|---|---|
| 22 | INT is_FreezeVideo (HIDS hCam, INT Wait) | int FreezeVideo(int wait) |

```
#if !defined USE_SHERPA
    int ret = is_FreezeVideo(m_hCam, IS_WAIT); // IDS Software Suite function call
#else
    int ret = m_hCam->FreezeVideo(IS_WAIT); // IDS sherpa function call
#endif
```

### Funktions-Äquivalent nicht vorhanden?

Wenn Sie eine sherpa Funktions-Äquivalent für eine uEye API Funktion (siehe IDS sherpa Anhang 2) nicht finden, gibt es folgende Möglichkeiten:

- Prüfen Sie, ob die Funktionalität überhaupt in IDS peak verfügbar ist (→ IDS peak Dokumentation)
- Wenn die Funktionalität weiter benötigt wird, kann sie anders abgebildet werden?
- Kontaktieren Sie die IDS Systemberatung

## "IDS peak" oder "IDS Software Suite"

Nachdem Sie im Sourcecode die "is_"-Kamerafunktionen mit IDS sherpa Funktionen ersetzt haben, kann Ihr Programm sowohl mit UI-Kameras (IDS Software Suite) als auch GV- und U3-Kameras (IDS peak) betrieben werden. Die IDS sherpa Adapterklasse überlädt die Funktionen mit zwei Implementierungen. Mit "*IdsSherpa_ueye.cpp*" werden weiterhin die bekannten uEye API Funktionen verwendet, mit der "IdsSherpa_peak.cpp" Implementierung können Sie ab jetzt IDS Vision-Kameras (Modellbezeichnung "U3" oder "GV") betreiben.

Sie müssen nur das Kamera-Handle mit der entsprechenden Implementierung anlegen. Im Beispielprogramm "IdsSherpaMulti" werden zwei Kameras aus beiden Kamerafamilien parallel, mit der selben (sherpa) Code-Basis betrieben.

```
// Create sherpa object for IDS Software Suite
m_hCam[1] = new IdsSherpa_ueye();

// Create sherpa object for IDS peak
m_hCam[2] = new IdsSherpa_peak();
```

Alles andere bleibt gleich! Die Vision-Kamera sollten sich durch "IdsSherpa_peak.cpp" genauso verhalten wie die UI-Kamera mit IDS Software Suite. Ausnahmen ergeben sich durch unterschiedliche Funktionsweisen der beiden Kameramodelle, die nicht über die Schnittstelle behoben werden können. Syntaktisch lassen sich alle Funktionen gleich aufrufen, egal ob es sich um die IdsSherpa_ueye oder IdsSherpa_peak Klasse handelt.

## Kontakt

IDS Imaging Development Systems GmbH
Dimbacher Str. 6-8
74182 Obersulm
Deutschland

T: +49 7134 96196-0
E: h.seitz@ids-imaging.de
W: www.ids-imaging.de