Omni Sensors mit LabVIEW verwenden

Markenzeichen

Microsoft, Windows sind eingetragene Markenzeichen der Firma Microsoft. LabVIEW ist eingetragenes Markenzeichen der Firma National Instruments

Haftungsausschuss

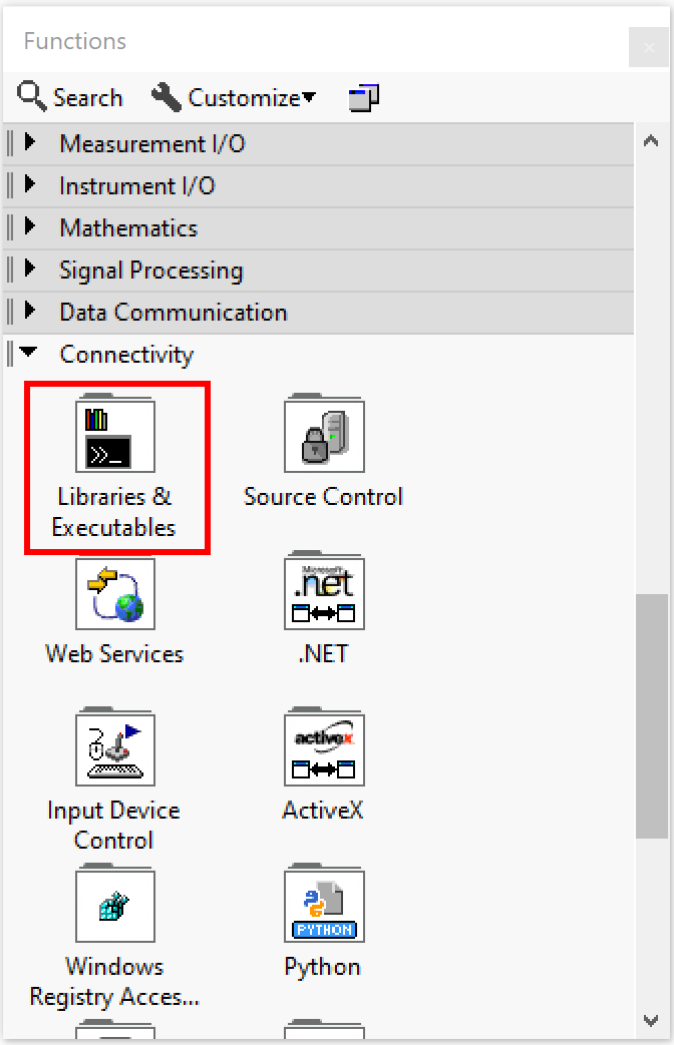
Die Informationen in diesem Dokument können ohne Vorankündigung geändert werden und stellen keine Verpflichtung von Omni Elektronik dar. Omni Elektronik übernimmt keine Verantwortung für Fehler oder Auslassungen in diesem Dokument. In keinem Fall haften Omni Elektronik AB, seine Mitarbeiter, seine Auftragnehmer oder die Autoren dieses Dokuments für Schäden, Verluste, Kosten, Gebühren, Ansprüche, Forderungen, Ansprüche auf entgangenen Gewinn, Gebühren oder Ausgaben jeglicher Art oder Art.

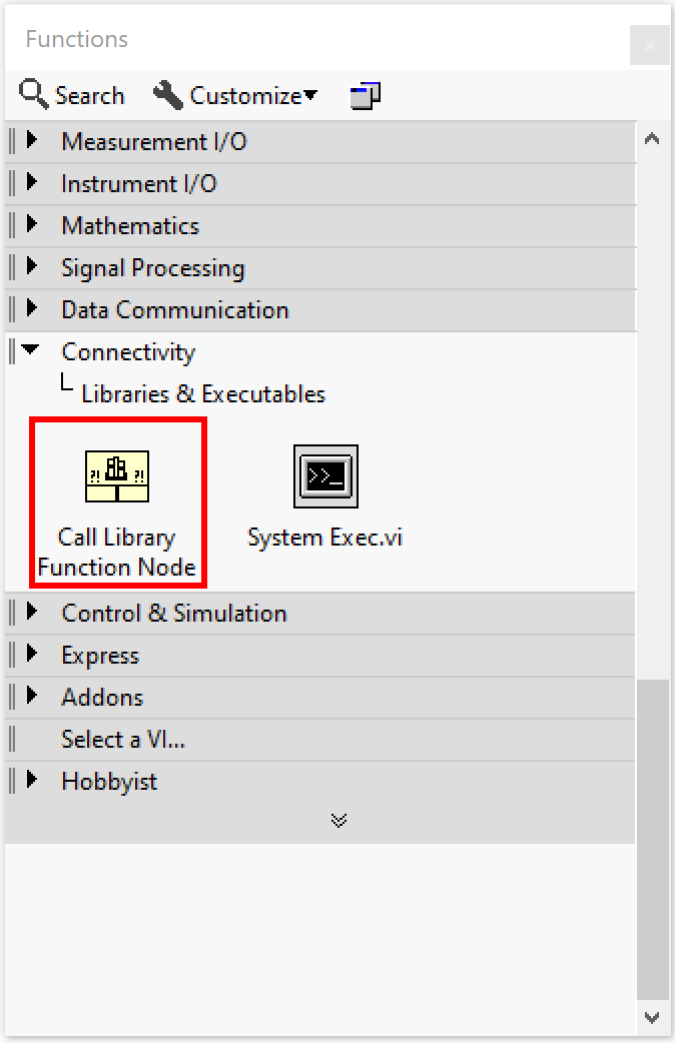
Inhalt

1 In LabVIEW auf den Sensor zugreifen

Der Zugriff auf den Sensor in LabVIEW erfolgt über den Call Library Block.

Um diesen in Ihrem Block Diagramm hinzuzufügen, wählen Sie unter Funktion „Connectivity -> Libraries & Executables -> Call Library Function Node“ aus. Den Block platzieren Sie dann in Ihrem Block Diagramm

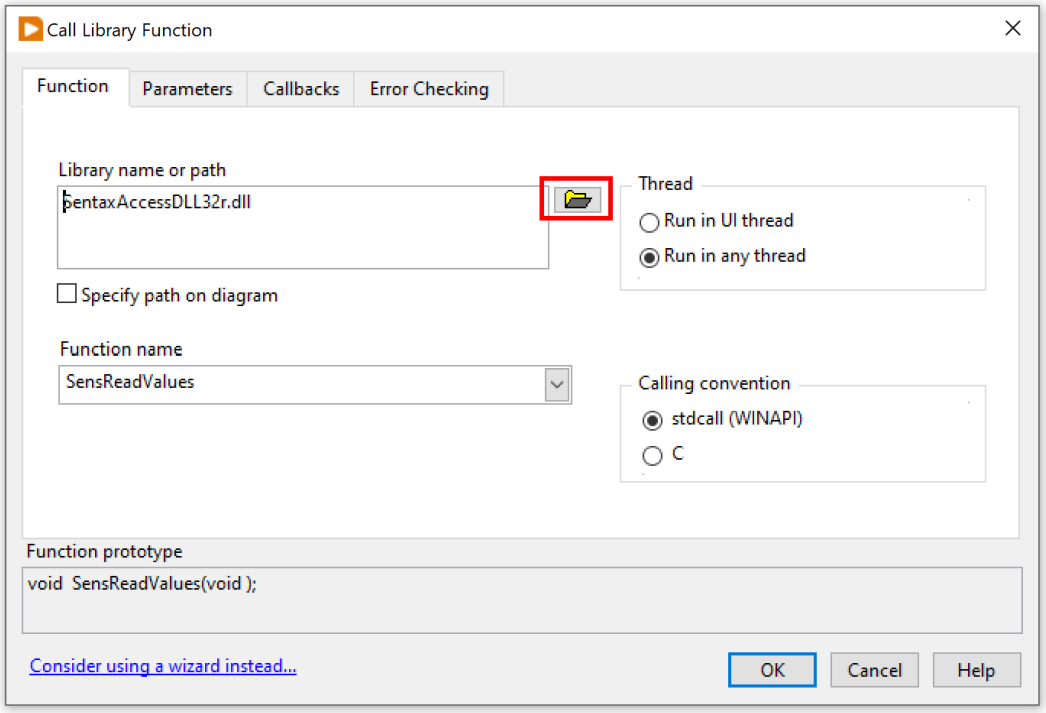




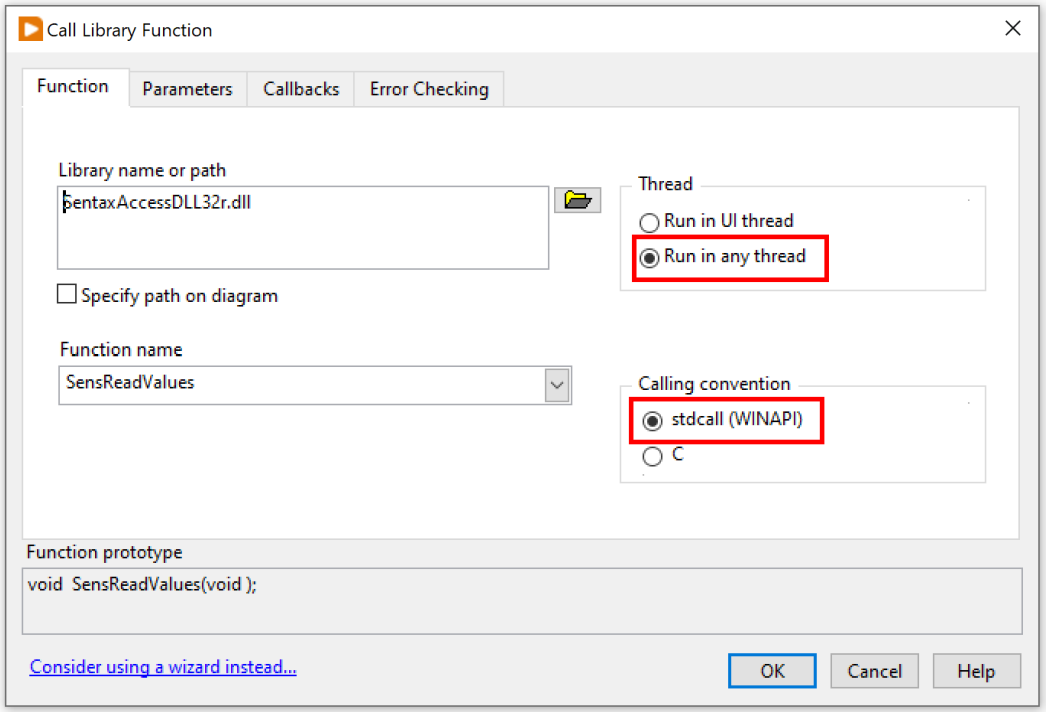
Nachdem der Block in Ihrem Block Diagramm ist, können Sie diesen nun mit einem Doppelklick öffnen und bearbeiten.

1.1 Call Library Block Konfigurieren

In dem Menü müssen Sie zuerst die Bereitgestellte DLL auswählen. Dazu wählen Sie unter der Datei Auswahl für „Library name or Path“ entweder die 32-Bit oder die 64-Bit DLL aus, je nachdem welche Ihrem System entspricht.



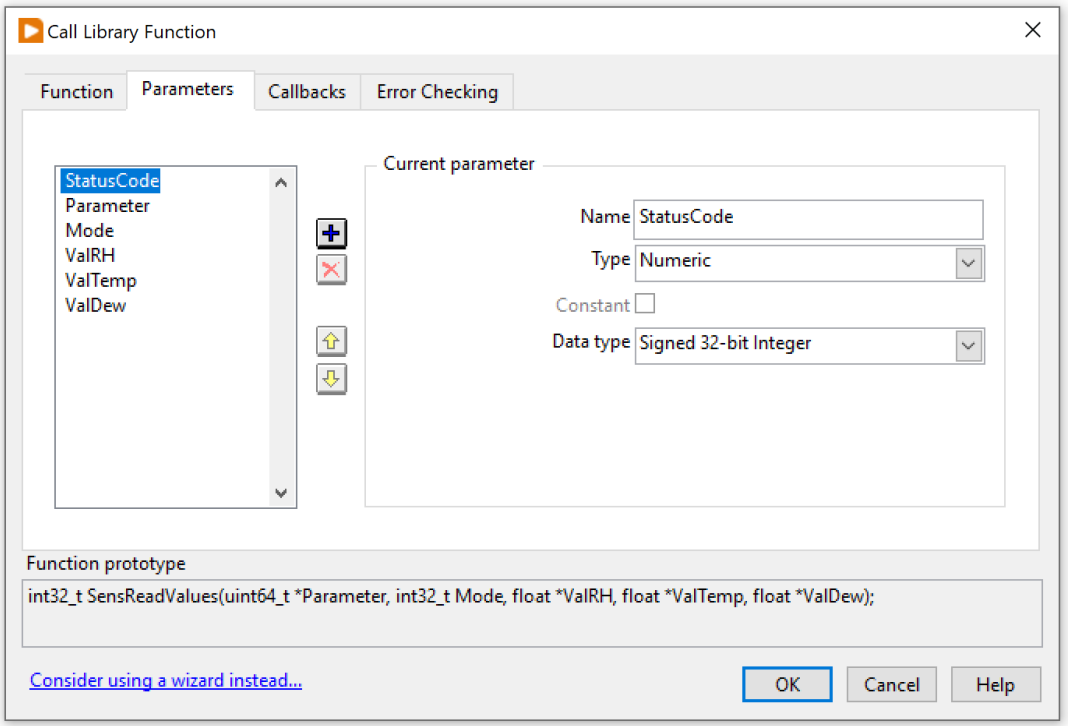
Danach wählen Sie unter Thread „Run in any thread“ und unter Calling Konvention „stdcall (WINAPI)“ aus



Nun können sie unter „Function name“ die von Ihnen benötigte Funktion auswählen.

Im Folgen wird beispielhaft an der „SensReadValues“ Funktion gezeigt wie die Parameter zu konfigurieren sind.

Dazu gehen Sie in den Parameter Reiter. Dort ändern Sie von dem „return type“ der Name auf „StatusCode“ und ändern den Typ auf Numeric. Der Daten Typ sollte „Signed 32-bit Integer“ sein.



Nun müssen Sie die Parameter der Funktion hinzufügen. Welche Parameter dies sein soll können Sie aus der Beschreibung der DLL-Funktionen entnehmen.

In dem Fall der „SensReadValues“ Funktion sind in Reihenfolge das

- uint64_t *Parameter
- int32_t Mode
- float *ValRH
- float *ValTemp
- float *ValDew

Um ein Parameter hinzuzufügen, betätigen Sie den Button mit dem Plus Symbol. Danach ändern Sie den Namen auf dem Namen des Parameters.

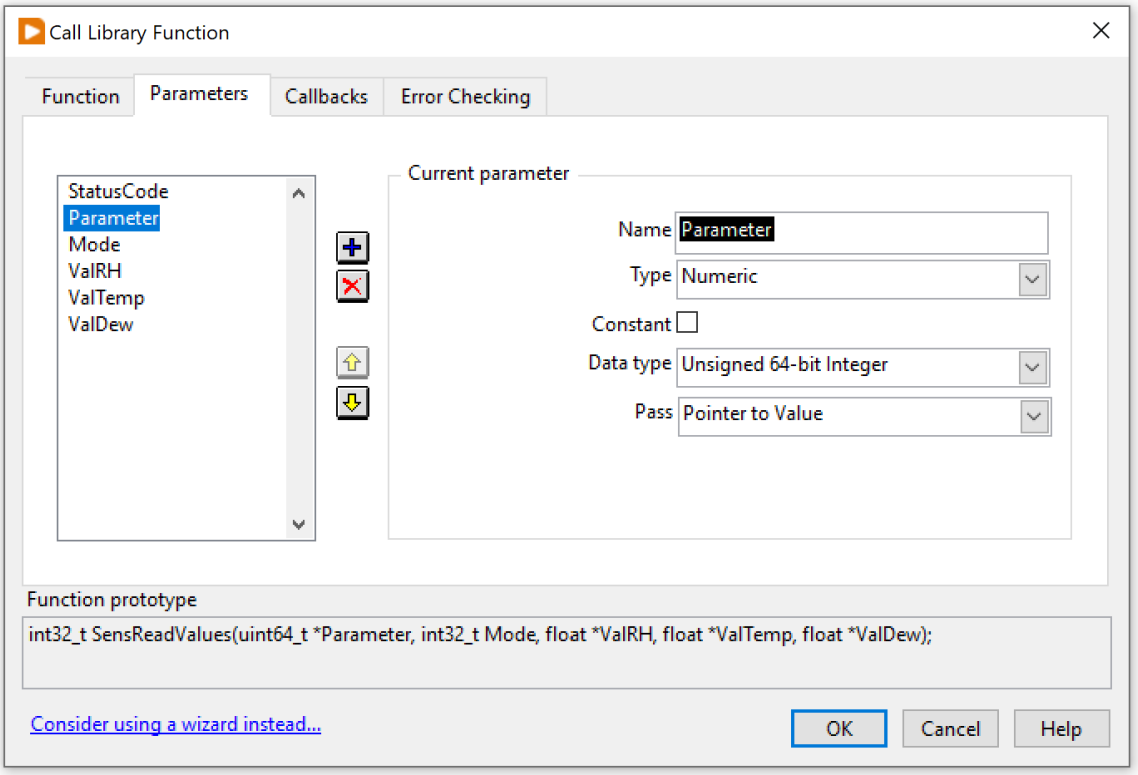
Falls der Parameter ein Numerischer Typ ist, wie uint64_t dann wähle „Numeric“ unter Typ aus. Falls es einen Zeichenketten Zeiger ist, so wählen sie String aus.

Enthält der Parameter Typ ein „u“ so handelt es sich um ein „Unsigned“ Datentyp, ansonsten ein „Signed“. Bei einem Integer wie „int64“ wählen sie den Integer mit der Korrespondieren Bit Anzahl. Sollte es sich um ein “float” handelt dann wählen Sie 4-Bit Single und bei einem „double“ 8-Bit Double. Zeichen wie „char“ und „wchar“ müssen als numerische Werte behandelt werden.

Sollte eine Zeiger auf eine C Struktur übergeben werden müssen, so sollte dann für den Typ „Adapt to Type“ ausgewählt werden und ein Cluster der der Daten Struktur entspricht übergeben werden.

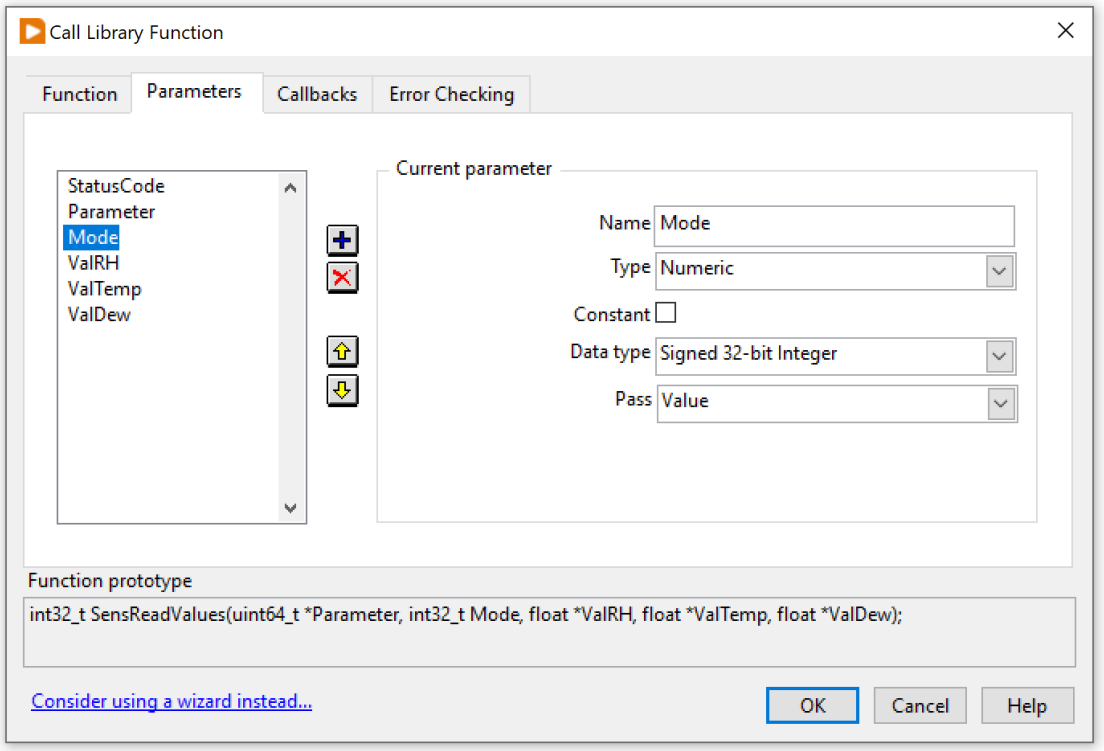
Sollte der Daten Typ ein Stern „*“ enthalten so handelt es sich um einen Pointer. In dem Fall muss Pass auf „Pointer to Value“ geändert werden, ansonsten „Value“

Also in dem Fall für „uint64_t *Parameter“ sollte es wie folgt aussehen.



In den Funktion Prototyp können Sie sehen, ob Sie den Parameter korrekt hinzugefügt wurde. Der Parameter daraus sollte dem Parameter aus der Beschreibung entsprechen.

Für den Paramter „int32_t Mode“ sollte es wie folgt aus sehen.



Nachdem alle Parameter hinzugefügt wurden, sollte der Funktion Prototyp dem der Beschreibung gleichen.

Nun ist der Call Library Funktion Block fertig konfiguriert. Nun können Sie den Block wie gewohnt in LabVIEW benutzen. Für ein Beispiel wie Sie den Block integrieren können schauen Sie sich das Beispielprojet an das bereitgestellt wird.

2 Funktionen der DLL

Die DLL stellt einige verschiedene Funktionen zur Verfügung, um mit dem Sensor zu interagieren.

Die UTF-8 Funktionen nehmen oder geben Zeichenketten in UTF-8 Kodierung an oder zurück. Bei den Unicode Funktionen sind die Zeichenketten in Unicode kodiert.

Diese sind wie folgt:

Für UTF-8:

- SensFindDevice (alt. SensFindDeviceA)
- SensReadValues (alt. SensReadValuesA)
- SensGetChangeFlag (alt. SensGetChangeFlagA)
- SensWaitReady (alt. SensWaitReadyA)
- getDeviceA
- getTaskA

Für Unicode:

- SensFindDeviceW
- SensReadValuesW
- SensGetChangeFlagW
- SensWaitReadyW
- getDeviceW
- getTaskW

Einige der Funktionen geben einen Fehler Code zurück. Dieser gibt an, ob die Funktion erfolgreich ausgeführt wurde oder ob ein Fehler während der Ausführung aufgetreten ist. Wenn ein Fehler aufgetreten ist, wird durch die Rückgabe dann auch angeben welcher.

Bezeichnung	Wert	Beschreibung
SENS_SUCCESS	0	Die Funktion wurde erfolgreich ausgeführt, es trat kein Fehler auf.
SENS_FAILED	-1	Es ist ein allgemeines, nicht näher beschreibbares Problem aufgetreten. Der Fehler wird z.B. geliefert, wenn ein Sensorzugriff während der Gerätesuche erfolgt, auf diese warten muss, und dabei ein Timeout auftritt. Da nicht festgestellt werden kann, warum die Suche so lange dauert, wird ein allgemeiner Fehler geliefert.
SENS_NOT_FOUND	-2	Es wurde ein Sensor angegeben, der im lokalen USB nicht registriert ist. Z.B. wurde eine ungültige Seriennummer verwendet, oder das Gerät wurde in der Zwischenzeit entfernt, oder die Gerätesuche fand kein weiteres Gerät.
SENS_UNABLE_TO_OPEN	-3	Die DLL kann den Zugriff über den Gerätetreiber nicht öffnen. Der Fehler kann auftreten, wenn das Gerät zur Zeit von einer anderen Anwendung aus verwendet wird.
SENS_IO_ERROR	-4	Ein Kommunikationsfehler zwischen Sensor und PC ist aufgetreten. Der Fehler sollte nicht auftreten und deutet meist auf ein technisches Problem hin.

SENS_TYPE_NOT_SUPPORTET	-5	Das gewählte Gerät scheint zwar ein MELTEC Sensor zu sein, wird jedoch von dieser DLL nicht unterstützt.
SENS_DEVICE_NOT_READY	-6	Das angesprochene Gerät ist (noch) nicht bereit. Der Fehler kann nach dem Einschalten eines Sensors auftreten, wenn Messwerte abgefragt werden, bevor diese vorliegen. Die einzelnen Sensoren benötigen meist einige Sekunden, bis die erste Messung verfügbar ist. Es kann auch sein, dass kein Sensorkopf auf die Sensorelektronik aufgesteckt wurde.
SENS_RH_NOT_MEASURED	-7	Der Feuchtemesswert wurde (noch) nicht ermittelt oder steht aus einem anderen Grund nicht zur Verfügung.
SENS_TEMP_NOT_MEASURED	-8	Der Temperaturmesswert wurde (noch) nicht ermittelt oder steht aus einem anderen Grund nicht zur Verfügung.
SENS_INVALID_MEASUREMENT	-9	Die Messwerte sind zurzeit ungültig, z.B. liefert der Sensorkopf keine Messwerte.
SENS_INVALID_FUNCTION	-10	Die Funktion ist unzulässig. Der Fehler entsteht z.B. dann, wenn versucht wird, bei einem älteren Sensor ein nicht vorhandenes Heizelement zu aktivieren.

2.1 SensFindDevice

Prototyp:

```
int32_t SensFindDeviceA (int32_t n, uint64_t * pszMask, void * pDevice);
int32_t SensFindDeviceW (int32_t n, uint64_t * pwszMask, void * pDevice);
```

Beschreibung:

Funktion durchsucht die aktuelle Geräteliste nach dem n-ten Gerät und füllt den Parameterblock mit dessen Parametern. Wenn das Gerät gefunden wurde, dann wird SENS_SUCCESS geliefert, sonst SENS_NOT_FOUND. Für die Suche kann ein Filterstring angegeben werden, der eine Auswahl nach bestimmten Gerätetypen erlaubt. Nicht erforderliche Parameter müssen als NULL übergeben werden. Um alle an den lokalen PC angeschlossenen Geräte aufzulisten, kann die Funktion in einer Schleife mit steigendem n so lange aufgerufen werden, wie SENS_SUCCESS geliefert wird.

Parameter	Typ	Beschreibung
n	Int32_t	Gibt den Index (n) des gewünschten Gerätes vor. Es wird bei 0 begonnen. Die Funktion liefert die Daten des n-ten Gerätes im USB zurück, welches den geforderten Parametern entspricht. Dabei werden alle mit dem PC verbundenen Sensoren berücksichtigt und der Reihe nach durchsucht
pszMask pwszMask	uint64_t */ CStr	Adresse eines Filterstrings oder NULL. Wenn eine String Adresse angegeben wird, dann werden nur Geräte berücksichtigt, in deren Typ-Bezeichnung dieser String an beliebiger Stelle enthalten ist.
pDevice	PSENSDEVICE	Zeiger auf einen Puffer mit der Struktur „SENSDEVICE“ oder NULL. Wird eine Pufferadresse angegeben, dann wird der Puffer bei erfolgreicher Suche mit den Daten des gesuchten Gerätes gefüllt.

Return Wert:
Die Funktion liefert SENS_SUCCESS, wenn das gewünschte Gerät gefunden wurde, sonst SENS_NOT_FOUND.

Der Puffer für die Gerätedaten ist wie folgt definiert:

```
typedef struct _SENSDEVICEA                    // Bus-Verzeichnis Eintrag, ASCII Code Version
{
        char        szTypeName[32];            // Gerätetype-Bezeichnung   char(uint16_t)
        char        szSerialNo[32];            // Seriennummer des Gerätes char (uint16_t)
        int32_t     nIndex;                    // Geräteindex

} SENSDEVICEA, * PSENSDEVICEA;

typedef struct _SENSDEVICEW                    // Bus-Verzeichnis Eintrag, Unicode Version
{
        wchar       szTypeName[32];            // Gerätetype-Bezeichnung
        wchar       szSerialNo[32];            // Seriennummer des Gerätes
        int32_t     nIndex;                    // Geräteindex

} SENSDEVICEW, * PSENSDEVICEW;
```

Hinweise:

Diese Funktion liefert alle wichtigen Informationen über die derzeit angeschlossenen Geräte. Durch multiplen Aufruf kann z.B. eine List Box als Geräteliste aufgebaut werden. Die Funktion greift dabei auf die DLL interne Liste der Geräte zu, wobei alle Geräte an allen USB-Schnittstellen der Reihe nach indiziert werden. Es werden maximal 50 Geräte gleichzeitig unterstützt. Die COM Port Nummer der virtuellen COM-Ports muss zwischen 1 und 999 liegen.

2.2 SensReadValues

Prototyp:

```
int32_t SensReadValuesA(uint64_t *Parameter, uint32_t bMode, float * pfValRH, float * pfValTemp,
float * pfValDew);

int32_t SensReadValuesW(uint64_t *Parameter, uint32_t bMode, float * pfValRH, float * pfValTemp,
float * pfValDew);
```

Beschreibung:

Funktion zur Abfrage aktueller Messwerte eines Gerätes. Es werden abhängig vom abgefragten Gerät bis zu 3 Messwerte geliefert, z.B. beim UFT75-AT Sensor relative Feuchte, Temperatur und Taupunkttemperatur. Messwerte für relative Feuchte

Parameter	Typ	Beschreibung
Parameter	uint64_t */ CStr	Parameter ist abhängig von „bMode“. Wird als Modus SENS_READ_BY_SERIAL_NUMBER angegeben, so ist „Parameter“ ein Zeiger auf einen nullterminierten String (ASCII oder Unicode), der die vollständige Seriennummer des Sensors enthält. Ist „bMode“ SENS_READ_BY_INDEX, so wird als Parameter der Geräteindex n angegeben, der bei der Suchfunktion SensFindDevice(n, ...) benutzt wurde, um das Gerät zu identifizieren.
bMode	uint32_t	Adressierungsmodus für Sensorauswahl. Wie zuvor beschrieben, kann der Sensor entweder über die Seriennummer, der den Geräteindex identifiziert werden.
pfValRH	float *	Zeiger auf eine Variable des Typen float, die den Messwert der relativen Feuchte aufnehmen soll (4-Byte Fließkomma). Der Feuchtemesswert kann nur zwischen 0.0 und 100.0 % liegen. Wird der Pointer nicht angegeben (NULL), so entfällt die Zuweisung dieses Wertes. Wird auf einen Sensor zugegriffen, der keine Feuchte misst, dann wird 0.0% geliefert.
pfValTemp	float *	Zeiger auf eine Variable des Typen float, die den aktuellen Temperaturmesswert aufnehmen soll (4-Byte Fließkomma). Der Temperaturmesswert kann nur zwischen -40.0 und +120.0 °C liegen. Wird der Pointer nicht angegeben (NULL), so entfällt die Zuweisung dieses Wertes. Wird auf einen Sensor zugegriffen, der keine Temperatur misst, dann wird -40.0°C zugewiesen.
pfValDew	float *	Zeiger auf eine Variable des Typen float, welche die berechnete Taupunkttemperatur aufnehmen soll (4-Byte Fließkomma). Der Taupunkt wird im PC aus den Messwerten der relativen Feuchte und der Temperatur berechnet. Deshalb kann dieser Wert nur dann verfügbar sein, wenn beide anderen Werte ebenfalls verfügbar sind. Der Wert kann nur zwischen -40.0 und +120.0 °C liegen. Wird der Pointer nicht angegeben (NULL), so entfällt die Zuweisung dieses Wertes. Kann der Wert nicht berechnet werden, so wird -40.0°C geliefert.

Adressierungsmodus	Wert	Funktion
SENS_READ_BY_-SERIAL_NUMBER	0	Der Funktionsparameter „Parameter” enthält einen Zeiger auf einen nullterminierten String, der die Seriennummer des Sensors enthält.
SENS_READ_BY_INDEX	1	Der Funktionsparameter “Parameter” enthält den Geräteindex, der auch verwendet wurde, um mit der Funktion „SensFindDevice()“ die Geräteparameter zu ermitteln. ACHTUNG: Der Index eines Gerätes kann sich verändern, wenn die USB-Konfiguration sich ändert. Es wird daher immer empfehlen, den Sensor über seine Seriennummer auszuwählen.

Return Wert:

Die Funktion liefert einen Fehlercode vom Typ „SENS_xxx“ als Funktionsergebnis zurück.

Hinweise: Feuchtemesswerte, die nicht verfügbar sind, werden als 0.0% geliefert. Temperaturmesswerte, die nicht verfügbar sind, werden als –40.0 °Celsius geliefert. Die Berechnung der Taupunkttemperatur kann nur dann erfolgen, wenn sowohl Feuchte als auch Temperatur vom Sensor gemessen wurden. Die Sensoren benötigen meist einige Sekunden nach dem Einschalten oder dem Wechseln des Sensorkopfes, bis die entsprechenden Werte verfügbar sind.

2.3 SensGetChangeFlag

Prototyp:

```
int32_t SensGetChangeFlagA(void);
int32_t SensGetChangeFlagW(void);
```

Beschreibung:

Funktion liefert TRUE (1), wenn sich die Sensorkonfiguration seit dem letzten Aufruf verändert hat (neue Geräte oder Geräte entfernt), bzw. FLASE (0), wenn keine Veränderung stattfand.

Return Wert:

Die Funktion liefert TRUE, wenn sich die Gerätekonfiguration seit dem letzten Aufruf verändert hat, und FALSE wenn nicht. Die ASCII Code und die Unicode Versionen sind identisch, jedoch aus kompatibilitätsgründen sind beide Varianten implementiert.

Hinweise:

Falls TRUE geliefert wird, dann sollte die neue Gerätekonfiguration unbedingt mit der Funktion SensFindDevice() neu ermittelt werden, da Geräte entfernt worden sein könnten bzw. neue Sensoren hinzugefügt wurden. Wenn das Betriebssystem eine Änderung der USB-Konfiguration meldet, dann veranlasst die DLL automatisch die Suche nach neuen Sensorgeräten. Diese wird multithreaded ausgeführt und muss mit der Applikation synchronisiert werden, um eine aktuelle Sensorliste zu erhalten. Die Sensorsuche ist normalerweise nach 150 bis 500 Millisekunden beendet. Die DLL unterstützt maximal 999 Sensorgeräte an den COM-Ports 1 bis 999. Sensoren, auf die während der Gerätesuche von anderer Seite aus zugergriffen wird, werden möglicherweise nicht erkannt.

2.4 SensWaitReady

Prototyp:

```
int32_t SensWaitReadyA(int32_t nTimeout);
int32_t SensWaitReadyW(int32_t nTimeout);
```

Beschreibung:

Die Funktion wartet auf den Abschluss der Gerätesuche maximal für die angegebene Länge in Millisekunden und liefert TRUE (1), wenn die Suche nach Sensorgeräten innerhalb der angegebenen Zeit beendet wurde. Die Applikation kann damit mit den Threads der DLL synchronisiert werden.

Return Wert:

Die Funktion liefert TRUE (1), wenn die Suche nach Sensorgeräten innerhalb der angegebenen Zeit beendet wurde oder nicht mehr läuft. Es wird FALSE (0) geliefert, wenn die Gerätesuche im angegebenen Zeitraum noch nicht abgeschlossen werden konnte oder ein anderer Fehler auftrat.

Hinweise:

Aufgrund der vollständig als multithreaded ausgelegten Implementierung der DLL-Funktionen kann die Anwendung bereits die Geräteliste abfragen, während die Gerätesuche im Hintergrund noch läuft. Dies kann nach dem Start der Applikation problematisch sein, da die DLL die Ausführung der Applikation nicht blockiert. Somit sind meist noch keine Sensorgeräte erkannt worden, wenn die Applikation direkt nach dem Start bereits die Sensorliste abfragt. Die Funktion "SensWaitReady()" sollte dazu verwendet werden, mindestens die erste Gerätesuche mit der Applikation zu synchronisieren. Die multithreaded Implementierung hat jedoch große Vorteile, da in keinem Fall mehr ein Anwendungs-Thread durch eine DLL-Funktion blockiert werden kann. Außerdem dauert die Gerätesuche unabhängig von der Anzahl der angeschlossenen Sensoren fast immer gleich lange, da für jede Schnittstelle ein eigener Kommunikations-Thread verwendet wird. Üblicherweise sollten alle Sensoren nach ca. 150 Millisekunden erkannt worden sein, spätestens jedoch nach 500 Millisekunden.

2.5 GetRemoteComputerEnable

Prototyp:

```
int32_t GetRemoteComputerEnable (void);
```

Beschreibung:

Funktion liefert den Aktivierungs-Status der Remote-Computer-Suche zurück.

Return Wert:

Die Funktion liefert den Aktivierungsstatus für die Remote-Computer-Suche zurück. Es wird FALSE (0) für deaktiviert und TRUE (1) für aktiviert geliefert.

2.6 getDevice

Prototyp:

```
int32_t getDeviceA(int32_t mode, uint64_t *parameter, void* sensorDef);
int32_t getDeviceW(int32_t mode, uint64_t *parameter, void* sensorDef);
```

Beschreibung:

Funktion füllt eine „SentaxDevice“ Struktur mit den Basisdaten eines erkannten Sensorgerätes.

Parameter	Typ	Beschreibung
mode	int32_t	Beschreibt verwendeten den Modus für die Adressierung des Sensorgerätes. By Index =1. By Serial Nummer = 0.
parameter	uint64_t * / CStr	Parameter zur Adressierung des Sensorgerätes, abhängig vom Parameter „mode“.
sensorDef	SentaxDevice*	Adresse einer Datenstruktur zur Aufnahme der Parameter des adressierten Sensorgerätes.

Wenn der Modus „byIndex“ verwendet wird, dann muss der Parameter „parameter“ einen Zeiger auf eine Variable vom Typ „uint64_t“ enthalten, welche dann den Index des zu adressierenden Gerätes enthält.

Wird der Modus „bySerialNumber“ verwendet, dann muss der Parameter „parameter“ einen Zeiger auf einen String mit der Seriennummer des adressierten Gerätes enthalten.

Return Wert:

Die Funktion liefert S_OK oder einen Fehlercode (E_FAIL, E_...), siehe "WinError.h".

Als Antwort füllt die DLL die angegebene Datenstruktur des Typen „SentaxDevice“. Es gibt wieder je eine Variante für UTF-8 und eine für Unicode.

```
struct SentaxDeviceW {
        wchar_t deviceSerialNumber[22];
        wchar_t sensorName[32];
        int32_t countOfTasks;
        int32_t deviceIndex;
};
```

Die Sentax Sensoren unterstützen jeweils 1 bis 4 Tasks. Jeder Task steht für eine unabhängige Messung. Hier gibt es keine Abfrage mehr für z.B. Feuchte und Temperatur in Einem, sondern je einen Task für Feuchte-Messung und einen Task für Temperatur-Messung, und z.B. einen weiteren Task für Taupunktmessung und einen weiteren für Absolute Feuchte. Der Parameter „coundOfTasks“ besagt, wie viele Tasks vom Sensor unterstützt werden. Der Parameter „deviceIndex“ kann zur Adressierung des Sensors über den Index verwendet werden. Die Tasks sind abhängig vom verwendeten Sensorgerät.

2.7 getTask()

Prototyp:

```
int32_t getTaskA(CStr  deviceSerialNumber, uint64_t taskIndex, void* sensorTask);
int32_t getTaskW(CStr deviceSerialNumber, uint64_t taskIndex, void* sensorTask);
```

Beschreibung:

Funktion füllt eine „SentaxTask“ Struktur mit den Messdaten und Eigenschaften des adressierten Sensorgerätes.

Parameter	Typ	Beschreibung
deviceSerialNumber	CStr	Seriennummer zur Adressierung des Sensorgerätes.
taskIndex	uint64_t	Index des adressierten Tasks, muss zwischen 0 und „countOfTasks“ - 1 liegen.
sensorTask	SentaxTask*	Adresse einer Datenstruktur vom Typ „SentaxTask“ zur Aufnahme der gelesenen Task Parameter.

Return Wert:

Die Funktion liefert S_OK oder einen Fehlercode (E_FAIL, E_...), siehe "WinError.h".

Die Datenstruktur zur Speicherung des Abfrageergebnisses ist wie folgt aufgebaut (Unicode variante):

```
struct SentaxTaskW {
        wchar_t name[32];
        wchar_t unit[8];
        float pointer;
        float min;
        float max;
        int32_t status;
};
```

Das Feld „name“ enthält den Namen des Tasks (also der Messstelle), z.B. „Feuchte“. Das Feld „unit“ entsprechend der verwendeten Einheit. Das Feld „pointer“ ist der letzte Messwert in der angegebenen Einheit. Die Parameter „min“ und „max“ definierten den Messbereich der aktuell eingestellten Skala. Das Feld „status“ enthält den Sensor-Status entsprechend den alten Status Codes.