Omni Sensors USB Transmitter System

Kommunikationsprotokoll

Embedded DLL

©2022 by Omni Elektronik GmbH,

Ringstr. 12, 51766 Engelskirchen, Tel. +49 2263 90 29 51 0, Internet www.omni-sensors.de
Technische Änderungen vorbehalten.

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.

©2022 by Omni Elektronik GmbH,

Ringstr. 12, 51766 Engelskirchen, Tel. +49 2263 90 29 51 0, Internet www.omni-sensors.de
Technische Änderungen vorbehalten.

Inhaltsverzeichnis

©2022 by Omni Elektronik GmbH,

Ringstr. 12, 51766 Engelskirchen, Tel. +49 2263 90 29 51 0, Internet www.omni-sensors.de
Technische Änderungen vorbehalten.

1 Einleitung

Omni Sensoren messen relative Feuchte, Temperatur, Taupunkt und absolute Feuchte. Sie werden direkt am USB Port eines PCs betrieben. Die Omni Sensoren werden am USB Port eines Computers betrieben. Die Verwendung unter Microsoft Windows ist ebenso möglich, wie die Verwendung unter Linux. Es können demnach auch Anwendungen auf dem Raspberry Pi oder einem anderen Computersystem mit USB Schnittstelle realisiert werden.

In Kapitel 5 wird das Kommunikationsprotokoll zur direkten Ansteuerung der Omni Sensoren durch eigene Anwendungen beschrieben.

Das Kapitel 6 beschreibt die Verwendung einer Embedded DLL zur Kommunikation mit den Omni Sensoren in eigenen Anwendungen und in Software von Drittanbietern (z.B. LabView). Diese DLL kann als Vereinfachung der Softwareimplementation dienen, da sie die kontinuierliche Verwaltung und Datenerfassung angeschlossener Sensoren übernimmt.

Weitere Informationen, sowie Applikationsbeispiele zur Kommunikation mit den Sensoren und der Verwendung der DLL, stellen wir Ihnen gerne auf Anfrage zur Verfügung.

info@omnielektronik.de

Bitte nennen Sie uns neben Ihren Kontaktdaten wenn möglich auch Informationen zur Zielanwendung (z.B. Typ und Anzahl der Sensoren, Programmiersprache und Entwicklungsumgebung, Drittanbietersoftware, Hardware, etc.).

2 Unterstützte Sensortypen

Zurzeit werden die folgenden Sensortypen unterstützt: OHT20, OT60, OT-150

3 Systemvoraussetzungen

Diese Dokumentation bezieht sich auf die Verwendung der Omni Sensoren auf einem Computer mit Microsoft Windows.

Die Omni Sensoren verwenden eine VCP-Schnittstelle nach dem USB CDC-Class Protokoll (https://www.usb.org/document-library/class-definitions-communication-devices-12). Dabei handelt es sich um einen standardisierten USB zu RS232 Konverter Die Omni Sensoren unterstützen eine Teilmenge des USB CDC Protokolls für die Datenübertragung. Die Auswahl der Baudrate oder anderer RS232 Kommunikationsparameter ist nicht erforderlich. Eventuelle Einstellungen werden ignoriert.

Die grundsätzliche Kompatibilität zum CDC Protokoll erlaubt es, einen CDC Class Treiber des Host-Betriebssystems (Windows Standardtreiber, oder die Entsprechungen in anderen Betriebssystemen)

©2022 by Omni Elektronik GmbH,

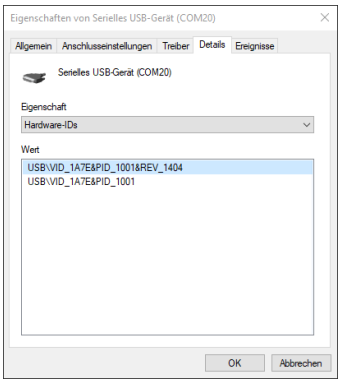
Ringstr. 12, 51766 Engelskirchen, Tel. +49 2263 90 29 51 0, Internet www.omni-sensors.de
Technische Änderungen vorbehalten.

verwenden zu können. So können die Sensoren z.B. auch unter Linux über das jeweilige Kommunikations-Device direkt angesprochen werden.

Ein entsprechender Gerätetreiber ist in neuen Windows Versionen (ab Windows 10) bereits enthalten. Für die Installation mit älteren Windows Versionen (vor Windows 10) wird möglicherweise eine INF Datei benötigt, welche dem System mitteilt, dass ein bestimmter Sensor mit dem CDC VCP betrieben werden soll.

Alle Sensortypen verwenden die Vendor-ID 1A7E und eine geräteabhängige Device-ID, z.B. 1001 für OHT20, OT150 oder OT60 Sensoren. Der genaue Sensortyp wird durch Abfrage des ID-Strings identifiziert.

Anmerkung: Unter Windows lässt sich die Vendor-ID mit dem Gerätemanager ermitteln. Beispiel:



4 Treiberinstallation mit Windows 8 und Windows 8.1

Obwohl ein originaler Treiber von Microsoft Verwendung findet, sind die Windows 8 Varianten nicht in der Lage dies zu erkennen und verlangen eine gesonderte Signatur in der Installations-INF-Datei. Weil diese nicht vorhanden ist, muss bei der Verwendung von Windows 8 und Windows 8.1 zur Installation der Sensorgeräte die Abfrage der Treibersignatur einmalig abgeschaltet werden. Dazu wird zunächst ein erweiterter Systemstart ausgeführt. Im dann angezeigten Menü wird die Treibersignaturabfrage dann abgeschaltet. Der Treiber kann dann anschließend installiert werden. Nach weiteren Neustarts ist diese Freischaltung wieder deaktiviert, daher muss diese Prozedur für jede folgende Treiberinstallation wiederholt werden.

Bei noch älteren Windows-Versionen ist der CDC Treiber bereits im System integriert, jedoch erkennen dies Windows-Versionen das CDC-Protokoll nicht automatisch und es bedarf einer zusätzlichen INF Datei um die Installation zu steuern.

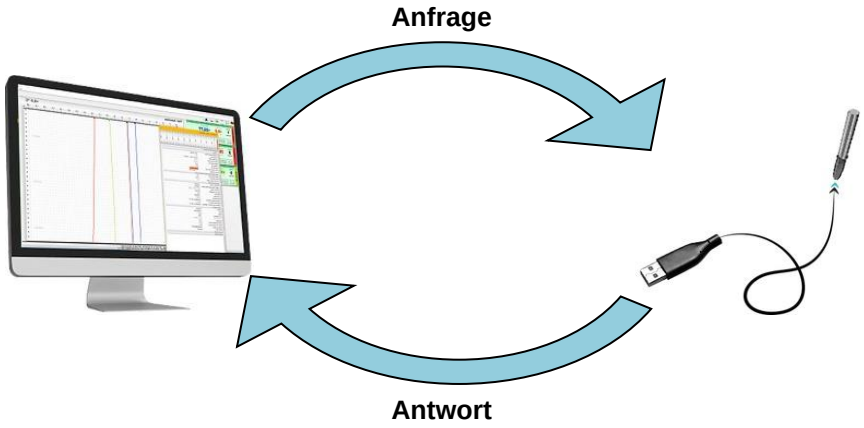
©2022 by Omni Elektronik GmbH,

Ringstr. 12, 51766 Engelskirchen, Tel. +49 2263 90 29 51 0, Internet www.omni-sensors.de
Technische Änderungen vorbehalten.

5 Kommunikationsprotokoll

Die Omni Sensoren verwenden ein Kommunikationsprotokoll mit variabler Datensatzlänge. Die Übertragung wird in Form einzelner Transaktionstelegramme ausgeführt. Der Host sendet ein Anforderungstelegramm, der Sensor beantwortet dieses mit einem Antworttelegramm.

Jedes Telegramm besteht aus zwei Kommandobytes, gefolgt von bis zu 62 Datenbytes. Die minimale Telegrammlänge beträgt 2 Bytes (nur Kommando, keine Daten). Die maximale Telegrammlänge beträgt 64 Bytes (max. 62 Bytes Daten).



Anfrage vom Host

Ein Anfragetelegramm beginnt mit einem Kommandobyte CMD, gefolgt vom bitweise invertierten Kommandobyte $\overline{CMD}$, anschließend folgen die Daten.

Byte 1	Byte 2	Byte 3 … Byte 64
CMD	$\overline{CMD}$	$DATA$

Antwort vom Sensor

Ein Antworttelegramm beginnt mit dem bitweise invertierten Kommandobyte $\overline{CMD}$, gefolgt vom Kommandobyte CMD, anschließend folgen die Daten.

Byte 0	Byte 1	Byte 2 … Byte N
$\overline{CMD}$	CMD	$DATA$

Die Übertragung von CMD und $\overline{CMD}$ dient der Erkennung des Telegrammstarts durch den Sensor. Der Sensor antwortet nur auf ein gültiges Kommandobyte CMD, auf welches das invertierte Kommandobyte $\overline{CMD}$ folgt.

©2022 by Omni Elektronik GmbH,

Ringstr. 12, 51766 Engelskirchen, Tel. +49 2263 90 29 51 0, Internet www.omni-sensors.de
Technische Änderungen vorbehalten.

Die Übertragung von $\overline{CMD}$ und CMD durch den Sender dient dem Host zur Erkennung einer gültigen Antwort. Der Host erkennt das zugehörige Antworttelegramm bei Empfang der zuvor ausgesendeten Kommandobytes in umgekehrter Reihenfolge $\overline{CMD}$ gefolgt von CMD. Alle anderen Daten werden als Fehler betrachtet und ignoriert.

Das Aussenden des Antworttelegramms erfolgt nach einer systemabhängigen Latenzzeit. Als Transaktionstimeout ist eine Zeit von 100 ms normalerweise ausreichend.

Üblicherweise beginnt die Datenübertragung mit der Identifikation des Sensors. Die Host Software muss dann den Identifikationsstring auswerten und sich auf den erkannten Sensortyp einstellen.

Beispiel

Kommando zur Identifizierung eines Sensortyps

Anfragetelegramm: 0x00 , 0xFF

Antworttelegramm: 0xFF , 0x00 , “MELTEC OHT20-A 1.4.4.2“ , 0x00

5.1 Kommandocodes

Abhängig vom Sensortyp und der Firmware Version werden derzeit die folgenden Kommandos unterstützt (Stand 2021-03). Es können weitere Kommandos zu Testzwecken implementiert sein. Diese Kommandos sind für die Anwendung nicht relevant und werden nachfolgend nicht aufgeführt.

Kommando	Code	Funktion
UFTCMD_IDENTIFY	0x00	Identifikations-String des Sensors anfordern
UFTCMD_GETSERIALNO	0x01	Seriennummer des Sensors anfordern
UFTCMD_READMEASURE	0x02	Messdaten anfordern
UFTCMD_HEATING_ON	0x03	Heizelement des Sensors einschalten
UFTCMD_HEATING_OFF	0x04	Heizelement des Sensors ausschalten
UFTCMD_READMEASURE_EX	0x12	Erweiterte Messdaten anfordern

5.2 Kommando „UFTCMD_IDENTIFY“

Kommando	UFTCMD_IDENTIFY
Codefolge CMD, $\overline{CMD}$	0x00 , 0xFF (0 , 255)
Beschreibung	Identifikations-String des Sensors anfordern
Daten senden	0 , 255 (nur Kommando, keine Daten)
Daten empfangen	Der Sensor antwortet mit den beiden invertierten Kommandobytes und dem Identifikations-String. Der String ist <NULL> terminiert. Die max. mögliche Länge des Strings beträgt 62 Bytes, inklusive des abschließenden <NULL> Bytes.

©2022 by Omni Elektronik GmbH,

Ringstr. 12, 51766 Engelskirchen, Tel. +49 2263 90 29 51 0, Internet www.omni-sensors.de
Technische Änderungen vorbehalten.

Beispiel:	Host sendet: 0 , 255 Sensor antwortet: 255 , 0 , "MELTEC OHT20-A V1.4.4.2" , 0
Anmerkungen:	Der Identifikations-String muss analysiert werden um den Sensortyp festzustellen. Dazu wird der String nach den Typ-Strings der unterstützten Sensoren durchsucht (*Kapitel* 2). Der Versionscode der Firmware beginnt mit dem Zeichen "V".

5.3 Kommando „UFTCMD_GETSERIALNO“

Kommando	UFTCMD_GETSERIALNO
Codefolge $CMD, \overline{CMD}$	0x01 , 0xFE (1 , 254)
Beschreibung	Seriennummer des Sensors anfordern
Daten senden	1 , 254 (nur Kommando, keine Daten)
Daten empfangen	Der Sensor antwortet mit den beiden invertierten Kommandobytes und der Seriennummer als ASCII String. Die Seriennummer ist immer genau 20 Bytes lang, gefolgt von <NULL>. 254 , 1 , D_0 , ... , D_{19} , 0
Beispiel:	Host sendet: 1 , 254 Sensor antwortet: 254 , 1 , "20200803-125418-1404" , 0
Anmerkungen:	Der Seriennummer-String ist für jedes Sensorgerät eindeutig. Ein Sensor kann daher immer wiedererkannt werden, auch wenn das Gerät mit einem anderen USB Port verbunden wurde. Über die Seriennummer kann z.B. eine Sensor-spezifische Konfiguration verwaltet werden.

5.4 Kommando „UFTCMD_READMEASURE“

Kommando	UFTCMD_READMEASURE
Codefolge $CMD, \overline{CMD}$	0x02 , 0xFD (2 , 253)
Beschreibung	Messdaten anfordern
Daten senden	2 , 253 (nur Kommando, keine Daten)
Daten empfangen	Der Sensor antwortet mit den beiden invertierten Kommandobytes, 2 x 16-Bit Messwerten im Little Endian Format und einem Flag-Byte. 253 , 2 , HUM_{LSB} , HUM_{MSB} , $TEMP_{LSB}$, $TEMP_{MSB}$, FLAG
Beispiel:	Host sendet: 2 , 253 Sensor antwortet: 253 , 2 , 1 , 128 , 9 , 3 , 192
Anmerkungen:	Sensoren ohne Temperaturmessung liefern den Wert TEMP = 0 Sensoren ohne Feuchtemessung liefern den Wert HUM = 0

©2022 by Omni Elektronik GmbH,

Ringstr. 12, 51766 Engelskirchen, Tel. +49 2263 90 29 51 0, Internet www.omni-sensors.de
Technische Änderungen vorbehalten.

	Die Messdaten müssen abhängig vom Sensortyp in die Werte für Temperatur und relative Luftfeuchte umgerechnet werden (siehe **Berechnung der Messwerte**)

Universelle Struktur des Messdatensatzes (erweiterte Daten nur verfügbar mit Kommando UFTCMD_READMEASURE_EX, siehe unten):

```
class UFTDataRecord {
 public:
   // Legacy data (provided by all types):
   union {
     uint16_t humidityValue;    // 0 - 16-bit native humidity measurement result
     uint8_t bMode[2];          // 0 - 8-bit mode bytes (OT60, OT150)
   };

   uint16_t tempValue;          // 2 - 16-bit native temperature measurement result

   union {
     uint8_t bFlags;            // 4 - native sensor status flags
     struct {
       uint8_t errCount:4;      // . - sensor head transmission or access error counter
       uint8_t errOverflow:1;   // . - error counter overflow bit, invalid measurement if set
       uint8_t heatingOn:1;     // . - humidity sensor heating activation signal
       uint8_t tempValid:1;     // . - a valid temperature measurement was fetched
       uint8_t humidityValid:1; // . - a valid humidity measurement was fetched
     };
   };

   // Data record extension since OHT20:
   struct {
     uint8_t sensorTypeID;      // 5 - sensor type identifier
     uint8_t dataFormatID;      // 6 - data format identifier
     uint8_t sensorPayload;     // 7 - sensor specific payload (parameter) byte
   } Extension;
                              // ----
                              // 8 - bytes length (or 5 bytes for legacy records)
};
```

Messdaten-Flags (bFlags):
Der Wert errCount wird bei jedem erfolgreichem Lesevorgang zurückgesetzt (0). Bei einer fehlgeschlagenen Messdatenabfrage wird der Zähler jeweils um eins erhöht. Nach 16 fehlerhaften Lesevorgängen tritt ein Overflow auf das folgende Statusbit auf, wobei der Zählerstand zugleich wieder 0 wird. Passiert dies, werden die beiden Bits tempValid und humidityValid zurückgesetzt (0) und die Messwerte sind nicht mehr gültig. Bei einem erfolgreichen Lesevorgang wird das zugehörige Gültigkeits-Bit für den Messwert gesetzt. Bei Feuchte/Temperatur Sensoren wird immer die Temperatur zuerst gelesen, da diese in der Regel in die Berechnung der Feuchtemessung einfließen muss.

Berechnung der Messwerte

Die Omni Sensoren mit der USB Device ID 1001 liefern Rohdaten. Abhängig vom Sensortyp müssen aus diesen Rohdaten die Werte für Temperatur in °C und Luftfeuchte in %rel berechnet werden. Für die OHT20 Feuchtesensoren geht die Temperaturmessung in die Berechnung der relativen Luftfeuchte mit ein.

©2022 by Omni Elektronik GmbH,

Ringstr. 12, 51766 Engelskirchen, Tel. +49 2263 90 29 51 0, Internet www.omni-sensors.de
Technische Änderungen vorbehalten.

Sensor	Messwert	Berechnung
OHT20	Temperatur	`fTemp = ((float)(Record.wValTemp) * 175.0f / 65535.0f) - 45.0f;`
	Feuchte	`fRH = (float)(Record.wValRH) * 100.0f / 65535.0f;`
	Taupunkt	Siehe unten.

Sensor	Messwert	Berechnung

OT60
OT150
	Temperatur	`// identification, for OT60=0, for OT150=1`
		`if(Record.bMode[0] & 0x01) {`
		`  // for OT150`
		`  fValue = (float)((short)Record.wValTemp) * 200.0 / 2048.0 - 50.0;`
		`}`
		`else {`
		`  // for OT60`
		`  fValue = (float)((short)Record.wValTemp) * 70.0 / 2048.0 - 10.0;`
		`}`

Sensor	Messwert	Berechnung
Thermostick und IR-Sensoren (siehe unten)	Temperatur, Kaltstelle, Warmstelle	`Skalenabhängiger Messwert multipliziert mit 10.` `In der Regel °C * 10.`

Sensor	Messwert	Berechnung
ADC Stick	Relativ	`Normalisierter (0 … 1) Messwert multipliziert mit 10000.`

Die Taupunkttemperatur wird bei allen Feuchte/Temperatur Sensoren folgendermaßen berechnet:

Sensor	Messwert	Berechnung

Alle Feuchte und Temperatur Typen	Taupunkt	`sdd = 6.1078f * exp((17.08085f * fTemp) / (234.175f + fTemp));`
		
		`if(fTemp < 0.0) {`
		`  sdd = sdd * exp(0.00972f * fTemp);`
		`}`
		
		`dd = fRH * sdd / 100.0f;`
		
		`if(dd) {`
		`  fValue = log(fabs(dd / 6.1078f));`
		`  fDewpoint = 234.175f * fValue / (17.08085f - fValue);`
		`}`
		`else {`
		`  fDewpoint = 0.0; // not available`
		`}`

5.5 Kommando „UFTCMD_HEATING_ON“

Kommando	UFTCMD_HEATING_ON

©2022 by Omni Elektronik GmbH,

Ringstr. 12, 51766 Engelskirchen, Tel. +49 2263 90 29 51 0, Internet www.omni-sensors.de
Technische Änderungen vorbehalten.

Codefolge $CMD, \overline{CMD}$	0x03 , 0xFC (3,252)
Beschreibung	Heizelement des Sensors einschalten
Daten senden	3 , 252 (nur Kommando, keine Daten)
Daten empfangen	Der Sensor antwortet mit den beiden invertierten Kommandobytes und einem Statusbyte. Bit 3 des Statusbytes enthält den Status des Heizelements (0=aus, 1=ein).
Beispiel:	Host sendet: 3 , 252 Sensor antwortet: 252 , 3 , 4 (Heizelement ein)
Anmerkungen:	Das Kommando steht nur für OHT20 Feuchte-Sensoren zur Verfügung. Das Heizelement dient dem Trocknen des Sensorelements. Während das Heizelement aktiviert ist sind die Messwerte nicht verwendbar, da die Temperaturmessung ein falsches Ergebnis liefert.

5.6 Kommando „UFTCMD_HEATING_OFF“

Kommando	UFTCMD_HEATING_OFF
Codefolge $CMD, \overline{CMD}$	0x04 , 0xFB (4,251)
Beschreibung	Heizelement des Sensors ausschalten
Daten senden	4 , 251 (nur Kommando, keine Daten)
Daten empfangen	Der Sensor antwortet mit den beiden invertierten Kommandobytes und einem Statusbyte. Bit 3 des Statusbytes enthält den Status des Heizelements (0=aus, 1=ein).
Beispiel:	Host sendet: 4 , 251 Sensor antwortet: 251 , 4 , 0
Anmerkungen:	Das Kommando steht nur für OHT20 Feuchtesensoren zur Verfügung.

5.7 Kommando „UFTCMD_READMEASURE_EX“

Kommando	UFTCMD_READMEASURE_EX
Codefolge $CMD, \overline{CMD}$	0x12 , 0xED (18,237)
Beschreibung	Erweiterte Messdaten anfordern
Daten senden	Keine (nur Kommando)
Daten empfangen	Der Sensor antwortet mit den beiden invertierten Kommandobytes und 8 Bytes erweiterter Messdaten. 2 x 16-Bit Messwerten, Little Endian Flag Byte Sensortyp Kennung (siehe unten) Sensorkopf Kennung (siehe unten)

©2022 by Omni Elektronik GmbH,

Ringstr. 12, 51766 Engelskirchen, Tel. +49 2263 90 29 51 0, Internet www.omni-sensors.de
Technische Änderungen vorbehalten.

	Parameterbyte
Beispiel:	Host sendet: 18 , 237 Sensor antwortet: 237 , 18 , 234 , 0 , 221 , 0 , 192 , 30 , 16 , 75
Anmerkungen:	Dieser Datensatz ist eine Erweiterung des Datensatzes des Kommandos UFTCMD_READMEASURE. Die zusätzliche Information kann anstelle der Stringauswertung des Identifikationsstrings (Kommando „UFTCMD_IDENTIFY“) verwendet werden, um den Sensortyp zu identifizieren und die Messwert zu interpretieren.

Liste der Typ-Kennungen (Auswahl)

Sensor-Typ	Kennung	Interpretation der Messdaten
OHT20	1	Alte Typen, unterstützen nur UFTCMD_READMEASURE, Messdatenformat wie oben beschrieben.
OHT20-AT	2	
OT60	3	
OT150	4	
OHT20-ATN	10	Neuere Typen, unterstützen in der Regel schon die Protokollerweiterungen. Zur Wahrung der Kompatibilität ist das Messdatenformat unverändert zu den entsprechenden alten Typen. **Hinweis:** Die Antwort-Strings der Kommandos 0 und 1 werden bei diesen Sensoren nicht mehr mit Leerzeichen aufgefüllt und es wird auch kein <CR><LF> mehr angehängt.
OT150-ATN	13	
OT150-BTN	15	
OT60-ATN	12	
OT60-BTN	14	
MTF60-ATN	16	
MTF60-BTN	18	
MTF150-ATN	17	
MTF150-BTN	19	
OHT20-BTN	20	
OHT20-ST	21	
THERMOSTICK	30	Der Messwert entspricht der in Sentax konfigurierten Skala multipliziert mit 10. Bei diesen Temperatursensoren ist die Skala meist auf °C eingestellt, der Messwert ist also direkt die Temperatur in °C mal 10. Beispiel: 200 = 20.0°C.
IRM350	31	
THERMOTRANSMIT	32	
THERMOREFERENCE	33	
AUTOSMART-IR	34	
OT150-TI	50	Zur Wahrung der Abwärtskompatibilität mit alter Software, z.B. Poseidon, liefern diese Typen die Kennung ihrer jeweiligen Vorgängertypen, und haben das gleiche Datenformat.
OT60-TI	51	
MTF60-TI	52	
ADCSTICK	99	Der Messbereich wird auf 0 bis 10000 abgebildet.

©2022 by Omni Elektronik GmbH,

Ringstr. 12, 51766 Engelskirchen, Tel. +49 2263 90 29 51 0, Internet www.omni-sensors.de
Technische Änderungen vorbehalten.

Liste der Kennungen der Sensorköpfe

Kopf-Typ Kennung	Interpretation der Messdaten
0x01 (1)	Alter Feuchte- und Temperatursensor OHT20
0x02 (2)	Neuer Feuchte- und Temperatursensor OHT20
0x04 (4)	Alter Temperaturmesskopf OT60 oder OT150
0x08 (8)	Messkopf für Infrarottemperaturmessung
0x10 (16)	Messkopf für Thermoelemente und ähnliche IR Temperatursensoren
0x20 (32)	Neuer Temperaturmesskopf OT60 oder OT150
0x40 (64)	Analog zu Digital Konverter (ADC)
0x80 (128)	Teststecker/Brückenstecker

Anmerkung: Derzeitig erfolgt die Codierung der Sensorköpfe bitweise mit nur einem Bit. Zukünftige Sensorköpfe werden mit Bitkombinationen codiert!

Parameterbyte: Das Parameterbyte ist Sensorspezifisch zu interpretieren. Es wird derzeit nur beim Sensorkopf 0x10 (16d) verwendet, und gibt den konfigurierten Typ des Thermoelements an. Der Inhalt des Parameterbytes ist der ASCII-Code des Thermoelementtyps. Unterstützt werden derzeit die Typen: B, E, J, K, N, R, S und T, sowie verschiedene IR Kennlinienpolynome, gekennzeichnet durch Kleinbuchstaben.

©2022 by Omni Elektronik GmbH,

Ringstr. 12, 51766 Engelskirchen, Tel. +49 2263 90 29 51 0, Internet www.omni-sensors.de
Technische Änderungen vorbehalten.

6 Embedded DLL

Die DLL wurde für die Verwendung mit den aktuellen 32-Bit und 64-Bit Windows Betriebssystemen konzipiert und als WIN32 und X64 Implementation verfügbar. Sie ist mit Windows 10 und vielen älteren Windows Versionen kompatibel.

6.1 Implementierung der DLL Funktionen

Durch den Start der DLL werden verschiedene voneinander unabhängige Module aktiviert:

- Lokale Sensorsuche.
- Permanentes automatisches Lesen und Speichern der Messwerte aller lokalen Sensoren.
- Anwendungs-Schnittstelle für Zugriff auf die Daten.

Die Suche nach lokalen Sensoren wird einmalig beim Start der DLL ausgeführt, danach nur noch bei Gerätewechsel Ereignissen (Sensor eingesteckt oder herausgezogen). Die Suche erfordert einige Millisekunden Zeit (typischerweise 50 ... 250 ms.).

Die DLL liest kontinuierlich die Messdaten aller erkannten lokalen Sensoren mit der höchst-möglichen Geschwindigkeit aus und speichert sie zur Abfrage durch eine Applikation zwischen. Es steht immer der zuletzt gelesene Datensatz zur Verfügung, alte Messdaten werden überschrieben. Die mögliche Abfragerate ist dabei sehr vom verwendeten PC-System und den angeschlossenen Sensoren abhängig. Üblicherweise werden Abfrageraten zwischen 20 und über 200 Hz erreicht, letztere jedoch nur bei Sentax kompatiblen Sensoren.

Die Messwerte der lokalen Sensoren stehen für die Abfrage durch eine Applikations-Software zur Verfügung. Die zuletzt erfassten Messwerte können beliebig oft und schnell über das Applikations-Interface ausgelesen werden. Die Benutzung des DLL-Interfaces durch die Applikation läuft komplett asynchron zur Erfassung der Messwerte von den Sensoren durch die DLL.

Ablauf der Gerätesuche

Beim Start der DLL sowie bei jedem Gerätewechsel Ereignis führt die DLL eine Suche nach lokalen Sensorgeräten durch. Dazu wird jeder verfügbare lokale VCP kurzzeitig geöffnet und es werden Kommandos zur Identifikation der Sensoren gesendet. Antwortet ein Sensor, so wird die Schnittstelle von der DLL ständig belegt und zur Abfrage der Messwerte verwendet. Antwortet kein Gerät, oder entspricht die Antwort des Gerätes nicht einem Identifikationsdatensatz für Omni Sensoren, dann wird die Schnittstelle wieder freigegeben und kann danach anderweitig verwendet werden. Die Abfrage aller COM-Ports erfolgt dabei parallel und somit fast gleichzeitig.

Diese Vorgehensweise ermöglicht eine komfortable Sensorverwaltung ohne die Notwendigkeit von zusätzlichen Angaben durch den Benutzer, wie z.B. eine Angabe des verwendeten COM-Ports. Letzteres kann gerade bei den VCPs sehr schwierig und undurchsichtig sein, da für jedes VCP Gerät

©2022 by Omni Elektronik GmbH,

Ringstr. 12, 51766 Engelskirchen, Tel. +49 2263 90 29 51 0, Internet www.omni-sensors.de
Technische Änderungen vorbehalten.

und jeden USB-Port unter Umständen eine eigene COM-Port Nummer vergeben wird, und diese leicht 100 und mehr erreichen können. Der Anwender verliert dadurch schnell den Überblick sodass eine automatische Erkennung immer eine große Erleichterung darstellt.

Es gibt jedoch einige Situationen, in denen diese Art der automatischen Gerätesuche Probleme verursachen kann, da die Kommunikation über VCPs ja nicht nur den Omni Sensoren vorbehalten ist. Fremde Geräte, bei denen die serielle Kommunikation sehr rudimentär und primitiv implementiert ist, fühlen sich durch die ausgegebenen Daten möglicherweise angesprochen und reagieren irgendwie, was zu Fehlern in fremder Soft- und Hardware führen kann. Diese möglichen Verhaltensweisen der fremden Hardware müssen beim Einsatz der Sentax Access DLL unbedingt berücksichtigt werden. Weiterhin werden alle Geräte, deren VCP während der Suche anderweitig verwendet wird, nicht erkannt.

6.2 Das Applikations-Programmier-Interface (API) der DLL

Die Sentax Variante der Access DLL integriert das alte Interface zur Erhaltung der Kompatibilität, sowie einige neue Funktionen zur erweiterten Abfrage von Geräte-Eigenschaften. Einige alte Funktionen werden nicht mehr unterstützt oder sind bedeutungslos.

6.3 Liste der exportierten Funktionen (Legacy):

Variante	Funktion	Ordinal Nummer (veraltet)	Kapitelreferenz
UTF-8	SensFindDevice	@1	6.6
	SensReadValues	@2	6.7
	SensSetHeating	@3	6.8
	SensGetChangeFlag	@4	6.9
	SensWaitReady	@5	6.10
	SetQueryInterval	@6	6.11
UTF-8	SensFindDeviceA	@11	6.6
	SensReadValuesA	@12	6.7
	SensSetHeatingA	@13	6.8
	SensGetChangeFlagA	@14	6.9
	SensWaitReadyA	@15	6.10
	SetQueryIntervalA	@16	6.11

©2022 by Omni Elektronik GmbH,

Ringstr. 12, 51766 Engelskirchen, Tel. +49 2263 90 29 51 0, Internet www.omni-sensors.de
Technische Änderungen vorbehalten.

Unicode	SensFindDeviceW	@21	6.6
	SensReadValuesW	@22	6.7
	SensSetHeatingW	@23	6.8
	SensGetChangeFlagW	@24	6.9
	SensWaitReadyW	@25	6.10
	SetQueryIntervalW	@26	6.11
Neutral	SensQueryBroadcast	@30 (nicht mehr unterstützt)	6.12
	SensSendBroadcast	@31 (nicht mehr unterstützt)	6.13
	SetRemoteComputerEnable	@32 (nicht mehr unterstützt)	6.14
	GetRemoteComputerEnable	@33 (nicht mehr unterstützt)	6.15
UTF-8	DllGetVersion	@99	6.16

6.4 Liste der exportierten Funktionen (neu):

Variante	Funktion	Ordinal Nummer (veraltet)	Kapitelreferenz
UTF-8	getDeviceA	@100	6.17
	getTaskA	@101	6.18
Unicode	getDeviceW	@200	6.17
	getTaskW	@201	6.18

6.5 Fehlercodes

Funktionen, die einen detaillierten Fehlercode liefern, verwenden einen der Folgenden Werte:

Bezeichnung	Wert	Beschreibung
SENS_HEATING_ENABLED	1	Die Funktion wurde erfolgreich ausgeführt, es trat kein Fehler auf. Das integrierte Heizelement ist aktiviert, was zu einer signifikanten Abweichung der Messwerte führen kann.
SENS_SUCCESS	0	Die Funktion wurde erfolgreich ausgeführt, es trat kein Fehler auf.

©2022 by Omni Elektronik GmbH,

Ringstr. 12, 51766 Engelskirchen, Tel. +49 2263 90 29 51 0, Internet www.omni-sensors.de
Technische Änderungen vorbehalten.

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 fest-gestellt 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 zurzeit 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 Omni 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.

©2022 by Omni Elektronik GmbH,

Ringstr. 12, 51766 Engelskirchen, Tel. +49 2263 90 29 51 0, Internet www.omni-sensors.de
Technische Änderungen vorbehalten.

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.

6.6 Funktion „SensFindDevice“

Die Funktion durchsucht die aktuelle Geräteliste nach dem *n-ten* Gerät und füllt den Parameter-Puffer 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 aufgerufen werden, solange SENS_SUCCESS geliefert wird.

Prototypen

```
LRESULT CALLBACK SensFindDeviceA(
    LONG          n,
    LPSTR         pszMask,
    PSENSDEVICE   pDevice
);

LRESULT CALLBACK SensFindDeviceW(
    LONG          n,
    LPWSTR        pwszMask,
    PSENSDEVICE   pDevice
);
```

Parameter

Parameter	Typ	Beschreibung
n	LONG	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	LPSTR (ASCII) LPWSTR (Unicode)	Adresse eines Filterstrings oder NULL. Wird eine Stringadresse angegeben, werden nur Geräte berücksichtigt, in deren Typ-Bezeichnung dieser String an beliebiger Stelle enthalten ist. Bei NULL wird dieser Parameter nicht berücksichtigt.
pDevice	PSENSDEVICE PSENSDEVICEA PSENSDEVICEW	Zeiger auf einen Parameter-Puffer mit der Struktur „SENSDEVICE“ oder NULL. Bei Angabe einer Pufferadresse, wird der Puffer bei erfolgreicher Suche mit den Daten des

©2022 by Omni Elektronik GmbH,

Ringstr. 12, 51766 Engelskirchen, Tel. +49 2263 90 29 51 0, Internet www.omni-sensors.de
Technische Änderungen vorbehalten.

		gesuchten Gerätes gefüllt. Bei NULL wird dieser Parameter nicht berücksichtigt.

Rückgabewert

Typ	Rückgabewert	Beschreibung
LRESULT	SENS_SUCCESS	das gewünschte Gerät wurde gefunden
	SENS_FAILED	das gewünschte Gerät wurde nicht gefunden

Definition des Parameter-Puffers

```
typedef struct _SENSDEVICEA   // Bus-Verzeichnis Eintrag, ASCII Code
Version
{
     TCHAR szTypeName[32];    // Gerätetype-Bezeichnung
     TCHAR szSerialNo[32];    // Seriennummer des Gerätes
     LONG  nIndex;            // Geräteindex
} SENSDEVICEA, * PSENSDEVICEA;

typedef struct _SENSDEVICEW   // Bus-Verzeichnis Eintrag, Unicode Version
{
     WCHAR szTypeName[32];    // Gerätetype-Bezeichnung
     WCHAR szSerialNo[32];    // Seriennummer des Gerätes
     LONG  nIndex;            // Geräteindex
} SENSDEVICEW, * PSENSDEVICEW;
```

Hinweis

Diese Funktion liefert alle wichtigen Informationen über die derzeit angeschlossenen Geräte. Durch multiplen Aufruf kann z.B. eine Listbox 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.

6.7 Funktion „SensReadValues“

Funktion zur Abfrage aktueller Messwerte eines Gerätes. Es werden abhängig vom abgefragten Gerät bis zu 3 Messwerte geliefert, z.B. beim OHT20-AT Sensor relative Feuchte, Temperatur und Taupunkttemperatur.

Prototyp

```
LRESULT CALLBACK SensReadValuesA(
    PVOID    Parameter,
```


©2022 by Omni Elektronik GmbH,

Ringstr. 12, 51766 Engelskirchen, Tel. +49 2263 90 29 51 0, Internet www.omni-sensors.de
Technische Änderungen vorbehalten.

```
    BOOL    bMode,
    float * pfValRH,
    float * pfValTemp,
    float * pfValDew
);

LRESULT CALLBACK SensReadValuesW(
    PVOID   Parameter,
    BOOL    bMode,
    float * pfValRH,
    float * pfValTemp,
    float * pfValDew
);
```

Parameter

Parameter	Typ:	Beschreibung
Parameter	PVOID	Parameter ist abhängig von „bMode“ wie folgt. SENS_READ_BY_SERIAL_NUMBER: „Parameter“ ist Zeiger auf einen nullterminierten String (ASCII oder Unicode), der die vollständige Seriennummer des Sensors enthält. SENS_READ_BY_INDEX: „Parameter“ ist der Geräteindex n, der bei der Suchfunktion SensFindDevice(n, ...) benutzt wurde, um das Gerät zu identifizieren.
bMode	BOOL	Adressierungsmodus für Sensorauswahl. Wie zuvor beschrieben, kann der Sensor entweder über die Seriennummer oder den Geräteindex identifiziert werden.
pfValRH	float *	Zeiger auf eine Variable vom Typ float, für den Messwert der relativen Feuchte (4-Byte Fließkomma) oder NULL Bei NULL entfällt die Zuweisung dieses Wertes. Wertebereich: 0.0 bis 100.0 % Wird auf einen Sensor zugegriffen, der keine Feuchte misst, wird 0.0% geliefert.
pfValTemp	float *	Zeiger auf eine Variable vom Typ float, für den aktuellen Temperaturmesswert (4-Byte Fließkomma) oder NULL Bei NULL entfällt die Zuweisung dieses Wertes. Wertebereich: -40.0 bis +120.0 °C Wird auf einen Sensor zugegriffen, der keine Temperatur misst, dann wird -40.0°C geliefert.

©2022 by Omni Elektronik GmbH,

Ringstr. 12, 51766 Engelskirchen, Tel. +49 2263 90 29 51 0, Internet www.omni-sensors.de
Technische Änderungen vorbehalten.

pfValDew	float *	Zeiger auf eine Variable vom Typ float für die berechnete Taupunkttemperatur (4-Byte Fließkomma) oder NULL Bei NULL entfällt die Zuweisung dieses Wertes. Wertebereich: -40.0 bis +120.0 °C Kann der Wert nicht berechnet werden, so wird -40.0°C geliefert. 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.

Rückgabewert

Typ	Rückgabewert	Beschreibung
LRESULT	xxx	Fehlercode vom Typ „SENS_xxx“ (*siehe: Fehlercodes*)

Adressierungsmodi

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 n, der 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.

Hinweis

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.

©2022 by Omni Elektronik GmbH,

Ringstr. 12, 51766 Engelskirchen, Tel. +49 2263 90 29 51 0, Internet www.omni-sensors.de
Technische Änderungen vorbehalten.

6.8 Funktion „SensSetHeating“

Funktion zur Aktivierung oder Deaktivierung des Heizelementes eines OHT20 Sensorgerätes, welches diese Funktion unterstützt.

Prototyp

```
LRESULT CALLBACK SensSetHeatingA(
    PVOID  Parameter,
    BOOL   bMode,
    BOOL   bEnable
);

LRESULT CALLBACK SensSetHeatingW(
    PVOID  Parameter,
    BOOL   bMode,
    BOOL   bEnable
);
```

Parameter

Parameter	Typ:	Beschreibung
Parameter	PVOID	Parameter ist abhängig von „bMode“ wie folgt. SENS_READ_BY_SERIAL_NUMBER: „Parameter“ ist Zeiger auf einen nullterminierten String (ASCII oder Unicode), der die vollständige Seriennummer des Sensors enthält. SENS_READ_BY_INDEX: „Parameter“ ist der Geräteindex n, der bei der Suchfunktion SensFindDevice(n, ...) benutzt wurde, um das Gerät zu identifizieren.
bMode	BOOL	Adressierungsmodus für Sensorauswahl. Wie zuvor beschrieben, kann der Sensor entweder über die Seriennummer oder den Geräteindex identifiziert werden.
bEnable	BOOL	Flag Heizelement aktivieren: TRUE Heizelement deaktivieren: FALSE

Rückgabewert

Typ	Rückgabewert	Beschreibung

©2022 by Omni Elektronik GmbH,

Ringstr. 12, 51766 Engelskirchen, Tel. +49 2263 90 29 51 0, Internet www.omni-sensors.de
Technische Änderungen vorbehalten.

LRESULT	xxx	Fehlercode vom Typ „SENS_xxx“ (*siehe: Fehlercodes*)

Adressierungsmodi

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 n, der 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.

Hinweis

Nur OHT20 Sensorgeräte ab Firmware Version 2.0.00 verfügen über die Fähigkeit, das Heizelement zu aktivieren. Wird die Funktion auf ein Gerät angewendet, welches nicht über ein Heizelement verfügt, dann wird der Status „SENS_INVALID_FUNCTION“ geliefert. Wurde die Heizung erfolgreich eingeschaltet, dann liefert die Messwertabfragefunktion „SensFindDevice()“ statt „SENS_SUCCESS“ den Status „SENS_HEATING_ENABLED“. Die Messwerte sind dann zwar korrekt erfasst worden, jedoch beeinflusst die Heizung die Messwerte erheblich. Bereits eine Temperaturerhöhung um ein Grad Celsius kann eine Abweichung von mehreren Prozent bei der relativen Feuchte verursachen.

6.9 Funktion „SensGetChangeFlag“

Die Funktion testet, ob sich die Sensorkonfiguration seit dem letzten Aufruf verändert hat (neue Geräte oder Geräte entfernt) und liefert TRUE (geändert) oder FALSE (keine Änderung) zurück.

Prototyp

```
BOOL CALLBACK SensGetChangeFlagA(VOID);
BOOL CALLBACK SensGetChangeFlagW(VOID);
```

Parameter

©2022 by Omni Elektronik GmbH,

Ringstr. 12, 51766 Engelskirchen, Tel. +49 2263 90 29 51 0, Internet www.omni-sensors.de
Technische Änderungen vorbehalten.

Parameter	Typ:	Beschreibung
---	VOID	Die Funktion hat keine Parameter.

Rückgabewert

Typ	Rückgabewert	Beschreibung
BOOL	TRUE	Gerätekonfiguration verändert seit letztem Aufruf
BOOL	FALSE	Gerätekonfiguration unverändert

Hinweis

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.

6.10 Funktion „SensWaitReady“

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

Prototyp

```
BOOL CALLBACK SensWaitReadyA(LONG nTimeout);
BOOL CALLBACK SensWaitReadyW(LONG nTimeout);
```

Parameter

Parameter	Typ:	Beschreibung
---	VOID	Die Funktion hat keine Parameter.

Rückgabewert

©2022 by Omni Elektronik GmbH,

Ringstr. 12, 51766 Engelskirchen, Tel. +49 2263 90 29 51 0, Internet www.omni-sensors.de
Technische Änderungen vorbehalten.

Typ	Rückgabewert	Beschreibung
BOOL	TRUE	Suche nach Sensorgeräten wurde innerhalb der angegebenen Zeit beendet oder läuft nicht mehr
BOOL	FALSE	Gerätesuche konnte im angegebenen Zeitraum noch nicht abgeschlossen werden, oder ein anderer Fehler trat auf

Hinweis

Aufgrund der vollständig als multithreaded ausgelegten Implementierung der DLL Funktionen, kann die Anwendung bereits die Geräteliste abfragen, während die Gerätesuche im Hintergrund noch läuft. Die DLL blockiert die Ausführung der Applikation dabei nicht.

Dies kann bei einer Abfrage der Sensorliste direkt dem Start der Applikation problematisch sein, da zu diesem Zeitpunkt meist noch keine Sensorgeräte erkannt wurden. 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 werden alle Sensoren nach ca. 150 bis 500 Millisekunden erkannt.

6.11 Funktion „SetQueryInterval“

Die Funktion legt das Abfrageintervall der lokal angeschlossenen Sensoren in Sekunden (mit Auflösung in Millisekunden) fest. Es wird ausschließlich die Abfrage-Intervallzeit für den Lokalen Server eingestellt. Remote-Sensoren werden nicht beeinflusst. Die Zeiten für die Remote-Sensoren müssen jeweils auf den lokalen Rechnern eingestellt werden.

Prototyp

```
LRESULT CALLBACK SetQueryInterval(
    PVOID  Parameter,
    BOOL   bMode,
    FLOAT  fSeconds
);

LRESULT CALLBACK SetQueryIntervalA(
    PVOID  Parameter,
    BOOL   bMode,
    FLOAT  fSeconds
);
```


©2022 by Omni Elektronik GmbH,

Ringstr. 12, 51766 Engelskirchen, Tel. +49 2263 90 29 51 0, Internet www.omni-sensors.de
Technische Änderungen vorbehalten.

```
LRESULT CALLBACK SetQueryIntervalW(
    PVOID  Parameter,
    BOOL   bMode,
    FLOAT  fSeconds
);
```

Parameter

Parameter	Typ:	Beschreibung
Parameter	PVOID	Parameter ist abhängig von „bMode“ wie folgt. SENS_READ_BY_SERIAL_NUMBER: „Parameter“ ist Zeiger auf einen nullterminierten String (ASCII oder Unicode), der die vollständige Seriennummer des Sensors enthält. SENS_READ_BY_INDEX: „Parameter“ ist der Geräteindex n, der bei der Suchfunktion SensFindDevice(n, ...) benutzt wurde, um das Gerät zu identifizieren.
bMode	BOOL	Adressierungsmodus für Sensorauswahl. Wie zuvor beschrieben, kann der Sensor entweder über die Seriennummer oder den Geräteindex identifiziert werden.
fSeconds	float	Einzustellende Intervallzeit in Sekunden (mit Millisekunden-Auflösung).

Rückgabewert

Typ	Rückgabewert	Beschreibung
LRESULT	xxx	Fehlercode vom Typ „SENS_xxx“ (*siehe: Fehlercodes*)

6.12 Funktion „SensQueryBroadcast“

Diese Funktion wird nicht mehr unterstützt!

Die Funktion fragt die zuletzt im Server eingetroffene benutzerdefinierte Rundrufnachricht und den Zeitpunkt des Eintreffens ab.

Prototyp

```
LRESULT CALLBACK SensQueryBroadcast(
    LONG_PTR *  ppBroadcastMsg,
    DWORD *     pdwTimecode
);
```


©2022 by Omni Elektronik GmbH,

Ringstr. 12, 51766 Engelskirchen, Tel. +49 2263 90 29 51 0, Internet www.omni-sensors.de
Technische Änderungen vorbehalten.

Parameter

Parameter	Typ:	Beschreibung
ppBroadcastMsg	LONG_PTR *	Zeiger auf eine 64-Bit Integer-Variable zur Aufnahme des Nachrichten-Codes.
pdwTimecode	DWORD *	Zeiger auf eine 32-Bit Integer Variable zur Aufnahme des Zeit-Codes.

Rückgabewert

Typ	Rückgabewert	Beschreibung
LRESULT	xxx	Fehlercode vom Typ „SENS_xxx“ (*siehe: Fehlercodes*)

6.13 Funktion „SensSendBroadcast“

Diese Funktion wird nicht mehr unterstützt!

Die Funktion sendet eine benutzerdefinierte Rundrufnachricht an alle Server im lokalen Netzwerk.

Prototyp

```
LRESULT CALLBACK SensSendBroadcast(LONG_PTR BroadcastMsg);
```

Parameter

Parameter	Typ:	Beschreibung
BroadcastMsg	LONG_PTR	Benutzerdefinierter 64-Bit Integer-Wert.

Rückgabewert

Typ	Rückgabewert	Beschreibung
LRESULT	xxx	Fehlercode vom Typ „SENS_xxx“ (*siehe: Fehlercodes*)

6.14 Funktion „SetRemoteComputerEnable“

Diese Funktion wird nicht mehr unterstützt!

Funktion bestimmt den Modus für die Remote-Computer-Suche im Lokalen Netzwerk. Wird als Wert 0 angegeben, so wird die Remote-Computer-Suche deaktiviert. Es sind dann nur noch die lokalen Sensoren sichtbar. Ein anderer Wert reaktiviert die Remote-Computer-Suche.

©2022 by Omni Elektronik GmbH,

Ringstr. 12, 51766 Engelskirchen, Tel. +49 2263 90 29 51 0, Internet www.omni-sensors.de
Technische Änderungen vorbehalten.

Prototyp

```
LRESULT CALLBACK SetRemoteComputerEnable(BOOL bEnable);
```

Parameter

Parameter	Typ:	Beschreibung
bEnable	BOOL	Flag TRUE : Aktivierung der Remote-Computer-Suche FALSE : Deaktivierung

Rückgabewert

Typ	Rückgabewert	Beschreibung
LRESULT	xxx	Fehlercode vom Typ „SENS_xxx“ (*siehe: Fehlercodes*)

6.15 Funktion „GetRemoteComputerEnable“

Diese Funktion wird nicht mehr unterstützt!

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

Prototyp

```
BOOL CALLBACK GetRemoteComputerEnable(VOID);;
```

Parameter

Parameter	Typ:	Beschreibung
---	VOID	Die Funktion hat keine Parameter.

Rückgabewert

Typ	Rückgabewert	Beschreibung
BOOL	TRUE	aktiviert
BOOL	FALSE	deaktiviert

©2022 by Omni Elektronik GmbH,

Ringstr. 12, 51766 Engelskirchen, Tel. +49 2263 90 29 51 0, Internet www.omni-sensors.de
Technische Änderungen vorbehalten.

6.16 Funktion „DllGetVersion“

Die Funktion füllt eine DLLVERSIONINFO Struktur mit den Versionsdaten der DLL, entsprechend den Empfehlungen von Microsoft für Windows DLL's.

Prototyp

```
HRESULT WINAPI DllGetVersion(DLLVERSIONINFO * pDllVerInfo);
```

Parameter

Parameter	Typ:	Beschreibung
pDllVerInfo	DLLVERSIONINFO *	Zeiger auf einen Datenpuffer mit der Struktur DLLVERSIONINFO (siehe "Shlwapi.h")

Rückgabewert

Typ	Rückgabewert	Beschreibung
HRESULT	S_OK E_FAIL, E_...	Fehlercode siehe "WinError.h"

Der Puffer für die Versionsdaten ist im Headerfile "Shlwapi.h" wie folgt definiert:

```
typedef struct _DLLVERSIONINFO
{
    DWORD  cbSize;
    DWORD  dwMajorVersion;  // Major version
    DWORD  dwMinorVersion;  // Minor version
    DWORD  dwBuildNumber;   // Build number
    DWORD  dwPlatformID;    // DLLVER_PLATFORM_*
} DLLVERSIONINFO;
```

6.17 Funktion „ getDevice()“

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

Prototyp

```
LRESULT WINAPI getDeviceA(
    SensorAdressingMode mode,
    void*               parameter,
    SentaxDeviceA*      sensorDef
);

LRESULT WINAPI getDeviceW(
```


©2022 by Omni Elektronik GmbH,

Ringstr. 12, 51766 Engelskirchen, Tel. +49 2263 90 29 51 0, Internet www.omni-sensors.de
Technische Änderungen vorbehalten.

```
    SensorAdressingMode mode,
    void*               parameter,
    SentaxDeviceW*      sensorDef
);
```

Parameter

Parameter	Typ:	Beschreibung
mode	SensorAdressingMode	Verwendeter Modus für die Addressierung des Sensorgerätes.
parameter	void*	Adressierung des Sensorgerätes, abhängig vom Parameter „mode“. Modus „byIndex“: „parameter“ muss einen Zeiger auf eine Variable vom Typ „uint64_t“ enthalten, welche dann den Index des zu adressierenden Gerätes enthält. Modus „bySerialNumber“: „parameter“ muss einen Zeiger auf einen String mit der Seriennummer des adressierten Gerätes enthalten.
sensorDef	SentaxDevice*	Adresse einer Datenstruktur zur Aufnahme der Parameter des adressierten Sensorgerätes.

Adressierungsmodi

Zur Adressierung des Sensors stehen 2 Modi zur Verfügung, die wie folgt definiert sind:

```
enum class SensorAdressingMode {
    byIndex        = 0,
    bySerialNumber    = 1,
};
```

Rückgabewert

Typ	Rückgabewert	Beschreibung
HRESULT	S_OK E_FAIL, E_...	Fehlercode siehe "WinError.h"

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

```
class SentaxDeviceW {
```


©2022 by Omni Elektronik GmbH,

Ringstr. 12, 51766 Engelskirchen, Tel. +49 2263 90 29 51 0, Internet www.omni-sensors.de
Technische Änderungen vorbehalten.

```
    public:
       wchar_t   deviceSerialNumber[22];
       wchar_t   sensorName[32];
       int32_t   countOfTasks;
       int32_t   deviceIndex;
};
```

Bei den Sentax Sensoren werden die Messwerte Temperatur, Feuchte, Taupunkt und Absolute Feuchte einzeln in separaten Tasks übertragen. Die Sentax Sensoren unterstützen jeweils 1 bis 4 Tasks. Jeder Task steht für eine unabhängige Messung. Es gibt keine Abfrage mehr für alle Messwerte gleichzeitig.

Der Parameter „countOfTasks“ besagt, wie viele Tasks vom Sensor unterstützt werden. Der Paramerter „deviceIndex“ kann zur Adressierung des Sensors über den Index verwendet werden. Die Tasks sind abhängig vom verwendeten Sensorgerät.

6.18 Funktion „ getTask()“

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

Prototyp

```
LRESULT WINAPI getTaskA(
    LPCSTR       deviceSerialNumber,
    uint64_t     taskIndex,
    SentaxTaskA*      sensorTask
);

LRESULT WINAPI getTaskW(
    LPCWSTR      deviceSerialNumber,
    uint64_t     taskIndex,
    SentaxTaskW*      sensorTask
);
```

Parameter

Parameter	Typ:	Beschreibung
deviceSerialNumber	Nullterminierter String	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.

Rückgabewert

©2022 by Omni Elektronik GmbH,

Ringstr. 12, 51766 Engelskirchen, Tel. +49 2263 90 29 51 0, Internet www.omni-sensors.de
Technische Änderungen vorbehalten.

Embedded DLL

Typ	Rückgabewert	Beschreibung
HRESULT	S_OK E_FAIL, E_...	Fehlercode siehe "WinError.h"

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

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

Parameter	Typ:	Beschreibung
name	wchar_t	Name des Tasks (also der Messstelle), z.B. „Feuchte“
unit	wchar_t	Verwendete Einheit des Messwerts
pointer	float	letzter Messwert in der angegebenen Einheit
min	float	Messbereich der aktuell eingestellten Skala, untere Bereichsgrenze
max	float	Messbereich der aktuell eingestellten Skala, obere Bereichsgrenze
status	int32_t	Sensorstatus entsprechend der alten Status Codes

Hinweis

Es ist möglich, dass die Einheit **„°C“** in der Unicode-Variante als **„À°C“** angezeigt wird. Wenn dies der Fall ist, dann wurde im Sensor-EEPROM ein Grad-Zeichen mit dem ASCII-Code verwendet, der bei einem Multi-Byte String als Escape-Zeichen verwendet wird. In diesem Fall kann es bei der Konvertierung nach Unicode zu einer Fehlinterpretation kommen, die zu der beschriebenen Anzeige führt. Sie können das Problem beheben, indem Sie mit der Sentax Applikation die Parametrierung des betreffenden Sensors öffnen und als Einheit wieder **„°C“** auswählen. Die Sentax-Anwendung konvertiert die Unicode-Eingabe dann wieder in einen kompatiblen Multi-Byte String und die Einheit wird dann dauerhaft korrekt angezeigt.

©2022 by Omni Elektronik GmbH,

Ringstr. 12, 51766 Engelskirchen, Tel. +49 2263 90 29 51 0, Internet www.omni-sensors.de
Technische Änderungen vorbehalten.

7 Änderungsnachweis

Revision	Änderungen
1.0	---

©2022 by Omni Elektronik GmbH,

Ringstr. 12, 51766 Engelskirchen, Tel. +49 2263 90 29 51 0, Internet www.omni-sensors.de
Technische Änderungen vorbehalten.