

Systemvoraussetzungen:

Die DLL wurde nur für die Verwendung in 32-Bit Windows Betriebssystemen erstellt, also für Windows XP, Windows 2000 oder Windows 2003 usw.. DOS basierende Windows Systeme, z.B. Windows 3.11, Windows '95, Windows '98 usw. werden weder von der DLL noch einem passenden Gerätetreiber unterstützt, und sind somit ungeeignet.

Für den Betrieb der unterstützten Sensoren ist neben der Sensor-Hardware ein MELTEC Gerätetreiber „UFT75AT.sys“ nötig, der als USB Treiber für das jeweilige Sensorgerät installiert wurde.

Die Nachfolgende Beschreibung verwendet überwiegend Darstellungen der Programmiersprache „C“. Die DLL kann jedoch mit den meisten anderen Programmiersprachen ebenfalls verwendet werden.

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. Jedoch ist bei dem Sensor das integrierte Heizelement aktiviert, was zu einer signifikanten Abweichung der Messwerte führen kann.
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 zur Zeit 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.

Funktion „SensFindDevice()“:

Prototyp:

LRESULT CALLBACK SensFindDevice(LONG n, LPSTR pszMask, PSENSDEVICE 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 solange aufgerufen werden, wie SENS_SUCCESS geliefert wird.

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	LPSTR	Adresse eines Filterstrings oder NULL. Wenn eine Stringadresse 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:

Typ LRESULT, 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 _SENSDEVICE                               Bus-Verzeichnis Eintrag
{
    BYTE        szTypeName[32];                          Gerätetype-Bezeichnung
    BYTE        szSerialNo[32];                          Seriennummer des Gerätes
    LONG        nIndex;                                  Geräteindex
} SENSDEVICE, * PSENSDEVICE;
```

Hinweise:

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 1000 Geräte unterstützt, der Geräteindex muss deshalb immer zwischen 0 und 999 liegen.

Funktion „ SensReadValues()“:

Prototyp:

LRESULT CALLBACK SensReadValues(PVOID Parameter, BOOL 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 liegen immer zwischen 0 und 100%. Messwerte für Temperaturen zwischen -40 und +120 °C.

Parameter:	Typ:	Beschreibung:
Parameter	PVOID	Parameter ist abhängig von „bMode“. Wird als Modus SENS_READ_BY_SERIAL_NUMBER angegeben, so ist „Parameter“ ein Zeiger auf einen nullterminierten String, 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	BOOL	Adressierungsmodus für Sensorauswahl. Wie zuvor beschrieben, kann der Sensor entweder über die Seriennummer der den Geräteindex identifiziert werden.
pfValRH	float *	Zeiger auf eine Variable vom Typ 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 -40.0 geliefert.
pfValTemp	float *	Zeiger auf eine Variable vom Typ 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 zugewiesen.
pfValDew	float *	Zeiger auf eine Variable vom Typ 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 geliefert.

Return Wert:

Typ **LRESULT**, die Funktion liefert einen Fehlercode vom Typ „SENS_xxx“ als Funktionsergebnis zurück.

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 empfohlen, den Sensor über seine Seriennummer auszuwählen.

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.

Funktion „SensSetHeating()“:

Prototyp:

LRESULT CALLBACK SensSetHeating(PVOID Parameter, BOOL bMode, BOOL bEnable);

Beschreibung:

Funktion zur Aktivierung oder Deaktivierung des Heizelementes eines UFT75 Sensorgerätes, welches diese Funktion unterstützt.

Parameter:	Typ:	Beschreibung:
Parameter	PVOID	Parameter ist abhängig von „bMode“. Wird als Modus SENS_READ_BY_SERIAL_NUMBER angegeben, so ist „Parameter“ ein Zeiger auf einen nullterminierten String, 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	BOOL	Adressierungsmodus für Sensorauswahl. Wie zuvor beschrieben, kann der Sensor entweder über die Seriennummer der den Geräteindex identifiziert werden.
bEnable	BOOL	Flag gibt an, ob das Heizelement aktiviert (TRUE) oder deaktiviert (FALSE) werden soll.

Return Wert:

Typ **LRESULT**, die Funktion liefert einen Fehlercode vom Typ „SENS_xxx“ als Funktionsergebnis zurück.

Mögliche 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, 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 empfohlen, den Sensor über seine Seriennummer auszuwählen.

Hinweise:

Nur UFT75 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 „SensReadValues()“ 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 mehrere Prozent bei der relativen Feuchte verursachen.

Funktion „ SensGetChangeFlag()“:

Prototyp:

BOOL CALLBACK _export SensGetChangeFlag(VOID);

Beschreibung:

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

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

Return Wert:

Typ **BOOL**, die Funktion liefert TRUE, wenn sich die Gerätekonfiguration seit dem letzten Aufruf verändert hat, und FALSE wenn nicht.

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 kann, abhängig von der Anzahl angeschlossener Sensoren, einige Sekunden dauern. Alle Abfragefunktionen funktionieren während dieser Zeit nicht, und warten bis zu 3 Sekunden auf den Abschluss der Gerätesuche. Wird die Gerätesuche nicht innerhalb dieses Zeitraums abgeschlossen, dann liefert die aufgerufene Funktion SENS_FAILED zurück. Gerätetreiber und DLL können theoretisch maximal 1000 Sensorgeräte (was nur durch den Einsatz von 8 oder 9 USB Controllern möglich wäre) gleichzeitig verwalten. Sensoren, auf die während der Gerätesuche von einer anderen Applikation aus zugegriffen wird, werden möglicherweise nicht erkannt.

Beispiel-Code:

Der folgende Code demonstriert in Standard „C“, wie verfügbare UFT75-AT Sensorgeräte gesucht und in eine Listbox eingetragen werden. Mit einem Timer werden die Messdaten des vom Anwender ausgewählten Sensors in regelmäßigen Intervallen gelesen und angezeigt. Der Code entspricht im Wesentlichen der Funktion des Beispielprogramms „UFTAccessTest.exe“ (vereinfacht und Ausschnittsweise dargestellt). Datentypen und DLL-Aufrufe sind rot markiert. Natürlich wird eine Dialog-Ressource mit den verwendeten Feldern benötigt, die hier nicht extra ausgeführt wird.

1. Beispiel zum erstellen einer Listbox Auswahl, welche die verfügbaren UFT75 Sensorgeräte auflistet (vereinfacht):

```
#include ... Windows Includes einfügen
#include "UFTAccess.h"

VOID SetupMyListBox(HWND hWnd)
{
SENSDEVICE DeviceInfo; /* Gerätedatenpuffer */
HWND hItem; /* ListBox Handle */
LONG i; /* Zähler */
LONG Index; /* ListBox Auswahl */

hItem = GetDlgItem(hWnd, /* ermittle ListBox Handle */
IDC_SENSORLIST);

SendMessage(hItem, /* init Liste */
LB_RESETCONTENT, 0, 0L);

ZeroMemory(&DeviceInfo, /* init Puffer */
sizeof(SENSDEVICE));

i = 0;

while(SensFindDevice(i, 0L, &DeviceInfo) == SENS_SUCCESS)
{
sprintf(szBuffer, "%s - %s", /* Eintrag formatieren */
DeviceInfo.szSerialNo, DeviceInfo.szDeviceID);

Index = SendMessage(hItem, /* Element hinzufügen */
LB_ADDSTRING, 0, (LPARAM)szBuffer);

if(Index != LB_ERR) /* wenn kein Fehler auftrat */
{
SendMessage(hItem, /* Element hinzufügen */
LB_SETITEMDATA, (LPARAM)Index, (LPARAM)i);
}

i++; if(i > 999) break; /* nächstes Gerät, max. 1000 */
}

SetDlgItemText(hWnd, IDC_VAL_RH, "---");
SetDlgItemText(hWnd, IDC_VAL_TEMP, "---");
SetDlgItemText(hWnd, IDC_VAL_DEW, "---");

SetDlgItemText(hWnd, IDC_STATUS, "Noch keine Daten gelesen!");
}
```

2. Beispiel (Ausschnitt) aus einer Dialog-Callback Funktion. Die Messwerte des in der Listbox gewählten Sensors werden über Timer (1) ausgelesen und im Dialog angezeigt:

```
switch(Message)                /* je nach Message Code */
{
case WM_TIMER:                  /* Timer-Message */
{
switch(wParam)                 /* je nach Timerkennung */
{
case 0x0001:                   /* neue Messung lesen */
{
LRESULT      Status;          /* Statuscode */
BYTE         szBuffer[256];   /* Arbeitspuffer */
HWND         hWnd;           /* Item Handle */
LONG         Index;          /* Listenauswahl */
LPSTR        p;              /* Hilfspointer */
float        fValRH;         /* Feuchtemessung */
float        fValTemp;       /* Temperaturmessung */
float        fValDew;        /* Taupunktberechnung */

hItem = GetDlgItem(hWnd,      /* ermittle ListBox Handle */
                    IDC_SENSORLIST);

Index = SendMessage(hItem,   /* ermittle Auswahl */
                    LB_GETCURSEL, 0, 0L);

if(Index == LB_ERR) return(FALSE); /* wenn kein Sensor gewählt */

ZeroMemory(szBuffer, 256); /* init Puffer */

SendMessage(hItem,          /* lese Eintrag aus Liste */
            LB_GETTEXT, (WPARAM)Index, (LPARAM)szBuffer);

p = strstr(szBuffer, " - "); /* Seriennummer abtrennen */
if(p) *p = 0;

Status = SensReadValues((PVOID)szBuffer,
                        SENS_READ_BY_SERIAL_NUMBER,
                        &fValRH, &fValTemp, &fValDew);

if(Status >= SENS_SUCCESS) /* wenn kein Fehler auftrat */
{
CheckDlgButton(hWnd, IDC_HEATING,
                (Status == SENS_HEATING_ENABLED) ? TRUE : FALSE);

sprintf(szBuffer, "%5.1f %%RH", fValRH);
SetDlgItemText(hWnd, IDC_VAL_RH, szBuffer);

sprintf(szBuffer, "%+5.1f °C", fValDew);
SetDlgItemText(hWnd, IDC_VAL_DEW, szBuffer);

sprintf(szBuffer, "%+5.1f °C", fValTemp); /* Temperatur anzeigen */
SetDlgItemText(hWnd, IDC_VAL_TEMP, szBuffer);
}
}
break;

default:
break;
}
}
return(FALSE);
```

... andere Messages auswerten