HYDAC INTERNATIONAL

FluidMonitoring Toolkit

FluMoT

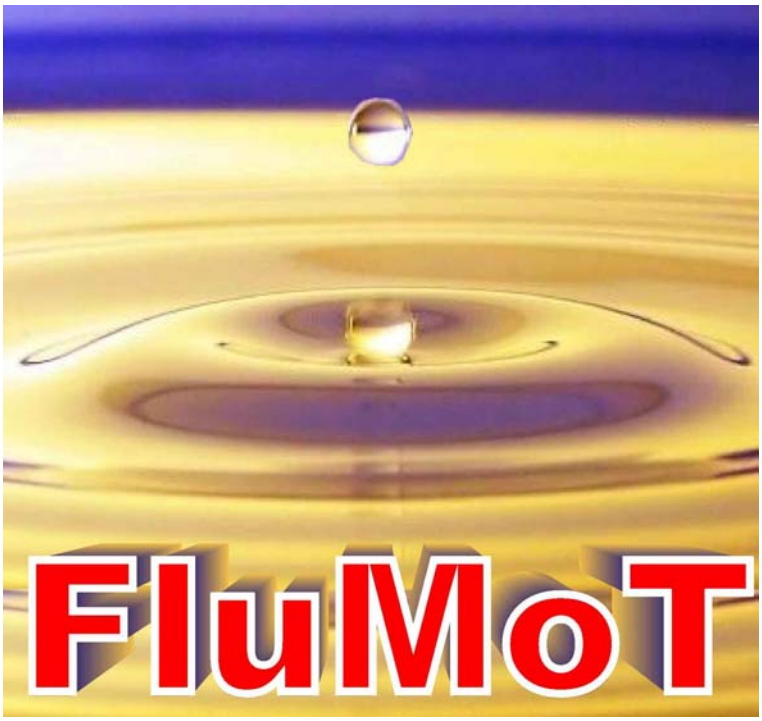
Version 1.1x

Für:

- **CS 1000 series**
- **CS 2000 series**
- **AS 1000 series**
- **HLB 1000 series**
- **FCU 1000 series**
- **FCU 2000 series**
- **FCU 8000 series**
- **CSM 1000 series**
- **CSM 2000 series**
- **FMM series**

Bedienungsanleitung

Doc.: 3377564



Warenzeichen

Die verwendeten Warenzeichen anderer Firmen bezeichnen ausschließlich die Produkte dieser Firmen.

Copyright © 2007 by HYDAC Filtertechnik GmbH Alle Rechte vorbehalten

Haftungsausschluss

Wir haben unser Möglichstes getan, die Richtigkeit des Inhalts dieses Dokuments zu gewährleisten, dennoch können Fehler nicht ausgeschlossen werden. Deshalb übernehmen wir keine Haftung für Fehler und Mängel in diesem Dokument, auch nicht für Folgeschäden, die daraus entstehen können. Die Angaben in dieser Druckschrift werden regelmäßig überprüft, und notwendige Korrekturen sind in den nachfolgenden Auflagen enthalten. Für Anregungen und Verbesserungsvorschläge sind wir dankbar.

Technische Änderungen bleiben vorbehalten.

Inhaltliche Änderungen dieses Handbuchs behalten wir uns ohne Ankündigung vor.

HYDAC Filtertechnik GmbH

Servicetechnik / Filtersysteme

Industriegebiet

D-66280 Sulzbach / Saar

Germany

Tel.: ++49 (0) 6897 / 509 - 01

Fax: ++49 (0) 6897 / 509 - 846

Inhalt

1 Einführung 6
1.1 Allgemeines 6
1.2 Zum Gebrauch dieser Bedienungsanleitung 6
1.3 Symbol- und Hinweiserklärung 6
2 Soft- und Hardware installieren 7
2.1 Systemvorrausetzungen Hardware 7
2.2 Installation vorbereiten 7
2.3 FluMoT installieren 7
2.4 FluMoT deinstallieren 12
3 FluMoT Inhalte 13
3.1 Einführung 13
3.1.1 Datentypen 14
3.2 DIN Messbus - DLL 15
3.2.1 API – Funktionen 15
3.2.2 Fehlerbehandlung 16
3.2.3 Statuskontrolle 17
GetErrorStateText_DMB() 17
3.2.4 Versionskontrolle 17
GetDLLVersion_DMB() 17
GetDLLVersionText_DMB() 17
3.2.5 Gerätesuche und Geräteinformation 17
SearchBusDevice_DMB() 17
GetDeviceSerialNumber_DMB() 17
GetDeviceSensorNumber_DMB() 18
GetDeviceCalibrationDate_DMB() 18
GetDeviceChannelCount_DMB() 18
GetDeviceChannelInfo_DMB() 19
SetBusAddress_DMB() 20
3.2.6 Messwerte lesen 20
SetMeasuringState_DMB() 20
GetDeviceState_DMB() 21
GetDeviceMeasuringValues_DMB() 21
3.3 HSI - DLL 22
3.3.1 API - Funktionen 22
3.3.2 Fehlerbehandlung 23
3.3.3 Statuskontrolle 23
GetErrorStateText_HSI() 23

3.3.4 DLL – Versionskontrolle 23
GetDLLVersion_HSI() 23
GetDLLVersionText_HSI() 24
3.3.5 Gerätesuche und Geräteinformation 24
SearchOneDevice_HSI() 24
SearchBusDevice_HSI() 24
GetDeviceChannelCount_HSI() 24
GetDeviceSerialNumber_HSI() 25
GetDeviceChannelInfo_HSI() 25
Verwalten von Busaddressen 26
GetBusAddress_HSI() 26
SetBusAddress_HSI() 26
3.3.6 Messwerte lesen 27
GetDeviceChannelsMask_HSI() 27
GetDeviceMeasuringValues_HSI() 28
Beispiele zur Messwert Interpretation 29
GetDeviceState_HSI() 30
3.4 Beispiele 31
4 Messkanal Übersicht 32
4.1 FCU 2000 Serie 32
4.2 FCU 8000 Serie 33
4.3 CS 2000 Serie 34

Registrierkarte
FluMoT Version 1.1x

Registrierung

Mit dem Öffnen der Datenträgerverpackung bzw. Installieren der Software haben Sie sich mit den im Software-Überlassungsvertrag aufgeführten Nutzungsbedingungen einverstanden erklärt.

Senden Sie ergänzend diese Registrierkarte ausgefüllt an uns zurück und Sie werden bei uns als Benutzer des Programms registriert.

Nutzen Sie die Vorteile die Ihnen eine Registrierung bietet:

- Kostenloser Support via E-Mail: filtersysteme_support@hydac.com
- News zur Software
- Informationen über Updates der Software

Wir garantieren Ihnen, Ihre Daten nicht an Dritte weiterzugeben.

FluMoT Registrierungs-Schlüssel

Firma

Straße

Postleitzahl, Ort

Name des Benutzers

E-Mail Adresse

Ort, Datum, Unterschrift

Bitte senden Sie die vollständig ausgefüllte Registrierungskarte per Post, Fax oder E-Mail zurück an:

HYDAC Filtertechnik GmbH, Servicetechnik / Filtersysteme,
Industriestraße, Werk 6, D-66280 Sulzbach / Saar,
Fax: ++49 (0) 6897 / 509-846, E-Mail: filtersysteme_support@hydac.com

Ohne Registrierung kann der Support durch HYDAC verweigert werden!

1 Einführung

1.1 Allgemeines

FluMoT (**FluidMonitoring Toolkit**) ist eine Sammlung von unterschiedlichen Entwicklungswerkzeuge, die Kommunikation zwischen PC und HYDAC - Sensoren ermöglichen.

Dieser Toolkit stellt einem Programmierer, der Anwendungen entwickelt, einige Standardfunktionen sowie Schnittstellen zur Verfügung.

Es handelt sich dabei um komplett Lösungen als auch um Schnittstellen, die für eine Softwareentwicklung gedacht sind.

Mit FluMoT können folgende Geräte abgefragt werden:

- ContaminationSensor CS 1000, CS 2000
- FluidControl Unit FCU1000, FCU2000, FCU8000
- HYDACLab HLB 1000
- AquaSensor AS 1000
- ContaminationSensor Modules CSM 1000, CSM 2000
- FluidMonitoring Modules FMM, FMMP, FMMHP, FMMP Unit

Wie die Sensoren bzw. Geräte angeschossen werden, entnehmen Sie bitte der jeweiligen Bedienungsanleitung.

1.2 Zum Gebrauch dieser Bedienungsanleitung

Im folgenden wird vorausgesetzt, dass Sie mit der Bedienung von WINDOWS 98, 2000, ME, XP, dem Aufbau und der Installation Windows typischer Programme vertraut sind!

1.3 Symbol- und Hinweiserklärung

In dieser Bedienungsanleitung werden folgende Benennungen und Zeichen für Hinweise verwendet:

Unter diesem Symbol werden die wichtigen **Informationen** zusammengefasst.

Mit diesem Symbol werden die **Anwendungstips** und besonders nützliche Informationen bezeichnet.

Dieses Symbol gibt wichtige **Hinweise** für den sachgerechten Umgang mit dem Produkt. Das Nichtbeachten dieser Hinweise kann zu Fehlbedienung bzw. Funktionsstörungen führen.

Bei Fragen, Problemen und Anregungen zu **FluMoT** wenden Sie sich bitte an unseren Technischen Vertrieb.

HYDAC FILTERTECHNIK GmbH
Servicetechnik / Filtersysteme
Postfach 12 51
D-66273 Sulzbach / Saar - Deutschland

E-Mail: filtersysteme@hydac.com
Fax.: ++49 (0) 6897 509 - 846

2 Soft- und Hardware installieren

2.1 Systemvorrausetzungen Hardware

- Pentium Prozessor 200 MHz oder höher mit WINDOWS 98, 2000, ME, XP
- 64 MB RAM-Speicher.
- Microsoft Internet Explorer 4.0 oder höher
- VGA-Grafikkarte (800x600 min.)
- Festplatte mit mindestens 30 MB freiem Speicherplatz
- Eine freie serielle Schnittstelle (RS232 / USB):
  1. Nicht mit einem Stecker belegt ist
  2. Nicht vom Betriebssystem benutzt wird
  3. Von keinem anderen Programm benutzt wird
     (wie z. B. Terminal-, Modem- oder Netzwerksoftware)
- Microsoft Windows kompatible Maus
- Administrator Rechte zur Softwareinstallation

2.2 Installation vorbereiten

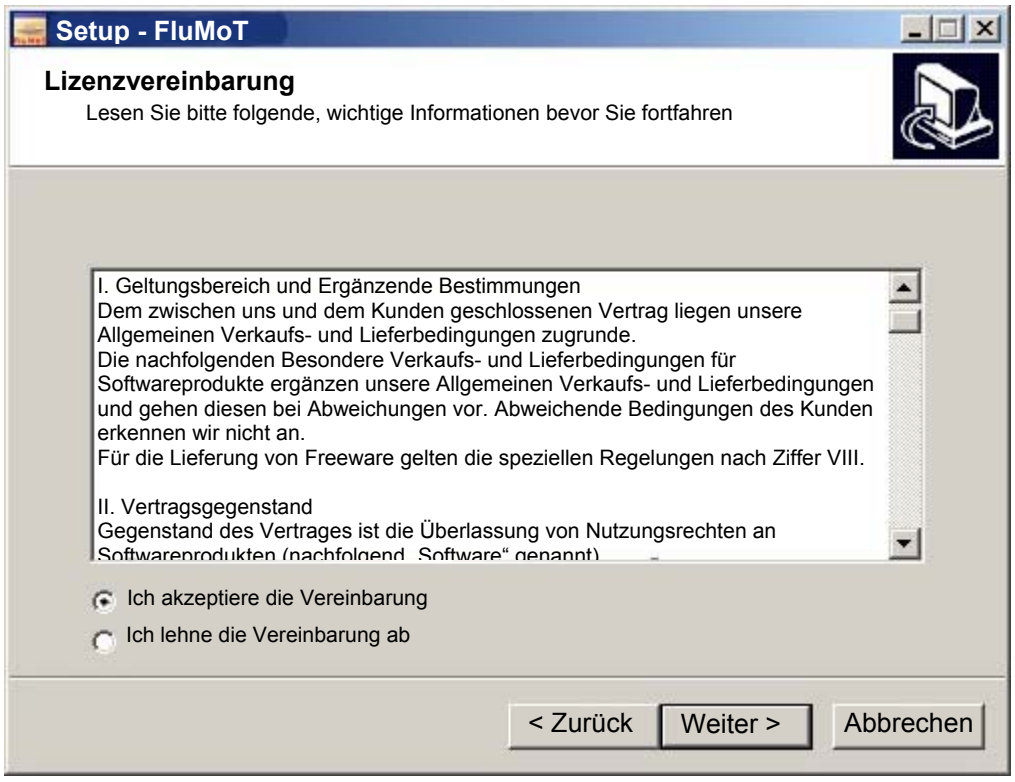
- Um die Funktion zu gewährleisten, empfehlen wir ältere Versionen von **FluMoT** zu deinstallieren.
- Wie die Sensoren bzw. Geräte angeschossen werden, entnehmen Sie bitte der jeweiligen Bedienungsanleitung.

2.3 FluMoT installieren

Zur Installation von FluMoS, starten Sie das Programm SETUP_FLUMOT_Vxxx.EXE auf der CD.

Der Setup-Assistent führt Sie durch die gesamte Installation. Zum Fortfahren drücken Sie „Weiter“.

Um die Installation fortzusetzen müssen Sie die Lizenzvereinbarung in dem nachfolgenden Fenster sorgfältig durchlesen und anschließend akzeptieren.

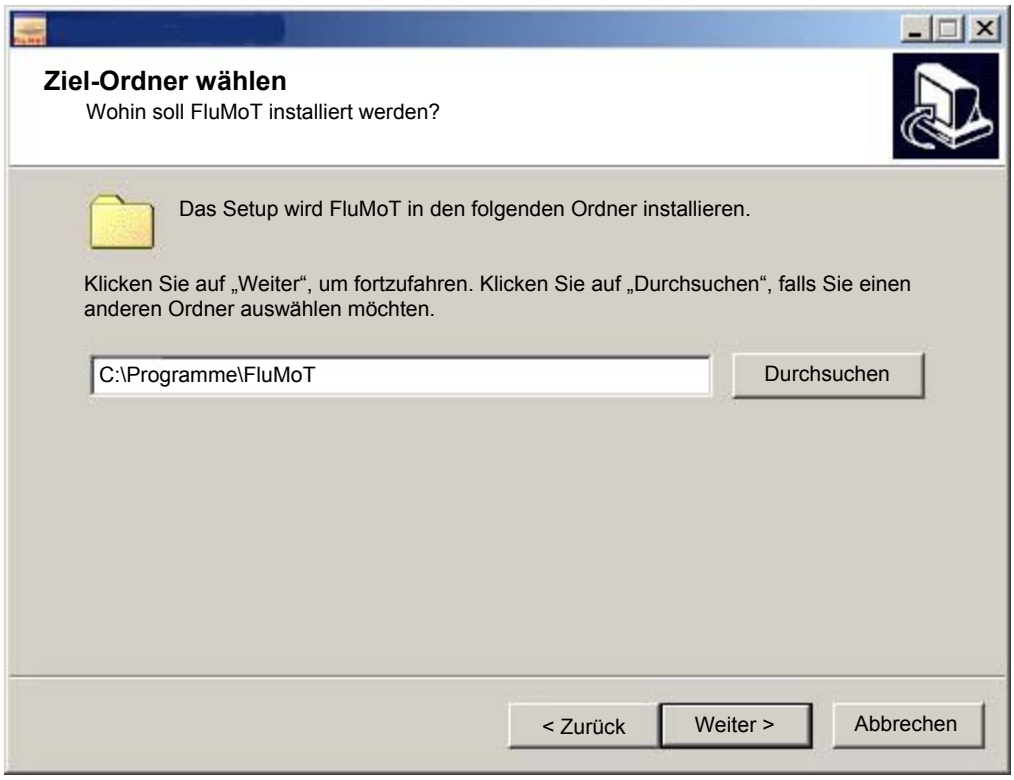


Um Ihre FluMoT Software zu aktivieren, tragen Sie den Registrierungsschlüssel von der FluMoT CD ein.

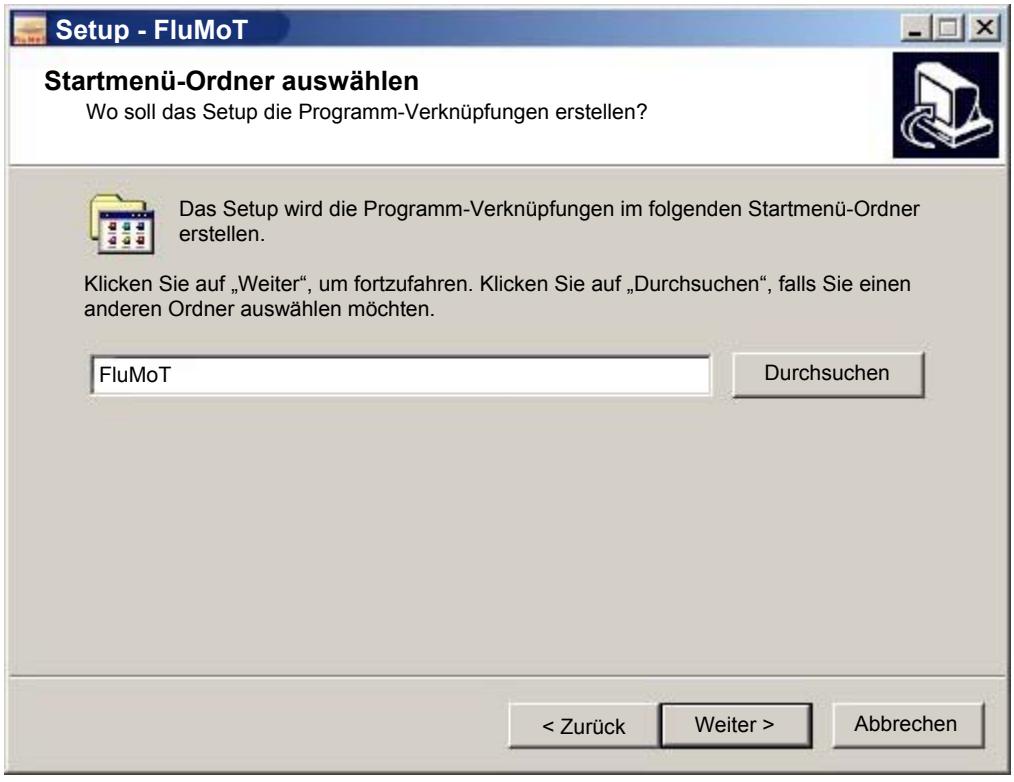
Bei der Installation werden Programmdateien in das Installationsverzeichnis übertragen.

Im nächsten Schritt wird das Installationsverzeichnis festgelegt.

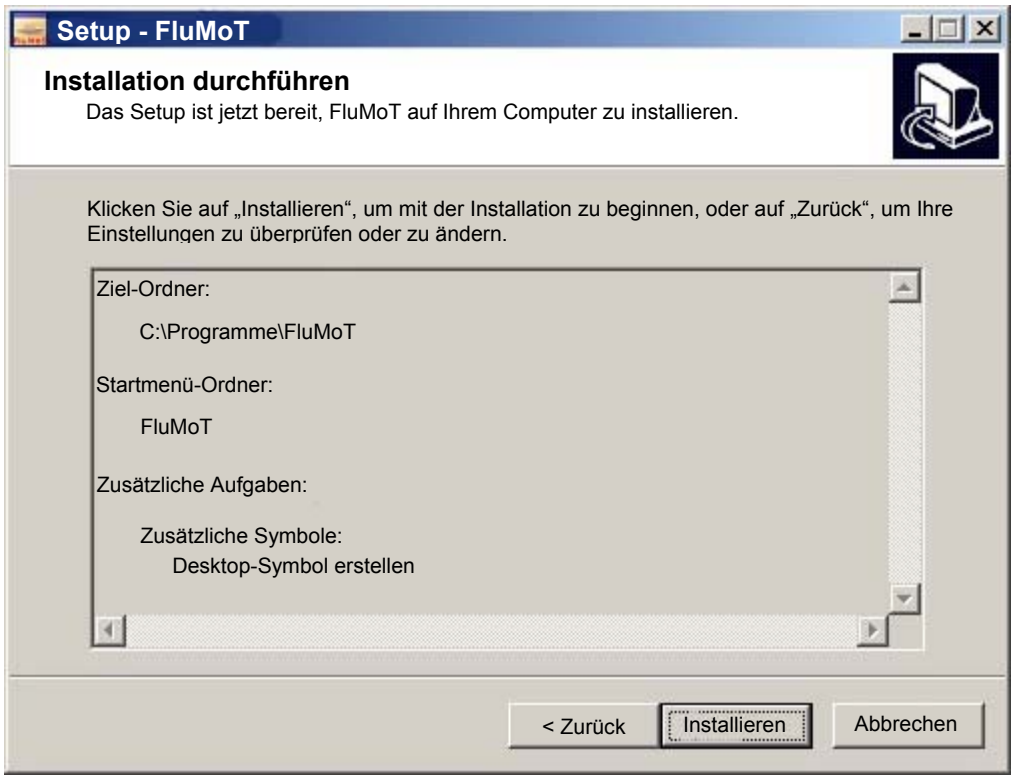
Sollte das Installationsverzeichnis bereits existieren, erfolgt die Frage, ob der Pfad überschrieben werden soll.



Danach wird ein Startmenü-Ordner erstellt.



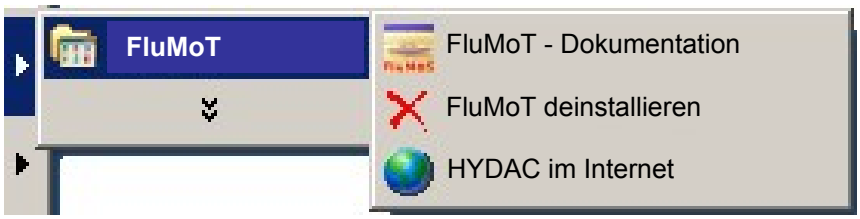
Nach der Bestätigung durch druck auf Weiter > wird der Installationsprozess gestartet.



Der Setup – Assistent wird mit dem „Fertigstellen“ Button geschlossen.

2.4 FluMoT deinstallieren

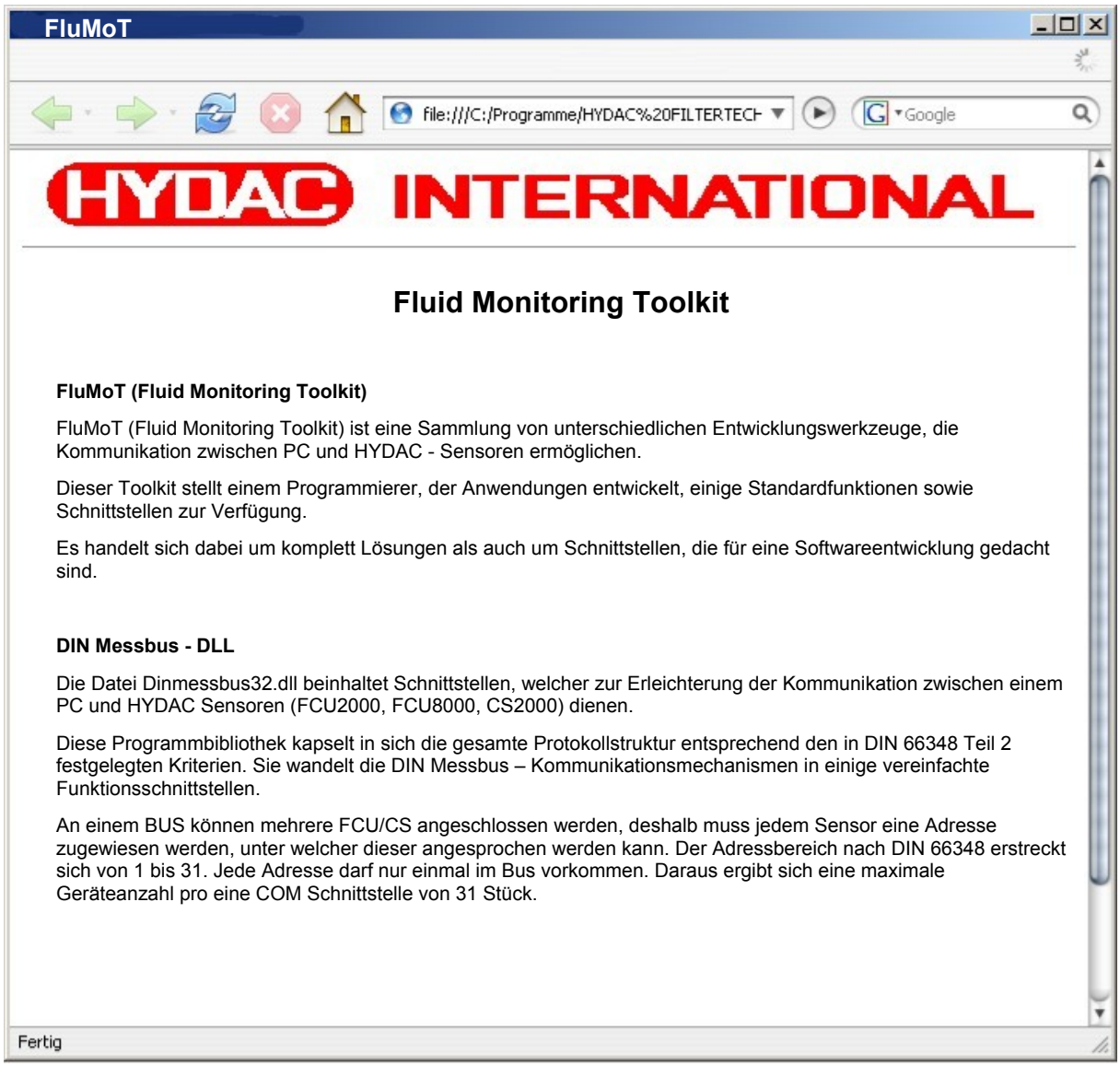
Zur Deinstallation von **FluMoT** führen Sie die im Installationsverzeichnis abgelegte Datei UNINS000.EXE aus oder starten Sie Deinstallation aus dem Startmenü:



3 FluMoT Inhalte

3.1 Einführung

Nach der Installation von FluMoT wird die erste Seite der HTML-Hilfe gestartet.
Diese beinhaltet Beschreibung aller Tools und Schnittstellen.



Ein Bestandteil von FluMoT sind Programmbibliotheken. (DLLs) Sie stellen die Schnittstellen zur Kommunikation mit unterschiedlichen HYDAC – Geräten bereit. Die unterschiedlichen HYDAC - Geräte sind mit verschiedenen Protokollen ansprechbar.

Aus diesem Grund beinhaltet FluMoT zwei DLLs (*dinmessbus32.dll, hecom32.dll)* welche nachfolgend im Detail beschrieben sind.

3.1.1 Datentypen

Um die Hochsprachenprogrammierung zu erleichtern, werden einfache Beispiele als kleine Projekte in Delphi7, LabView 7 und Excel -Makros (VBA 6) im Quellcode mitgeliefert.

In allen Programmbibliotheken von FluMoT werden immer die gleichen Datentypen verwendet. Sie werden in nachfolgender Tabelle aufgelistet.

Typ	*Beschreibung*	*Delphi*	*C/C++*	*VB/VBA*	*Labview*
Integer	Bereich: -2147483648 ... 2147483647 Format: 32 Bit, mit Vorzeichen	Integer	Integer	Long	Long
Double	Bereich: 5.0 x 10^-324 ... 1.7 x 10^308 Format: 8 Byte	Double	Float	Double	Double
String	Repräsentiert einen Zeiger auf einen Char - Wert. Das Ende der Zeichenkette wird durch ein Null – Zeichen festgelegt. Format: 1 Byte pro Zeichen.	Char	Char*	String	String (C String Pointer)

3.2 DIN Messbus - DLL

Die Datei Dinmessbus32.dll beinhaltet Schnittstellen, welcher zur Erleichterung der Kommunikation zwischen einem PC und HYDAC Sensoren (FCU2000, FCU8000 ab Index G und CS2000) dienen.

Diese Programmbibliothek kapselt in sich die gesamte Protokollstruktur entsprechend den in DIN 66348 Teil 2 festgelegten Kriterien. Sie wandelt die DIN Messbus – Kommunikationsmechanismen in einige vereinfachte Funktionsschnittstellen.

An einem BUS können mehrere FCU/CS angeschlossen werden, deshalb muss jedem Sensor eine Adresse zugewiesen werden, unter welcher dieser angesprochen werden kann. Der Adressbereich nach DIN 66348 erstreckt sich von 1 bis 31. Jede Adresse darf nur einmal im Bus vorkommen. Daraus ergibt sich eine maximale Geräteanzahl pro je COM Schnittstelle von 31 Stück.

3.2.1 API – Funktionen

Alle DLL – Funktionen werden in dieser Anleitung mit Hilfe von Pascal – Syntax beschrieben. In folgender Tabelle wird die Übersicht der Funktionen in der *dinmessbus32.dll* dargestellt.

Funktion	***Kurzbeschreibung***
GetDLLVersion_DMB	DLL – Version als Zahl
GetDLLVersionText_DMB	DLL – Version als Text
SearchBusDevice_DMB	SensorID (auch in einem Bussystem) lesen
GetDeviceSerialNumber_DMB	Seriennummer des Gerätes lesen
GetDeviceSensorNumber_DMB	Sensornummer des Gerätes lesen
GetDeviceCalibrationDate_DMB	Datum der letzten Kalibrierung lesen
GetDeviceChannelCount_DMB	Anzahl der Messkanäle lesen
GetDeviceChannelInfo_DMB	Messkanal – Eigenschaften lesen
SetBusAddress_DMB	Busadresse setzen
SetMeasuringState_DMB	Messung starten/stoppen
GetDeviceState_DMB	Gerätestatus ermitteln
GetDeviceMeasuringValues_DMB	Messwerte lesen

3.2.2 Fehlerbehandlung

Fast alle Funktionen liefern im Fehlerfall einen Fehlercode. Dieser Fehlercode kann folgende Werte beinhalten:

Wert	*Beschreibung*	
0	DMB_NO_ERRORCODE	Kein Fehler. Gerät ist betriebsbereit.
10	DMB_TRANSMIT_ERRORCODE	Fehler bei der Übertragung der Daten zum Gerät.
11	DMB_RECEIVE_ERRORCODE	Fehler bei der Datenübertragung vom Gerät.
12	DMB_INVALIDMODE_ERRORCODE	Falsche Mode - Einstellung
13	DMB_INVALIDADDR_ERRORCODE	Falsche Busadresse
14	DMB_INVALIDMODEL_ERRORCODE	Unbekannte Geräteserie
15	DMB_INVALIDCHANNUM_ERRORCODE	Kanalnummer falsch
16	DMB_NODEVICE_ERRORCODE	Kein Gerät gefunden
17	DMB_PROTOCOL_ERRORCODE	DIN Messbus Protokollfehler
18		COM Port - Fehler
20	DMB_CAL_LOST_ERRORCODE	Kalibrierfaktoren falsch
21	DMB_CONST_LOST_ERRORCODE	Konstante Parameter falsch (z.B. Seriennummer)
22	DMB_PARA_LOST_ERRORCODE	Variable Parameter falsch
23	DMB_I2C_ERROR_ERRORCODE	I²C - Busfehler
24	DMB_EE_CHECK_ERRORCODE	Checksumme in EEPROM falsch
25	DMB_WRONG_FORMAT_ERRORCODE	Syntaktischer Fehler im Befehl
26	DMB_COMMAND_ERROR_ERRORCODE	Semantischer Fehler im Befehl
27	DMB_PROT_LOST_ERRORCODE	Log beschädigt
28	DMB_TX_ERROR_ERRORCODE	Fehler im Übertragungsprotokoll
29	DMB_Q_ERROR_ERRORCODE	Durchflussfehler
30	DMB_VDD_ERROR_ERRORCODE	Fehler ±VDD
31	DMB_ISENSOR_ERROR_ERRORCODE	Fehler LED-Strom Partikelsensor
32	DMB_VIN_ERROR_ERRORCODE	Batteriespannung zu gering

3.2.3 Statuskontrolle

GetErrorStateText_DMB()

Mit dieser Funktion kann anhand von Stauscode eine passende Statusmeldung (auf Englisch) ausgegeben werden.

Syntax: **function** GetErrorStateText_DMB (State: **Integer**): **String**;

Parameter: *State* – Kommunikationsstatus.

Rückgabewert: Statusmeldung vom Sensor (Englisch).

Antwort: 16: no device found

3.2.4 Versionskontrolle

GetDLLVersion_DMB()

Mit dieser Funktion kann die Bibliothekversion ermittelt werden.

Syntax: **function** GetDLLVersion_DMB(): **double**;

Rückgabewert: Die Versionsnummer wird als Double – Zahl zurückgeliefert.

Antwort: 1,1

GetDLLVersionText_DMB()

Mit dieser Funktion kann die Bibliothekversion ermittelt werden.

Syntax: **function** GetDLLVersionText_DMB(): **String**;

Rückgabewert: Die Versionsnummer und Ausgabedatum werden als Text zurückgeliefert.

Antwort: v1.01 09.11.2007

3.2.5 Gerätesuche und Geräteinformation

SearchBusDevice_DMB()

Diese Funktion dient zur Suche eines Gerätes an einem bestimmten COM Port mit einer bestimmten Busadresse.

Syntax: **function** SearchBusDevice_DMB (PortNumber: **Integer**; Address: **Integer**; **var** State: **Integer**): **String**;

Parameter: *PortNumber* – Die Nummer des COM Portes.
Address – Busadresse des Gerätes als Integerzahl von 1 bis 31.
State – Referenz auf Kommunikationsstatus - Variable.

Rückgabewert: SensorID im Erfolgsfall.

Antwort: CS2200 V04.01 (die Zahl 4.01 die Firmwareversion des Gerätes beschreibt)

GetDeviceSerialNumber_DMB()

Diese Funktion liefert die Seriennummer eines Gerätes als String.

Syntax: **function** GetDeviceSerialNumber_DMB (PortNumber, Address: **Integer**; **var** State: **Integer**): **String**;

Parameter: *PortNumber* – Die Nummer des COM Portes.
Address – Busadresse des Gerätes als Integerzahl von 1 bis 31.
State – Referenz auf Kommunikationsstatus - Variable.

Rückgabewert: Seriennummer des Gerätes als String – Variable.

Antwort: 406C120456

GetDeviceSensorNumber_DMB()

Diese Funktion liefert die Sensornummer eines Gerätes als String.

Syntax: **function** GetDeviceSensorNumber_DMB(PortNumber, Address: **Integer**; **var** State: **Integer**): **String**;

Parameter: *PortNumber* – Die Nummer des COM Portes.
Address – Busadresse des Gerätes als Integerzahl von 1 bis 31.
State – Referenz auf Kommunikationsstatus - Variable.

Rückgabewert: Sensornummer des Gerätes als String – Variable.

Antwort: 120456 (Sensornummer ist auch in Seriennummer enthalten)

GetDeviceCalibrationDate_DMB()

Diese Funktion liefert das letzte Kalibrierdatum eines Gerätes.

Syntax: **function** GetDeviceCalibrationDate_DMB(PortNumber, Address: **Integer**; **var** State: **Integer**): **String**;

Parameter: *PortNumber* – Die Nummer des COM Portes.
Address – Busadresse des Gerätes als Integerzahl von 1 bis 31.
State – Referenz auf Kommunikationsstatus - Variable.

Rückgabewert: Datum der letzten Kalibrierung

Antwort: 03.02.2005

GetDeviceChannelCount_DMB()

Diese Funktion liefert die Anzahl der Messkanäle eines Gerätes.

Syntax: **function** GetDeviceChannelCount_DMB(PortNumber, Address: **Integer**; **var** State: **Integer**): **Integer**;

Parameter: *PortNumber* – Die Nummer des COM Portes.
Address – Busadresse des Gerätes als Integerzahl von 1 bis 31.
State – Referenz auf Kommunikationsstatus - Variable.

Rückgabewert: Anzahl der Messkanäle.

Antwort: 5 (vom CS 2000: 4 Kanäle mit Partikelzahlen bzw. Verschmutzungsklassen und Durchfluss)

GetDeviceChannelInfo_DMB()

Diese Funktion dient zur Ermittlung der Messkanal – Eigenschaften eines Gerät. (z.B. Kanalname, Messeinheit, usw.)

Syntax: **function** GetDeviceChannelInfo_DMB(PortNumber, Address, ChNumber, Mode: **Integer**; **var** State: **Integer**): **String**;

Parameter:
PortNumber – Die Nummer des COM Portes.
Address – Busadresse des Gerätes als Integerzahl von 1 bis 31.
ChNumber – Kanalnummer. (von 0 beginnend)
State – Referenz auf Kommunikationsstatus - Variable.

Mode – Messdateneinheiten (wird auch in Funktion „GetDeviceMeasuringValues_DMB“ verwendet)

0 – Partikelzahlen
1 – NAS/SAE - Klassen
2 – ISO – Code

Im Modus 1 bedeutet Wert –1 die Klasse „00“ SAE/NAS und der Wert –2 für „000“ für SAE Klasse (siehe Geräteübersicht in Kapitel 4)

Rückgabewert: Die Antwort besteht aus 5 Subzeilen, die mit einem Trennzeichen voneinander getrennt sind. Die Struktur einer solchen Antwort wird in der folgenden Tabelle dargestellt:

Zeilennummer	***Parameter***	***Anmerkung***
1	Name	Bezeichnung des Kanales
2	Unit	Messbereich, Einheit
3	Decimals	Nachkommastellen Alle Zahlenangaben erfolgen Ganzzahlig. Der Parameter Decimals gibt an, wie viele Stellen der Zahl hinter dem Dezimalpunkt stehen. Z.B. bedeutet die Zahlenangabe: LowerRange = -250, UpperRange = 1000 und Decimals = 1 einen Messbereich von –25,0 bis 100,0.
4	LowerRange	Untere Grenze des Messbereiches
5	UpperRange	Obere Grenze des Messbereiches

Antwort: Temp¶°C¶2¶-2500¶10000¶
(Das Trennzeichen ist ¶ welches den ASCII Code 13 (Return) besitzt)

SetBusAddress_DMB()

Mit folgender Funktion wird eine neue Busadresse im Gerät gesetzt. Im Erfolgsfall gibt die Funktion neue Busadresse als ganze Zahl zurück, sonst – die alte Busadresse.

Syntax: **function** SetBusAddress_DMB (PortNumber, Address, NewAddress: **Integer**; **var** State: **Integer**): **Integer**;

Parameter: *PortNumber* – Die Nummer des COM Portes.
Address – Busadresse des Gerätes als Integerzahl von 1 bis 31.
NewAddress – gewünschte neue Busadresse des Gerätes als Integerzahl von 1 bis 31.
State – Referenz auf Kommunikationsstatus - Variable.

Rückgabewert: Neue Busadresse des Gerätes als Integer – Zahl im Intervall [1..31].

Beispiel: 30 (Neue Busadresse)

Bei einigen Geräten ist ein Neustart oder Reset (Stromversorgung ein/aus) erforderlich.

3.2.6 Messwerte lesen

SetMeasuringState_DMB()

Mit diesem Befehl wird eine Messung gestartet oder gestoppt. Außerdem ist es möglich, mit diesem Befehl den Fehlerstatus des Gerätes rückzusetzen. (falls kein fataler Fehler vorliegt)

Syntax: **function** SetMeasuringState_DMB(PortNumber, Address, Mode: **Integer**; **var** State**: Integer): Integer**;

Parameter: *PortNumber* – Die Nummer des COM Portes.
Address – Busadresse des Gerätes als Integerzahl von 1 bis 31.
Mode – Modus, bezeichnet folgende Aktionen:
0 – Start Messung
1 – Stop Messung
2 – Reset Errorstatus
State – Referenz auf Kommunikationsstatus - Variable.

Rückgabewert: *Betriebszustand*
Die erste Stelle zeigt die Nummer des Messmodus:
1x --> M1, 2x --> M2, usw.
Die zweite Stelle definiert den genauen Zustand:
für M1 (Messen), M2 (Messen u. Schalten), M3 (Filtern bis) gilt:
x0 Messung aus
x1 Warten auf gültigen Durchfluss
x2 Messung läuft

für M4 (Filtern von bis) gilt:
40 Messung aus
41 Warten auf gültigen Durchfluss
42 Messung läuft, Test auf untere Grenze
43 Wartezeit läuft
44 Wartezeit abgelaufen, warten auf gültigen Durchfluss
45 Messung läuft, Test auf obere Grenze

Beispiel: Antwort: 20 (Messmodus M2, Messung aus)

GetDeviceState_DMB()

Mit diesem Befehl kann der aktuelle Status des Gerätes ermittelt werden.

Syntax: **function** GetDeviceState _DMB(PortNumber, Address: **Integer**; **var** State**: Integer): Integer**;

Parameter: *PortNumber* – Die Nummer des COM Portes.
Address – Busadresse des Gerätes als Integerzahl von 1 bis 31.
State – Referenz auf Kommunikationsstatus - Variable.

Rückgabewert: Status des Gerätes. Mögliche Werte:

0 – kein Fehler

1 – neue Messung liegt vor

2 – Filter verschmutzt

4 – Batteriespannung zu gering

Beispiel: Antwort: 1 (neue Messwerte sind vorhanden)

GetDeviceMeasuringValues_DMB()

Mit diesem Befehl werden die Messwerte angefordert und übertragen. Die Ansicht der Messwerte muss mit den Namen von Messkanälen übereinstimmen. (siehe Befehl „GetDeviceChannelInfo_DMB) Als Rückgabewert bekommt man eine String mit Messdaten, die anhand von Kanal – Eigenschaften zu interpretieren ist.

Syntax: **function** GetDeviceMeasuringValues_DMB(PortNumber, Address, Mode: **Integer**; const DeviceID: **String**; **var** State**: Integer): String**;

Parameter: *PortNumber* – Die Nummer des COM Portes.
Address – Busadresse des Gerätes als Integerzahl von 1 bis 31.
Mode – Messdateneinheiten (wird auch in Funktion „GetDeviceChannelInfo_DMB“ verwendet)
DeviceID – Ergebnis der Funktion „SerachBusDevice_DMB“ (zum Beispiel: „CS2200 V04.01“
0 – Partikelzahlen
1 – NAS/SAE - Klassen
2 – ISO-Code
3 – DeviceID – SensorID:
State – Referenz auf Kommunikationsstatus - Variable.

Rückgabewert: Messwerte

Beispiel: Antwort: „7680¶1860¶5¶0¶122¶0¶0¶“, wobei das Zeichen ¶ als Trennzeichen gilt und den ASCII – Code 13 (Return) besitzt.

3.3 HSI - DLL

Die Kommunikation zwischen digitalen Sensoren / Geräten von HYDAC und entsprechenden Auswertegeräten wird auch mit Hydac Sensor Interface (HSI) realisiert.
Dabei handelt es sich um folgenden HSI – Sensoren:

- AS 1000, HLB 1000, CS 1000 und FCU 1000.

Das HSI ist eine digitale 1-Draht Schnittstelle, welche es ermöglicht, Sensoren und PCs miteinander zu verbinden. Ein Sensor / Gerät sendet Messwerte über diese Schnittstelle an eine angeschlossene Auswerteeinheit (zum Beispiel: PC). Die Art und Weise, wie die Daten dabei verpackt werden, wird als HECOM – Protokoll bezeichnet.

Der Adressbereich gemäß HECOM erstreckt sich von 97 bis 122 (ASCII - Code der Zeichen: a ... z). So ergibt sich eine maximale Geräteanzahl von 26 je COM - Schnittstelle. Jeder Sensor muss eine eindeutige Adresse (a... z) zugewiesen werden. Dies gewährleistet, dass der Sensor im BUS angesprochen werden kann.

Sollte nur ein Sensor angeschlossen sein, kann auch das Zeichen ’+’ (ASCII Code 43) benutzt werden.

3.3.1 API - Funktionen

Alle DLL – Funktionen werden in dieser Anleitung mit Hilfe von Pascal – Syntax beschrieben.
Folgende Funktionen stehen in der Datei *hecom32.dll* zur Verfügung:

Funktion	***Kurzbeschreibung***
GetDLLVersion_HSI	DLL – Version als Zahl
GetDLLVersionText_HSI	DLL – Version als Text
SearchOneDevice_HSI	SensorID ermitteln, falls kein Bussystem vorhanden
SearchBusDevice_HSI	SensorID ermitteln in einem Bussystem
GetDeviceChannelCount_HSI	Anzahl der Messkanäle ermitteln
GetDeviceSerialNumber_HSI	Seriennummer ermitteln
GetDeviceChannelInfo_HSI	Messkanal – Eigenschaften ermitteln
GetBusAddress_HSI	Busadresse ermitteln
SetBusAddress_HSI	Busadresse setzen
GetDeviceChannelsMask_HSI	Struktur der Messwerte lesen
GetDeviceMeasuringValues_HSI	Messwerte lesen
GetDeviceState_HSI	Sensorstatus ermitteln

3.3.2 Fehlerbehandlung

Fast alle Funktionen liefern im Fehlerfall einen Fehlercode. Dieser Fehlercode kann folgende Werte beinhalten:

Wert	*Beschreibung*	
0	HSI_NO_ERRORCODE	Kein Fehler. Gerät ist betriebsbereit.
1	HSI_TRANSMITT_ERRORCODE	Fehler bei der Übertragung von Daten zum Gerät.
2	HSI_RECEIVE_ERRORCODE	Fehler bei der Datenübertragung vom Gerät.
3	HSI_TOO_MUCH_DEVICES	Zu viele Geräte gefunden.
4	HSI_SEARCH_ERRORCODE	Fehler in SensorID des Gerätes.
5	HSI_NOCHANNELS_ERRORCODE	Gerät besitzt keinen Kanal.
6	HSI_CHANNELINDEX_ERRORCODE	Falsche Kanalnummer.
7	HSI_CHECKSUM_ERRORCODE	Checksumme falsch.
8	HSI_OPENPORT_ERRORCODE	COM Port kann nicht geöffnet werden.
9	HSI_CHANNELSMASK_ERRORCODE	Messwerte sind außer Toleranzbereich.
10	HSI_NODEVICE_ERRORCODE	Kein Gerät gefunden.
11	HSI_PROTOCOL_ERRORCODE	HSI – Protokollfehler.

3.3.3 Statuskontrolle

GetErrorStateText_HSI()

Mit dieser Funktion kann anhand von Stauscode eine passende Statusmeldung (auf Englisch) ausgegeben werden.

Syntax: **function** GetErrorStateText_HSI(State: **Integer**): **String**;

Parameter: *State* – Kommunikationsstatus.

Rückgabewert: Statusmeldung vom Sensor (Englisch).

Beispiel: Antwort: „10: no device“

3.3.4 DLL – Versionskontrolle

GetDLLVersion_HSI()

Mit dieser Funktion kann die Bibliothekversion ermittelt werden.

Syntax: **function** GetDLLVersion_HSI(): **double**;

Rückgabewert: Die Versionsnummer wird als Double – Zahl zurückgeliefert.

Beispiel: Antwort: 1,03

GetDLLVersionText_HSI()

Mit dieser Funktion kann die Bibliothekversion ermittelt werden.

Syntax: **function** GetDLLVersionText_HSI(): **String**;

Rückgabewert: Die Versionsnummer und Ausgabedatum werden als Text zurückgeliefert.

Beispiel: Antwort: „v.1,03 03.07.2007”

3.3.5 Gerätesuche und Geräteinformation

SearchOneDevice_HSI()

Diese Funktion dient zur Suche eines Gerätes an einem bestimmten COM – Port ohne BUS. Es muss sichergestellt werden, dass an COM – Port **genau ein** Gerät angeschlossen ist.

Syntax: **function** SearchOneDevice_HSI(PortNumber: **Integer**; **var** State: **Integer**): **String**;

Parameter: *PortNumber* – Die Nummer des COM Portes.
State – Referenz auf Kommunikationsstatus - Variable.

Rückgabewert: SensorID im Erfolgsfall.

Beispiel: Eine Antwort kann z.B. folgendermaßen aussehen: „CS1320 V02.21“, wobei die Zahl 2.21 die Firmwareversion des Sensors bezeichnet.

SearchBusDevice_HSI()

Diese Funktion dient zur Suche eines Gerätes an einem bestimmten COM Port mit einer bestimmten Busadresse.

Syntax: **function** SearchBusDevice_HSI (PortNumber: **Integer**; Address: **Integer**; **var** State: **Integer**): **String**;

Parameter: *PortNumber* – Die Nummer des COM Portes.
Address – Busadresse des Gerätes als ASCII - Code des Zeichens.
State – Referenz auf Kommunikationsstatus - Variable.

Rückgabewert: SensorID im Erfolgsfall.

Beispiel: Eine Antwort kann z.B. folgendermaßen aussehen: „AS1000 V02.00“, wobei die Zahl 2.00 die Firmwareversion des Gerätes bezeichnet.

GetDeviceChannelCount_HSI()

Diese Funktion liefert die Anzahl der Kanäle in einem Gerät.

Syntax: **function** GetDeviceChannelCount_HSI (PortNumber, Address: **Integer**; **var** State: **Integer**): **Integer**;

Parameter: *PortNumber* – Die Nummer des COM Portes.
Address – Busadresse des Gerätes als ASCII - Code des Zeichens.
State – Referenz auf Kommunikationsstatus - Variable.

Rückgabewert: Anzahl der Messkanäle.

Beispiel: Antwort: 10 (vom CS 1000)

GetDeviceSerialNumber_HSI()

Diese Funktion liefert die Seriennummer eines Gerätes.

Syntax: **function** GetDeviceSerialNumber_HSI(PortNumber, Address: **Integer**; **var** State: **Integer**): **String**;

Parameter: *PortNumber* – Die Nummer des COM Portes.
Address – Busadresse des Gerätes als ASCII - Code des Zeichens.
State – Referenz auf Kommunikationsstatus - Variable.

Rückgabewert: Seriennummer des Gerätes als String – Variable.

Beispiel: 4711

GetDeviceChannelInfo_HSI()

Diese Funktion dient zur Ermittlung der Kanal – Eigenschaften in einem Gerät. (z.B. Kanalname, Messeinheit usw.)

Syntax: **function** GetDeviceChannelInfo_HSI(PortNumber, Address, ChNumber: **Integer**; **var** State: **Integer**): **String**;

Parameter: *PortNumber* – Die Nummer des COM Portes.
Address – Busadresse des Gerätes als ASCII - Code des Zeichens.
ChNumber – Kanalnummer. (von 0 beginnend)
State – Referenz auf Kommunikationsstatus - Variable.

Rückgabewert: Die Antwort besteht aus 5 Subzeilen, die mit einem Trennzeichen voneinander getrennt sind. Die Struktur einer solchen Antwort wird in der folgenden Tabelle dargestellt:

Zeilennummer	***Parameter***	***Anmerkung***
1	Name	Bezeichnung des Kanales
2	Unit	Messbereich, Einheit
3	Decimals	Nachkommastellen Alle Zahlenangaben erfolgen ganzzahlig. Der Parameter Decimals gibt an, wie viele Stellen der Zahl hinter dem Dezimalpunkt stehen. Z.B. bedeutet die Zahlenangabe: LowerRange = -250, UpperRange = 1000 und Decimals = 1 einen Messbereich von -25,0 bis 100,0
4	LowerRange	Untere Grenze des Messbereiches
5	UpperRange	Obere Grenze des Messbereiches

Beispiel: Eine Antwort kann z.B. folgendermaßen aussehen: „Temp¶°C¶2¶-2500¶10000¶ “, wobei das Zeichen ¶ als Trennzeichen gilt und den ASCII – Code 13 (Return) besitzt.

Verwalten von Busaddressen

Nicht alle HSI – Sensoren unterstützen die nachfolgenden zwei Befehle!

GetBusAddress_HSI()

Diese Funktion gibt die Busadresse eines Gerätes zurück.

Es darf nur ein einziges Gerät an dem COM Port angeschlossen sein.

Syntax:	**function** GetBusAddress_HSI(PortNumber: **Integer**; **var** State: **Integer**): **Integer**;
Parameter:	*PortNumber* – Die Nummer des COM Portes. *State* – Referenz auf Kommunikationsstatus - Variable.
Rückgabewert:	Busadresse des Gerätes als ASCII - Code des Zeichens.
Beispiel:	97 (Zeichen ‘a’)

SetBusAddress_HSI()

Setzt eine neue Busadresse im Sensor.

Syntax:	**function** SetBusAddress_HSI (PortNumber, Address, NewAddress: **Integer**; **var** State: **Integer**): **Integer**;
Parameter:	*PortNumber* – Die Nummer des COM Portes. *Address* – Busadresse des Gerätes als ASCII - Code des Zeichens. *NewAddress* – gewünschte neue Busadresse des Gerätes als ASCII - Code des Zeichens. *State* – Referenz auf Kommunikationsstatus - Variable.
Rückgabewert:	Neue Busadresse des Gerätes als ASCII - Code des Zeichens. Im Erfolgsfall wird neue Adresse zurückgegeben, sonst alte.
Beispiel:	98 (Neue Busdresse ist ‘b’)

3.3.6 Messwerte lesen

GetDeviceChannelsMask_HSI()

Mit diesem Befehl kann ermittelt werden, wie sich die Messwerte zusammensetzen, z.B. ob es sich um 8-bit oder 16-bit Zahlen handelt. Dieser Befehl soll nur einmal ausgeführt werden, damit die sog. „Gerätemaske“ im Weiteren verwendet werden kann.

Syntax: **function** GetDeviceChannelsMask_HSI(PortNumber, Address: **Integer**; **var** State**: Integer): String**;

Parameter: *PortNumber* – Die Nummer des COM Portes.
Address – Busadresse des Gerätes als ASCII - Code des Zeichens.
State – Referenz auf Kommunikationsstatus - Variable.

Rückgabewert: Zeiger auf eine Zeichenvariable, die sog. „Gerätemaske“.

Beispiel: 3¶7¶0¶0¶2¶4¶2¶, wobei das Zeichen ¶ als Trennzeichen gilt und den ASCII – Code 13 (Return) besitzt.

Die Struktur einer solchen „Gerätemaske“ wird in der folgenden Tabelle dargestellt:

Zeilennummer	***Parameter***	***Anmerkung***
1	ChannelCount	Anzahl der Messkanäle
2	ActivityMask	Jedem Kanal ist ein Bit zugeordnet, das anzeigt ob der Kanal aktiv oder nicht aktiv ist
3	MinMask	Jedem Kanal ist ein Bit zugeordnet, das anzeigt ob der Kanal Min – Werte besitzt oder nicht
4	MaxMask	Jedem Kanal ist ein Bit zugeordnet, das anzeigt ob der Kanal Max – Werte besitzt oder nicht
5	DataSize, Kanal 1	Datengröße im 1. Messbereich
6	DataSize, Kanal 2	Datengröße im 2. Messbereich
7	...	... (für jeden Kanal)

Der Parameter „DataSize“ kann nur die Werte 1, 2 oder 4 besitzen. Diese Werte entsprechen 8-, 16- oder 32-bit Werten.

Die „ActivityMask“ gibt an, welche Kanäle tatsächlich aktiv sind. Bei der Messwertübertragung werden inaktive Kanäle nicht übertragen. Bit 0 der Maske zeigt an ob Kanal 0 aktiv ist, Bit 1 Kanal 1 und so weiter.

Mit der Min- und Maxmask wird festgelegt, ob es zu dem jeweiligen Messwert auch noch einen minimal Wert und/oder einen maximal Wert gibt. Bit 0 gehört auch hier zu Kanal 0. Ist ein Kanal nicht aktiv, so sind auch die minimal oder maximal Werte grundsätzlich nicht aktiv. Das bedeutet, ein minimal oder maximal Wert darf ohne Messwert nicht vorkommen.

GetDeviceMeasuringValues_HSI()

Mit diesem Befehl werden die Messwerte angefordert und übertragen.
Die Zusammensetzung der Messwerte muss vorher mit dem Befehl „GetDeviceChannelsMask_HSI“ festgestellt werden. Dabei bekommt man als Antwort sie sog. „Gerätemaske“, die jedes Mal mit dem Befehl „GetDeviceMeasuringValues_HSI“ bestätigt werden muss. Der Grund dafür ist eine Möglichkeit, die Struktur der Messwerte im Betrieb dynamisch anzupassen.

Die Struktur der „Gerätemaske“ wurde bereits für die Funktion „GetDeviceChannelsMask_HSI“ beschrieben.

Syntax:	**function** GetDeviceMeasuringValues_HSI(PortNumber, Address: **Integer**; **const** DeviceChannelsMask: **String**; **var** State**: Integer): String**;
Parameter:	*PortNumber* – Die Nummer des COM Portes. *Address* – Busadresse des Gerätes als ASCII - Code des Zeichens. *DeviceChannelsMask* – „Gerätemaske“. *State* – Referenz auf Kommunikationsstatus - Variable.
Rückgabewert:	Messwerte
Beispiel:	127¶104¶80¶20¶21¶22¶19¶246¶100¶0¶, wobei das Zeichen ¶ als Trennzeichen gilt und den ASCII – Code 13 (Return) besitzt.

Beispiele zur Messwert Interpretation

1) Gerätemaske: „3¶7¶0¶0¶2¶4¶2¶“:

Parameter	***Wert***		
ChannelCount	3		
	Bit0 Kanal0	Bit1 Kanal1	Bit2 Kanal2
ActivityMask	1	1	1
MinMask	0	0	0
MaxMask	0	0	0
DataSize, Kanal 1	2		
DataSize, Kanal 2		4	
DataSize, Kanal 3			2

Die Messwerte vom Gerät: „135¶47¶7¶“stehen hier für:

	Wert
Kanal 1	135
Kanal 2	47
Kanal 3	7

2) Gerätemaske: „3¶6¶2¶0¶2¶4¶2¶“:

Parameter	***Wert***		
ChannelCount	3		
	Bit0 Kanal0	Bit1 Kanal1	Bit2 Kanal2
ActivityMask	1	1	0
MinMask	0	1	0
MaxMask	0	0	0
DataSize, Kanal 1	2		
DataSize, Kanal 2		4	
DataSize, Kanal 3			2

Die Messwerte vom Gerät: „135¶47¶7¶“ stehen nun für:

	Wert
Kanal 1	135
Kanal 2	47
Kanal 2, Minimum	7

GetDeviceState_HSI()

Der Sensorstatus dient dazu festzustellen, ob das angeschlossene Gerät betriebsbereit ist, oder in einen Fehlerzustand eingetreten ist. Der Sensorstatus hat folgenden Aufbau:

- 8-bit Statusbyte,
- 16-bit Statuscode bzw. Fehlercode (mit Vorzeichen)
- Optionaler Statustext

Das *Statusbyte* gibt den aktuellen Zustand des Gerätes an. Die einzelnen Zustände können über den folgenden Statuscode näher spezifiziert werden.

Folgende Werte für das Statusbyte sind definiert:

0:	Betriebsbereit	Kein aktiver Fehler vorhanden, Gerät ist betriebsbereit.
1:	Stand-by	Kein aktiver Fehler vorhanden, Gerät ist aber zur Zeit nicht betriebsbereit, eventuell sind einzelne Gerätefunktionen abgeschaltet, oder Gerät ist in einer Anlaufphase, etc.
2:	Leichter Fehler	Es ist ein leichter Fehler vorhanden, der quittiert werden kann.
3:	Mittlerer Fehler	Es ist ein mittelschwerer Fehler vorhanden, der durch Ein/Ausschalten eventuell behebbar ist.
4:	Schwerer Fehler	Es ist ein schwerer Fehler vorhanden, das Gerät muss zum Hersteller zurück.

Der *Statuscode* spezifiziert den aktuellen Zustand näher. Es ist ein 16-bit Wert. Die genaue Bedeutung ist von Gerät zu Gerät unterschiedlich. Der Anwender kann dann dem Handbuch nähere Infos zu dem Statuscode entnehmen.

Der *Statustext* ist optional und max. 32 Zeichen lang. Er dient dazu, dass ein Bediengerät den Status eines Sensors im Klartext anzeigen kann.

Syntax:	**function** GetDeviceState_HSI(PortNumber, Address: **Integer**; **var** StateByte, StateCode, State: **Integer**): **String**;
Parameter:	*PortNumber* – Die Nummer des COM Portes. *Address* – Busadresse des Gerätes als ASCII - Code des Zeichens. *StateByte* – StatusByte. *StateCode* – Statuscode. *State* – Referenz auf Kommunikationsstatus - Variable.
Rückgabewert:	Statustext.
Beispiel:	Antwort: „ASIC-CRC-Error“, *StateByte* = 3, *StateCode* =17

3.4 Beispiele

Um die Hochsprachenprogrammierung mit FluMoT DLLs zu erleichtern, werden einfache Beispiele als kleine Projekte in Delphi7, LabView 7 und Excel -Makros (VBA 6) im Quellcode mitgeliefert.

Diese Beispiele befinden sich nach der Installation im Verzeichnis:

[LW]:\…..\FluMoT\Dlls\Examples

Dabei handelt es sich nicht um die fertigen Softwareprodukten, sondern um die kleinen Demo – Programmen.

LabView – Beispiel (EXE - Datei) ist nur ausführbar, wenn LabView Runtime Engine 7 installiert ist.
Sollte keine Runtime Engine 7 installiert sein, finden Sie im Beispiel – Ordner eine vollständige Installation.

4 Messkanal Übersicht

In der nachfolgenden Tabelle wird ein Übersicht der Messkanäle von unterschiedlichen HYDAC – Sensoren dargestellt.

4.1 FCU 2000 Serie

FCU 20xx						
	Partikelzahlen		NAS/SAE Klasse		ISO Code	
Kanal 1	5-15 µm	[0..4096000]	NAS 5-15 µm	[-1..15]	ISO >5 µm	[0..25]
Kanal 2	15-25 µm	[0..729000]	NAS 15-25 µm	[-1..15]	ISO >15 µm	[0..25]
Kanal 3	25-50 µm	[0..129600]	NAS 25-50 µm	[-1..15]	ISO >25 µm	[0..25]
Kanal 4	>50 µm	[0..23040]	NAS >50 µm	[-1..15]	ISO >50 µm	[0..25]
Kanal 5	Flow ml/min	[0..800]	Flow ml/min	[0..800]	Flow ml/min	[0..800]
-	-	-	-	-	-	-

FCU 21xx						
	Partikelzahlen		NAS/SAE Klasse		ISO Code	
Kanal 1	2-5µm	[0..20484000]	NAS 2-5µm	[-1..15]	ISO >2µm	[0..25]
Kanal 2	5-15µm	[0..4096000]	NAS 5-15µm	[-1..15]	ISO >5µm	[0..25]
Kanal 3	15-25µm	[0..729000]	NAS 15-25µm	[-1..15]	ISO >15µm	[0..25]
Kanal 4	>25 µm	[0..129600]	NAS >25µm	[-1..15]	ISO >25µm	[0..25]
Kanal 5	Flow ml/min	[0..800]	Flow ml/min	[0..800]	Flow ml/min	[0..800]
-	-	-	-	-	-	-

FCU 22xx						
	Partikelzahlen		NAS/SAE Klasse		ISO Code	
Kanal 1	> 4 µm	[0..3200000]	SAE A	[-2..15]	ISO >4µm	[0..25]
Kanal 2	> 6 µm	[0..1250000]	SAE B	[-2..15]	ISO >6µm	[0..25]
Kanal 3	> 14 µm	[0..222000]	SAE C	[-2..15]	ISO >14 µm	[0..25]
Kanal 4	> 21 µm	[0..39200]	SAE D	[-2..15]	ISO >21 µm	[0..25]
Kanal 7	Flow ml/min	[0..800]	Flow ml/min	[0..800]	Flow ml/min	[0..800]
-	-	-	-	-	-	-

4.2 FCU 8000 Serie

FCU 81xx						
	Partikelzahlen		NAS/SAE Klassen		ISO Code	
Kanal 1	2-5µm	[0..20484000]	NAS 2-5µm	[-1..15]	ISO > 2 µm	[0..25]
Kanal 2	5-15µm	[0..4096000]	NAS 5-15µm	[-1..15]	ISO > 5 µm	[0..25]
Kanal 3	15-25µm	[0..729000]	NAS 15-25µm	[-1..15]	ISO > 15 µm	[0..25]
Kanal 4	25-50µm	[0..129600]	NAS 25-50µm	[-1..15]	ISO > 25 µm	[0..25]
Kanal 5	50-100µm	[0..23040]	NAS 50-100µm	[-1..15]	ISO > 50 µm	[0..25]
Kanal 6	>100µm	[0..4096]	NAS > 100µm	[-1..15]	ISO > 100 µm	[0..25]
Kanal 7	Flow ml/min	[0..800]	Flow ml/min	[0..800]	Flow ml/min	[0..800]

FCU 82xx						
	Partikelzahlen		NAS/SAE Klassen		ISO Code	
Kanal 1	> 4 µm	[0..3200000]	SAE A	[-2..15]	ISO > 4 µm	[0..25]
Kanal 2	> 6 µm	[0..1250000]	SAE B	[-2..15]	ISO > 6 µm	[0..25]
Kanal 3	>14 µm	[0..222000]	SAE C	[-2..15]	ISO > 14 µm	[0..25]
Kanal 4	> 21 µm	[0..39200]	SAE D	[-2..15]	ISO > 21 µm	[0..25]
Kanal 5	> 38 µm	[0..6780]	SAE E	[-2..15]	ISO > 38 µm	[0..25]
Kanal 6	> 70 µm	[0..1020]	SAE F	[-2..15]	ISO > 70 µm	[0..25]
Kanal 7	Flow ml/min	[0..800]	Flow ml/min	[0..800]	Flow ml/min	[0..800]

4.3 CS 2000 Serie

CS 20xx						
	Partikelzahlen		NAS/SAE Klasse		ISO Code	
Kanal 1	5-15 µm	[0..4096000]	NAS 5-15 µm	[-1..15]	ISO > 5 µm	[0..25]
Kanal 2	15-25 µm	[0..729000]	NAS 15-25 µm	[-1..15]	ISO > 15 µm	[0..25]
Kanal 3	25-50 µm	[0..129600]	NAS 25-50 µm	[-1..15]	ISO > 25 µm	[0..25]
Kanal 4	>50 µm	[0..23040]	NAS >50 µm	[-1..15]	ISO > 50 µm	[0..25]
Kanal 5	Flow ml/min	[0..800]	Flow ml/min	[0..800]	Flow ml/min	[0..800]
Kanal 6	Analog 1 (bei Firmware Version ≥ 4.00)					
Kanal 7	Analog 2 (bei Firmware Version ≥ 4.00)					

CS 21xx						
	Partikelzahlen		NAS/SAE Klasse		ISO Code	
Kanal 1	2-5µm	[0..20484000]	NAS 2-5 µm	[-1..15]	ISO > 2 µm	[0..25]
Kanal 2	5-15µm	[0..4096000]	NAS 5-15 µm	[-1..15]	ISO > 5 µm	[0..25]
Kanal 3	15-25µm	[0..729000]	NAS 15-25 µm	[-1..15]	ISO > 15 µm	[0..25]
Kanal 4	>25 µm	[0..129600]	NAS >25 µm	[-1..15]	ISO > 25 µm	[0..25]
Kanal 5	Flow ml/min	[0..800]	Flow ml/min	[0..800]	Flow ml/min	[0..800]
Kanal 6	Analog 1 (bei Firmware Version ≥ 4.00)					
Kanal 7	Analog 2 (bei Firmware Version ≥ 4.00)					

CS 22xx						
	Partikelzahlen		NAS/SAE Klasse		ISO Code	
Kanal 1	> 4 µm	[0..3200000]	SAE A	[-2..15]	ISO > 4 µm	[0..25]
Kanal 2	> 6 µm	[0..1250000]	SAE B	[-2..15]	ISO > 6 µm	[0..25]
Kanal 3	> 14 µm	[0..222000]	SAE C	[-2..15]	ISO > 14 µm	[0..25]
Kanal 4	> 21 µm	[0..39200]	SAE D	[-2..15]	ISO > 21 µm	[0..25]
Kanal 7	Flow ml/min	[0..800]	Flow ml/min	[0..800]	Flow ml/min	[0..800]
Kanal 6	Analog 1 (bei Firmware Version ≥ 4.00)					
Kanal 7	Analog 2 (bei Firmware Version ≥ 4.00)					

Notizen

HYDAC INTERNATIONAL

HYDAC Filtertechnik GmbH
Bereich Servicetechnik / Service Technology Division
Industriegebiet
66280 Sulzbach/Saar
Germany

Postfach 1251
66273 Sulzbach/Saar
Germany

Tel: +49 (0) 6897 509 01
Fax: +49 (0) 6897 509 846 (Technik / Technical Department)
Fax: +49 (0) 6897 509 577 (Verkauf / Sales Department)

Internet: www.hydac.com
email: filtersysteme@hydac.com