technologies

Programmierhilfe BRH 100DC

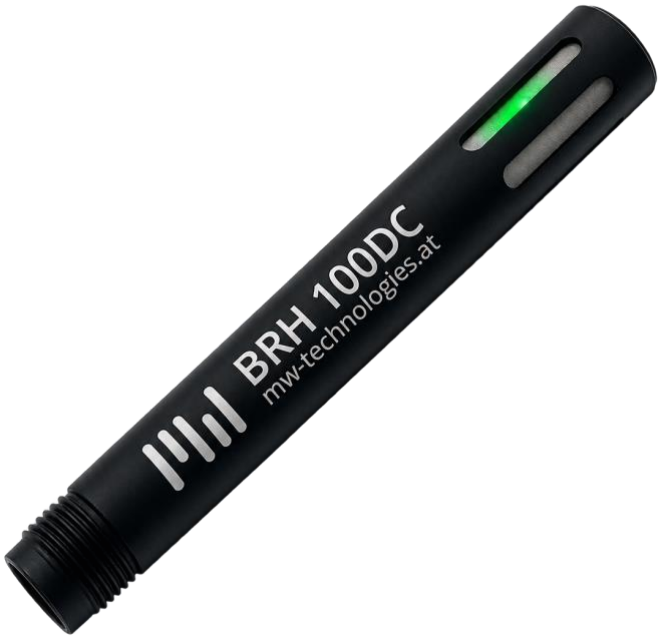
Datacenter Feuchte- und Temperatursensor

Hilfestellung und Beispiele für eine Softwareintegration

16. September 2026

Dieses Dokument beschreibt die praxisnahe Ansteuerung des BRH 100DC über RS485 / Modbus RTU. Der Fokus liegt auf der Integration in eigene Software, SPS, HMI und Datenlogger sowie auf einem schnellen Funktionstest mit einem üblichen Modbus-Master.

Die Programmierhilfe enthält für den Endanwender relevante Schnittstellen, Messwerte, Geräteinformationen und LED-/Kommunikationsfunktionen.



1. Inhalt

Inhaltsverzeichnis

2. Zweck und Geltungsbereich

Der BRH 100DC stellt Temperatur, relative Feuchte und daraus berechnete Klimagrößen über Modbus RTU bereit. Die Messwerte können wahlweise als skalierte 16-Bit-Werte oder als IEEE754-FLOAT32 ausgelesen werden.

- Temperatur in °C und °F
- Relative Feuchte in %RH
- Taupunkt in °C und °F
- Absolute Feuchte in g/m³
- Mischungsverhältnis in g/kg

Die Messwertaktualisierung erfolgt im 1-s-Raster. Für die Busabfrage wird ein Polling-Intervall von mindestens 100 ms empfohlen.

3. Elektrischer Anschluss

Der BRH 100DC verwendet einen 4-poligen M12-Anschluss. Für die Softwareintegration ist insbesondere die korrekte Zuordnung der RS485-Leitungen relevant.

Pin	Signal	Funktion
1	+UB	Versorgungsspannung
2	B-RS485 / Data-	RS485 B / invertierte Datenleitung
3	GND	Masse
4	A-RS485 / Data+	RS485 A / nicht invertierte Datenleitung

4. Kommunikationsgrundlagen

Parameter	BRH 100DC Default / Verhalten
Schnittstelle	RS485 / Modbus RTU Slave
Slave-ID	235
Baudrate	9600 Baud
Datenbits	8
Parität	None (N)
Stopbits	1
Messwertaktualisierung	1 s
empfohlenes minimales Polling	≥ 100 ms

Unterstützte Baudraten: 9600, 19200, 38400, 57600, 76800 und 115200 Baud.

Unterstützte Modbus-Funktionen:

- FC03 – Read Holding Registers: Messwerte und Parameter lesen.
- FC06 – Write Single Register: einzelne Kommunikations- oder LED-Parameter schreiben.
- FC16 – Write Multiple Registers: mehrere Register bzw. 32-Bit-Werte schreiben, sofern für einen Parameter vorgesehen.
- FC04 – Read Input Registers ist beim BRH 100DC nicht umgesetzt. Für Messwerte FC03 verwenden.

FLOAT32: Ein FLOAT32 belegt zwei 16-Bit-Register. Die Wortreihenfolge ist Low-Word zuerst, anschließend High-Word.

5. Einstieg mit einem Modbus-Master

Für einen ersten Funktionstest kann ein beliebiger Modbus-RTU-Master verwendet werden, z. B. eine SPS, ein USB/RS485-Testprogramm oder eine eigene PC-Anwendung.

Parameter	Wert
Port	z. B. COM5 am USB/RS485-Adapter
Verbindung	9600, N, 8, 1
Slave-ID	235
Function Code	03 – Read Holding Registers
Registergröße	16 Bit
Erster Test	Start 4000, Count 4

Mit Startadresse 4000 und Count 4 werden die vier wichtigsten Messwerte in einem Block gelesen: Temperatur, relative Feuchte, Taupunkt und absolute Feuchte.

4.1 Beispiel zur Interpretation

Adresse	Rohwert	Bedeutung	Skalierung	Ergebnis
4000	2534	Temperatur	÷100	25,34 °C
4001	4521	Relative Feuchte	÷100	45,21 %RH
4002	1284	Taupunkt	÷100	12,84 °C
4003	1037	Absolute Feuchte	÷100	10,37 g/m³

Vorzeichen: Temperatur- und Taupunktregister als vorzeichenbehaftete INT16 interpretieren. Das ist besonders bei Werten unter 0 °C wichtig.

6. Register für Endanwender

Alle nachfolgend aufgeführten Adressen sind Modbus-Holding-Register und werden mit FC03 gelesen. Schreibzugriffe sind nur bei ausdrücklich als RW oder WO gekennzeichneten Registern vorgesehen.

5.1 Geräteidentifikation

Adresse DEZ	HEX	Zugriff	Inhalt	Datentyp
0…7	0x0000…0x0007	RO	Seriennummer	ASCII, 16 Byte
8	0x0008	RO	Firmwareversion	UINT16: Hi=Major, Lo=Minor
9…16	0x0009…0x0010	RO	Gerätebezeichnung	ASCII, 16 Byte

Gerätebezeichnung: Für Endanwender ist der Identifikationsbereich als Lesebereich dokumentiert. Schreib- und Servicefunktionen zur Geräteidentifikation sind nicht Bestandteil dieser Programmierhilfe.

5.2 Messwerte INT16

DEZ	HEX	Zugriff	Messwert	Skalierung
4000	0x0FA0	RO	Temperatur	°C ×100
4001	0x0FA1	RO	Relative Feuchte	%RH ×100
4002	0x0FA2	RO	Taupunkt	°C ×100
4003	0x0FA3	RO	Absolute Feuchte	g/m³ ×100
4014	0x0FAE	RO	Temperatur Fahrenheit	°F ×100
4015	0x0FAF	RO	Taupunkt Fahrenheit	°F ×100
4016	0x0FB0	RO	Mischungsverhältnis	g/kg ×100

Für SPS/HMI-Anwendungen sind die INT16-Werte meist die einfachste Integrationsvariante. Die physikalische Größe ergibt sich aus Rohwert ÷100.

5.3 Messwerte FLOAT32

DEZ	HEX	Zugriff	Messwert	Einheit
1000/1001	0x03E8–0x03E9	RO	Temperatur	°C
1002/1003	0x03EA–0x03EB	RO	Relative Feuchte	%RH
1004/1005	0x03EC–0x03ED	RO	Taupunkt	°C
1006/1007	0x03EE–0x03EF	RO	Absolute Feuchte	g/m³
1026/1027	0x0402–0x0403	RO	Temperatur Fahrenheit	°F
1028/1029	0x0404–0x0405	RO	Taupunkt Fahrenheit	°F
1030/1031	0x0406–0x0407	RO	Mischungsverhältnis	g/kg

Konsistenz: Beide Register eines FLOAT32-Werts immer innerhalb eines einzigen Modbus-Leseauftrags lesen.

Mischungsverhältnis: Für die Berechnung wird ein Umgebungsdruck verwendet. Im aktuellen BRH-100DC

Endanwenderprofil ist dafür kein separates öffentliches Einstellregister spezifiziert; der interne Default beträgt 1013,25 mbar.

5.4 Serielle Kommunikationsparameter

DEZ	HEX	Zugriff	Parameter	Beschreibung
530	0x0212	RW	SerialBaudCode	0=9600, 1=19200, 2=38400, 3=57600, 4=76800, 5=115200
531	0x0213	RW	SerialParity	0=None, 1=Even, 2=Odd
532	0x0214	RW	SerialStopBits	1 oder 2
533	0x0215	RW	ModbusAddress	1…247; Default 235
534	0x0216	RW	ResponseDelayMs	Antwortverzögerung in ms
535	0x0217	RW	FrameTimeoutMs	Frame-Timeout in ms
536	0x0218	WO	SerialApply	0xA5A5 schreiben, um neue Parameter zu übernehmen

Ablauf: Zuerst die gewünschten Kommunikationsregister schreiben, anschließend Register 536 mit 0xA5A5 beschreiben. Danach die Verbindung mit den neuen Parametern neu aufbauen.

7. Implementierungsbeispiele in Python

Die Beispiele orientieren sich an der Struktur der BRH-100RA-Programmierhilfe und verwenden pymodbus. Port und Slave-ID sind an die eigene Installation anzupassen.

6.1 INT16-Messwerte lesen

```
from pymodbus.client import ModbusSerialClient

def to_int16(v):
   return v - 65536 if v > 32767 else v

c = ModbusSerialClient(
   port='COM5', baudrate=9600, parity='N',
   stopbits=1, bytesize=8, timeout=0.5
)

if c.connect():
   r = c.read_holding_registers(4000, 4, slave=235)
   t  = to_int16(r.registers[0]) / 100.0
   rh = r.registers[1] / 100.0
   td = to_int16(r.registers[2]) / 100.0
   ah = r.registers[3] / 100.0
   print(t, rh, td, ah)

c.close()
```

6.2 FLOAT32 zusammensetzen

```
import struct

def regs_to_float(lo, hi):
   raw32 = ((hi & 0xFFFF) << 16) | (lo & 0xFFFF)
   return struct.unpack('<f', struct.pack('<I', raw32))[0]

r = c.read_holding_registers(1000, 8, slave=235)
t  = regs_to_float(r.registers[0], r.registers[1])
rh = regs_to_float(r.registers[2], r.registers[3])
td = regs_to_float(r.registers[4], r.registers[5])
ah = regs_to_float(r.registers[6], r.registers[7])
print(t, rh, td, ah)
```

Wortreihenfolge: Der erste gelesene 16-Bit-Wert ist das Low-Word, der zweite das High-Word.

6.3 Modbus-Adresse ändern

```
from pymodbus.client import ModbusSerialClient

c = ModbusSerialClient(
   port='COM5', baudrate=9600, parity='N',
   stopbits=1, bytesize=8, timeout=0.5
)

if c.connect():
   # neue Slave-ID 236 setzen
   c.write_register(533, 236, slave=235)

   # neue serielle Parameter übernehmen
   c.write_register(536, 0xA5A5, slave=235)

c.close()
# Danach mit Slave-ID 236 neu verbinden.
```

Sicherer Ablauf: Immer nur die tatsächlich gewünschten seriellen Parameter ändern. Ein Schreibzugriff auf andere Setup- oder Servicebereiche ist für die Änderung der Slave-ID nicht erforderlich.

6.4 Geräteidentifikation lesen

```
def regs_to_ascii(regs):
   b = bytearray()
   for reg in regs:
      b.append((reg >> 8) & 0xFF)
      b.append(reg & 0xFF)
   return b.rstrip(b'\x00').decode('ascii', errors='replace')

r = c.read_holding_registers(0, 17, slave=235)
serial = regs_to_ascii(r.registers[0:8])
fw = r.registers[8]
model = regs_to_ascii(r.registers[9:17])
print(serial, f'{(fw >> 8) & 0xFF}.{fw & 0xFF}', model)
```

8. RGB-Statusanzeige

Der BRH 100DC kann über eine RGB-Statusanzeige verfügen. Die Anzeige dient als schnelle Diagnosehilfe für Betrieb, Modbus-Kommunikation und Messwertzustand.

Zustand	Anzeige	Bedeutung
Boot / Selbsttest	Weiß kurz	Gerät startet
Normalbetrieb	Grün langsam blinkend	Messwerte gültig, Modbus bereit
Modbus RX/TX	Blau kurzer Puls	Telegramm empfangen oder beantwortet
Messwertwarnung	Gelb langsam blinkend	Messwertwarnung / RH-Grenzbereich
Sensorfehler	Rot langsam blinkend	Sensor nicht erreichbar oder dauerhafter Sensorfehler
Bus-/Kommunikationsfehler	Blau/Rot wechselnd	RS485-/Modbus-Fehlerzähler über Grenzwert
Service-/Konfigurationsmodus	Cyan	Konfigurationsmodus aktiv
Interner Firmwarefehler	Rot schnell blinkend	Interner Fehlerzustand

7.1 RGB-Register

DEZ	HEX	Zugriff	Parameter	Beschreibung
505	0x01F9	RO	RGB_Status	Statusregister, Bitfeld
570	0x023A	RW	RGB_Mode	LED-Betriebsart
571	0x023B	RW	RGB_Pattern	Blink-/Farbmuster
572	0x023C	RW	RGB_Brightness	Helligkeit 0…100 %
573	0x023D	WO	RGB_Apply	0xA5A5 übernimmt RGB-Einstellungen
574	0x023E	RW	RGB_Enable	0=aus, 1=ein

7.2 RGB_Status – relevante Bits

Bit	Wert	Anzeige / Zustand
b0	1	RGB aus / deaktiviert
b1	2	Boot / Selbsttest
b2	4	Normalbetrieb
b3	8	Modbus RX/TX
b4	16	Sensorfehler
b5	32	Bus-/Kommunikationsfehler
b6	64	Service-/Konfigurationsmodus

Bit	Wert	Anzeige / Zustand
b7	128	Relative Feuchte > 80 %RH

9. Typische Fehlerbilder

Beobachtung	Prüfung / Abhilfe
Keine Antwort	COM-Port, Versorgung, Slave-ID 235, 9600 8N1 sowie A/B prüfen; A/B bei unklarer Adapterbezeichnung testweise tauschen.
FC04 liefert keine Werte	FC03 verwenden. FC04 ist beim BRH 100DC nicht umgesetzt.
Messwerte um Faktor 100 zu groß	INT16-Skalierung beachten: Rohwert ÷100.
Negative Temperaturen erscheinen sehr groß	Temperatur und Taupunkt als signed INT16 interpretieren.
FLOAT32 unplausibel	Low-Word vor High-Word zusammensetzen und beide Register in einem Leseauftrag lesen.
Nach Änderung keine Kommunikation	Nach Schreiben von 530…535 Register 536 mit 0xA5A5 schreiben und anschließend mit den neuen Parametern neu verbinden.
LED-Muster verhält sich unerwartet	Mode/Pattern-Werte nur verwenden, wenn sie zur eingesetzten Firmwareversion dokumentiert sind.

10. Empfehlungen für die Integration

- Für die Erstinbetriebnahme FC03, Start 4000, Count 4 verwenden.
- Für SPS/HMI vorzugsweise INT16-Messwerte nutzen; für PC-Software und Datenlogger sind FLOAT32-Werte praktisch.
- FLOAT32 immer als zusammengehöriges Registerpaar in einem Request lesen.
- Das Polling nicht schneller als 100 ms ausführen; neue Messwerte werden im 1-s-Raster bereitgestellt.
- Serielle Parameter zuerst schreiben und anschließend mit 0xA5A5 in Register 536 übernehmen.
- Nur die in dieser Programmierhilfe veröffentlichten Endanwenderregister beschreiben.
- Bei mehreren Geräten auf einem Bus eindeutige Slave-IDs vergeben und RS485 busförmig verdrahten.

Praxisempfehlung: Für kontinuierliche Klimadatenerfassung ist ein Abfragezyklus von etwa 10 s pro Sensor ausreichend, da der Messwertsatz wird jedoch einmal pro Sekunde aktualisiert wird.

11. Kompakte Registerreferenz

Bereich	Adressen	Zugriff	Verwendung
Geräteidentifikation	0…16	RO	Seriennummer, Firmwareversion, Gerätebezeichnung
RGB-Status	505	RO	Geräte-/LED-Zustand
Serielle Parameter	530…535	RW	Baudrate, Parität, Stopbits, Adresse, Timing
SerialApply	536	WO	0xA5A5 – neue serielle Parameter übernehmen
RGB-Steuerung	570…574	RO/RW/WO	LED-Funktionen; Pattern nur firmwareversionsbezogen verwenden
FLOAT32 Messwerte	1000…1007, 1026…1031	RO	Klimamesswerte als IEEE754
INT16 Messwerte	4000…4003, 4014…4016	RO	Skalierte Klimamesswerte ×100

12. Revision History

Datum	Revision	Änderung
2026-09-16	Rev. 0	Erstausgabe der BRH-100DC-Endanwender-Programmierhilfe.

sensors. simplified.