Benutzerhandbuch myDatalogC33x

Gültig ab:

- **Firmware Version: 01v024**
- **Server Version: 50v007**
- **Hardware Version: 1.2**

Microtronics

Kapitel 1 Inhaltsverzeichnis

Kapitel 2 Konformitätserklärung

2.1 myDatalogC33x WIFI/2G/3G/4G World

EU-Konformitätserklärung

EU Declaration of Conformity / Déclaration de conformité UE

Produktbezeichnung:
Product:
Désignation du produit:

Stationäres, kompaktes, frei programmierbares Gerät zur Erfassung, Verarbeitung und Übertragung von Signalen und Geräteinformationen

Type :
Type code:
Type:

myDatalogC3xx 2G/3G/4G World

Gültig ab:
Valid from:
Valide à partir de:

Rev. 1.2

Microtronics

Hersteller:
Manufacturer :
Fabricant:

Microtronics Engineering GmbH
Hauptstrasse 7
A-3244 Ruprechtshofen

Das bezeichnete Produkt stimmt mit den folgenden Europäischen Richtlinien überein.
The designated product is in conformity with the following european directives.
Le produit décrit est conforme aux directives européennes suivantes.

		Europäische Norm	
(2014/30/EU)	**EMC Directive**		
		EN61000-4-3 (testlevel 3) EN55032 (class A) EN61326-1	
(2014/35/EU)	**LVD Directive**		
		EN61010-1	
(2014/53/EU)	**RED Directive**		
	Safety & Health 3.1a	EN62368-1 EN62368-1+A11:2017 EN62311 EN62479	
	EMC 3.1b	EN301489-1 V2.2.3 EN301489-52 V1.1.0	
	Radio spectrum efficiency 3.2	EN301511 V12.5.1 EN301908-1 V13.1.1 EN301908-2 V13.1.1 EN301908-13 V13.1.1 EN300328 V2.1.1	
(2015/863/EU)	**RoHS Directive**		
	Prevention 4.1	EN IEC 63000	

Ruprechtshofen, den 22.09.2023

Ort und Datum der Ausstellung
Place and date of issue
Lieu et date d'établissement

Hans-Peter Buber, Managing Director
Unterschrift
name and signature of authorised person
Nom et signature de la personne autorisée

Kapitel 3 Technische Daten

Spannungsversorgung	9...32VDC (+/-10%) Zusätzliche Informationen finden Sie unter "Technische Details zur Energieversorgung" auf Seite 67.
Leistungsaufnahme	typ. 5W (ohne Sensoren) max. 9W (ohne Sensoren)
Integrierter Pufferakku	Li-Po -Akku mit 500mAh für: • Applikative Reaktion auf den Ausfall der Versorgung • Ausbuchen aus dem Mobilfunknetz bei Ausfall der Versorgung Zusätzliche Informationen finden Sie unter "Technische Details zum integrierten Pufferakku" auf Seite 65.
Gehäuse	Material: Lexan/Noryl Gewicht: 190g Schutzart: IP20 / IP40 (Anschlussbereich / Betätigungsbereich) Abmessungen (BHT): Abmessungen (BHT): 70 x 92 x 63mm (ohne Antenne)
Betriebstemperatur	-20...+60°C
Luftfeuchtigkeit	15...90%rH nicht kondensierend
Lagertemperatur	-40...+85°C
Ladetemperatur (Pufferakku)	0 ...+45°C
Anzeige	RGB-LED (frei verwendbar, gesteuert durch die Device Logic)
Bedienung	MDN Taste (frei verwendbar, Auswertung durch die Device Logic) Taste zum Auslösen eines Resets
Systemzeit	Hardware Real-Time Clock mit eigenständiger Pufferbatterie (Laufzeit >10 Jahre) und automatische Zeitsynchronisation mit dem Server. Zusätzliche Informationen finden Sie unter "Technische Details zur Systemzeit" auf Seite 67.
Antennenanschluss	1 x SMA-F für Mobilfunk 1 x SMA-F für WiFi

Universaleingänge	3 x analog oder digital Modi: • 0...20mA: Auflösung 6,36µA, max. 23,96mA, Bürde 96Ω • 4...20mA: Auflösung 6,36µA, max. 23,96mA, Bürde 96Ω • 0...2V: Auflösung 610µV, max. 2,5V, Bürde 10k086 • 0...10V: Auflösung 7,97mV , max. 32V, Bürde 4k7 • PWM: 1...99%, max. 100Hz, Impulslänge min. 1ms, max. 32V, Bürde 4k7 • Frequenz: 1...1000Hz, max. 32V, 4k7 • Digital: low <0,99V, high >2,31V, max. 32V, Bürde 4k7 • Zähler: Impulslänge min. 1ms, max. 32V, Bürde 4k7 Zusätzliche Informationen finden Sie unter "Technische Details zu den Universaleingängen" auf Seite 59.

Serielle Schnittstelle	1 x RS485 (2-Leiter)
	• Baudrate: 2400-115200
	• Stopbits: 1, 2
	• Parität: N, E, O
	• Datenbits: 7, 8
	• Abschlusswiderstand: Aus, 120Ω
	• Pull up an RS485 B: Aus, 390Ω
	• Pull down an RS485 A: Aus, 390Ω
	
	1 x CAN / CAN FD
	• max. Datenübertragungsrate:
	  • CAN: 1Mbit/s
	  • CAN FD: 8Mbit/s
	• unterstützte Spezifikation:
	  • CAN: V2.0B
	  • CAN-FD: V1.0
	• keine galvanische Trennung
	• Abschlusswiderstand: Aus, 120Ω , 2k
	
	1 x RS232 (4-Leiter)
	• Baudrate: 2400-115200
	• Stopbits: 1, 2
	• Parität: N, E, O
	• Datenbits: 7, 8
	• Flusskontrolle: Aus, RTS/CTS
	
	Zusätzliche Informationen finden Sie unter "Technische Details zur RS485-Schnittstelle" auf Seite 60, "Technische Details zur CAN-Schnittstelle" auf Seite 61 und "Technische Details zur RS232-Schnittstelle" auf Seite 63.

Ausgänge	2 x potentialfreier Schaltkontakt • galvanisch getrennt • Parallelschaltung eines Optoswitchs und eines Relais Relais (hohe Schaltströme) • U_{max}: 32V • I_{max}: 2A Optoswitch (hohe Schaltfrequenzen) • I_{max}: 130mA • U_{max}: 32V • R_{on}: 35Ω • f_{max}: 1000Hz Zusätzliche Informationen finden Sie unter "Technische Details zu den Ausgängen" auf Seite 65.
USB-Schnittstelle	1 x Mini-B USB 2.0 Slave für die Verbindung mit einem PC. Für die Kommunikation mit dem myDatalogC33x muss am PC das Konfigurationsprogramm DeviceConfig installiert sein oder die webbasierte Entwicklungsumgebung rapidM2M Studio verwendet werden. Zusätzliche Informationen finden Sie unter "Technische Details zur USB-Schnittstelle" auf Seite 64.
Datenspeicher	3 MB interner Flash-Speicher. Die Größe der Datensätze ist variabel (max. 1024 Byte) und wird durch die vom User erstellte Device Logic bestimmt. Der systembedingte Overhead beträgt pro Datensatz 11 Byte . Zusätzliche Informationen finden Sie unter"Funktionsweise des internen Datenspeichers " auf Seite 31.
Konfigurationsspeicher	10 unabhängige Blöcke mit je 4000 Bytes
Registrierungsspeicher	Flash: 4 Blöcke mit je 1kB und vordefinierten Verwendungszwecken zur Ablage gerätespezifischer Daten RAM: 1 optionaler Block mit max. 1kB zur Ablage applikationsspezifischer Daten Zusätzliche Informationen finden Sie unter "Registrierungsspeicherblöcke" auf Seite 36.
Speicher für das Device Logic Binary	8MB (unkomprimierte Größe) Zusätzliche Informationen finden Sie unter "Speicherorganisation" auf Seite 33.

Datenübertragung	2G/3G/4G Welt (myDatalogC33x WIFI/2G/3G/4G World): • 2G GPRS 900MHz / 1800MHz • 2G GPRS 850MHz / 1900MHz • UMTS B1, B2, B5, B8 • LTE FDD B1, B2, B3, B4, B5, B7, B8, B12, B13, B18, B19, B20, B26, B28 • LTE TDD B38, B39, B40, B41 WiFi: [1)] • Einzelband 2,4 GHz • IEEE 802.11 b/g/n • Kanalbandbreite: 20MHz • Unterstützte Kanäle: 1-14 Netzwerkschnittstelle: • 10/100 Ethernet
SIM	Der myDatalogC33x ist mit einem integrierten SIM-Chip versehen.

[1)] kann auch für die Kommunikation mit anderen Geräten und Sensoren genutzt werden (sofern nicht als Uplink eingesetzt)

Kapitel 4 Allgemeine Angaben

Die Informationen dieses Handbuchs wurden sorgfältig geprüft und nach bestem Wissen zusammengestellt. Der Hersteller übernimmt dennoch keine Verantwortung für möglicherweise in diesem Handbuch enthaltene falsche Angaben. Der Hersteller ist nicht verantwortlich für direkte, indirekte, versehentliche oder Folgeschäden, die aus Fehlern oder Unterlassungen in diesem Handbuch entstanden, selbst wenn auf die Möglichkeit solcher Schäden hingewiesen wurde. Im Interesse der fortlaufenden Produktentwicklung behält sich der Hersteller jederzeit und ohne vorherige Ankündigung oder Verpflichtung das Recht auf Verbesserungen an diesem Handbuch und der hierin beschriebenen Produkte vor.

> ***Hinweis:*** *Die Angaben dieses Handbuches sind ab den auf der Titelseite angeführten Versionsständen gültig. Überarbeitete Ausgaben dieses Handbuchs sowie Software und Treiber-Updates sind im Servicebereich des myDatanet-Servers erhältlich.*

4.1 Übersetzung

Bei Lieferungen in die Länder des europäischen Wirtschaftsraumes ist das Handbuch in die Sprache des Verwenderlandes zu übersetzen. Sollten im übersetzten Text Unstimmigkeiten auftreten, ist das Original-Handbuch (deutsch) zur Klärung heranzuziehen oder der Hersteller zu kontaktieren.

4.2 Copyright

4.3 Gebrauchsnamen

Die Wiedergabe von Gebrauchsnamen, Handelsnamen, Warenbezeichnungen und dgl. in diesem Handbuch berechtigen nicht zu der Annahme, dass solche Namen ohne weiteres von jedermann benutzt werden dürfen; oft handelt es sich um gesetzlich geschützte eingetragene Warenzeichen, auch wenn sie nicht als solche gekennzeichnet sind.

4.4 Sicherheitshinweise

Für Anschluss, Inbetriebnahme und Betrieb des myDatalogC33x sind die nachfolgenden Informationen und übergeordneten gesetzlichen Bestimmungen des Landes (z.B. ÖVE), wie gültigen Ex-Vorschriften sowie die für den jeweiligen Einzelfall geltenden Sicherheits- und Unfallverhütungsvorschriften zu beachten.

Lesen Sie dieses Handbuch komplett durch, bevor Sie dieses Gerät auspacken, aufstellen oder bedienen. Beachten Sie alle Gefahren-, Warn- und Vorsichtshinweise. Nichtbeachtung kann zu schweren Verletzungen des Bedieners oder Schäden am Gerät führen.

Stellen Sie sicher, dass die Sicherheitseinrichtung dieses Messgerätes nicht beeinträchtigt wird. Verwenden bzw. installieren Sie das Messsystem nur auf solche Art und Weise, wie sie in diesem Handbuch beschrieben wird.

Wichtiger Hinweis: *Das Produkt ist nicht zur Nutzung im Freien freigegeben, da es keinen Schutz gegen das Eindringen von Feuchtigkeit und nur sehr geringen Schutz gegen das Eindringen von Staub besitzt.*

4.4.1 Verwendung der Gefahrenhinweise

GEFAHR:
Kennzeichnet eine mögliche oder drohende Gefahrensituation, die den Tod oder eine ernsthafte Verletzung zur Folge haben wird, wenn sie nicht vermieden wird.

WARNUNG:
Kennzeichnet eine mögliche oder drohende Gefahrensituation, die den Tod oder eine ernsthafte Verletzung zur Folge haben kann, wenn sie nicht vermieden wird.

VORSICHT:
Kennzeichnet eine mögliche Gefahrensituation, die leichte oder mittelschwere Verletzungen oder Schäden an diesem Instrument zur Folge haben kann.

Wichtiger Hinweis: *Kennzeichnet eine Situation, die Schäden an diesem Instrument zur Folge haben kann, wenn sie nicht vermieden wird. Informationen, die besonders hervorgehoben werden müssen.*

Hinweis: *Kennzeichnet eine Situation, die keine Personenschäden zur Folge hat.*

Hinweis: *Informationen, die Angaben im Haupttext ergänzen.*

4.4.2 Allgemeine Sicherheitshinweise

WARNUNG:
Gefährliche elektrische Spannung kann zu elektrischem Schlag und Verbrennungen führen. Schalten Sie immer alle verwendeten Spannungsversorgungen für das Gerät ab, bevor Sie es installieren, Wartungsarbeiten durchführen oder Störungen beheben.

WARNUNG:
Sorgen Sie dafür, dass das Gerät während jeder Versendung/Rücksendung als Luftfracht vollständig deaktiviert ist und sich nicht selbständig wieder aktivieren kann. Informationen dazu finden Sie im Kapitel "Aufbewahrung des Produkts" auf Seite 26. Bei offenen Fragen wenden Sie sich an den Hersteller (siehe "Kontaktinformationen" auf Seite 253).

WARNUNG:
Verwenden Sie dieses Gerät nie in Bereichen, in denen der Betrieb von Funkeinrichtungen untersagt ist. Das Gerät darf nicht in Krankenhäusern und/oder in der Nähe von medizinischen Geräten, wie etwa Herzschrittmachern oder Hörgeräten, betrieben werden, da deren Funktionsweise durch das im Gerät enthaltene Mobilfunkmodem beeinträchtigt werden kann.

WARNUNG:
Verwenden Sie dieses Gerät nie in explosionsgefährdeten Bereichen sowie in der Nähe von hochbrennbaren Bereichen (Tankstellen, Brennstofflagerstätten, Chemiewerken und Sprengstätten) oder in der Nähe von brennbaren Gasen, Dämpfen oder Staub.

4.4.3 Sicherheits-/Vorsichtsmaßnahmen im Umgang mit Mobilfunkmodems

Die folgenden Sicherheits-/Vorsichtsmaßnahmen sind bei allen Phasen des Einbaus, des Betriebs, der Wartung oder der Reparatur eines Mobilfunkmodems zu beachten. Der Hersteller haftet nicht, wenn der Kunde diese Vorsichtsmaßnahmen außer Acht lässt.

VORSICHT:
Die Mobilfunkverbindung darf nicht in gefährlichen Umgebungen verwendet werden.

Der Hersteller und seine Lieferanten übernehmen weder ausdrückliche noch indirekte Garantie für die Verwendung bei Hochrisikoaktivitäten.

Zusätzlich zu den folgenden Sicherheitsbetrachtungen sind alle Richtlinien des Landes zu befolgen, in dem das Gerät installiert wird.

> ***Wichtiger Hinweis:*** *Für die Verbindung mittels Mobilfunkmodem, bei dessen Verwendung Funksignale und -netzwerke zum Einsatz kommen, wird zu keiner Zeit und unter keinen Umständen gehaftet. Das Mobilfunkmodem muss eingeschaltet sein und in einem Gebiet betrieben werden, in dem eine ausreichende Signalstärke vorhanden ist.*

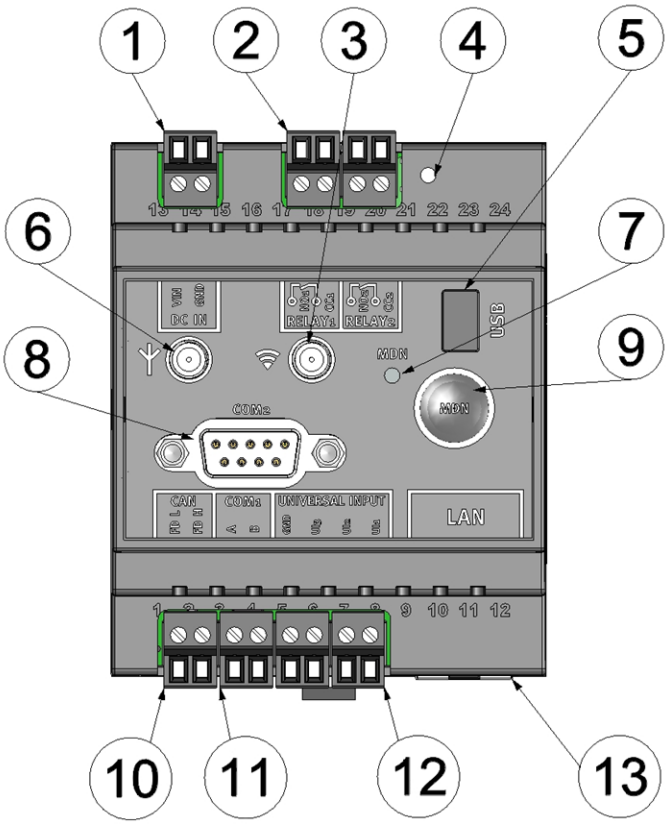
4.4.3.1 Sicherheits-/Vorsichtsmaßnahmen für den Mobilfunkmodemeinbau

- Dieses Gerät darf nur durch einen geschulten Techniker eingebaut werden, der anerkannte Einbaupraktiken für Funkfrequenzsender anwendet, einschließlich der korrekten Erdung von externen Antennen.
- Das Gerät darf nicht in Krankenhäusern und/oder in der Nähe von medizinischen Geräten, wie etwa Herzschrittmachern oder Hörgeräten, betrieben werden.
- Das Gerät darf nicht in der Nähe von hochbrennbaren Bereichen, wie etwa Tankstellen, Brennstofflagerstätten, Chemiewerken und Sprengstätten betrieben werden.
- Das Gerät darf nicht in der Nähe von brennbaren Gasen, Dämpfen oder Staub betrieben werden.
- Das Gerät darf weder starken Vibrationen noch Stößen ausgesetzt werden.
- Das Mobilfunkmodem kann Störungen verursachen, wenn es sich in der Nähe von Fernsehgeräten, Radios oder Computern befindet.
- Das Mobilfunkmodem nicht öffnen. Eine Änderung des Geräts ist unzulässig und führt zum Verlust der Betriebsgenehmigung.
- Die Nutzung von GSM-Diensten (SMS-Nachrichten, Datenkommunikation, GPRS, etc.) führt unter Umständen zu zusätzlichen Kosten. Der Benutzer ist allein verantwortlich für hierdurch erfolgte Schäden und Kosten.
- Bauen Sie das Gerät nicht anders ein, als in der Bedienungsanleitung angegeben. Eine fehlerhafte Verwendung führt zum Erlöschen der Garantie.

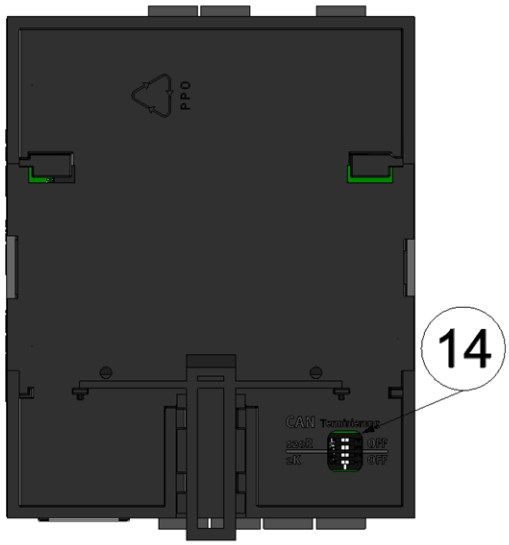
4.4.3.2 Sicherheitsmaßnahmen für den Antenneneinbau

- Nur Antennen verwenden, die vom Hersteller empfohlen oder geliefert werden.
- Die Antenne muss mindestens im Abstand von 20cm zu Personen aufgestellt werden.
- Die Antenne darf nicht außerhalb von geschützten Gebäuden aufsteigen und muss gegen Blitzschläge geschützt werden!
- Die Spannungsversorgung muss abgestellt werden, bevor eine Antenne ausgetauscht wird.

4.5 Übersicht



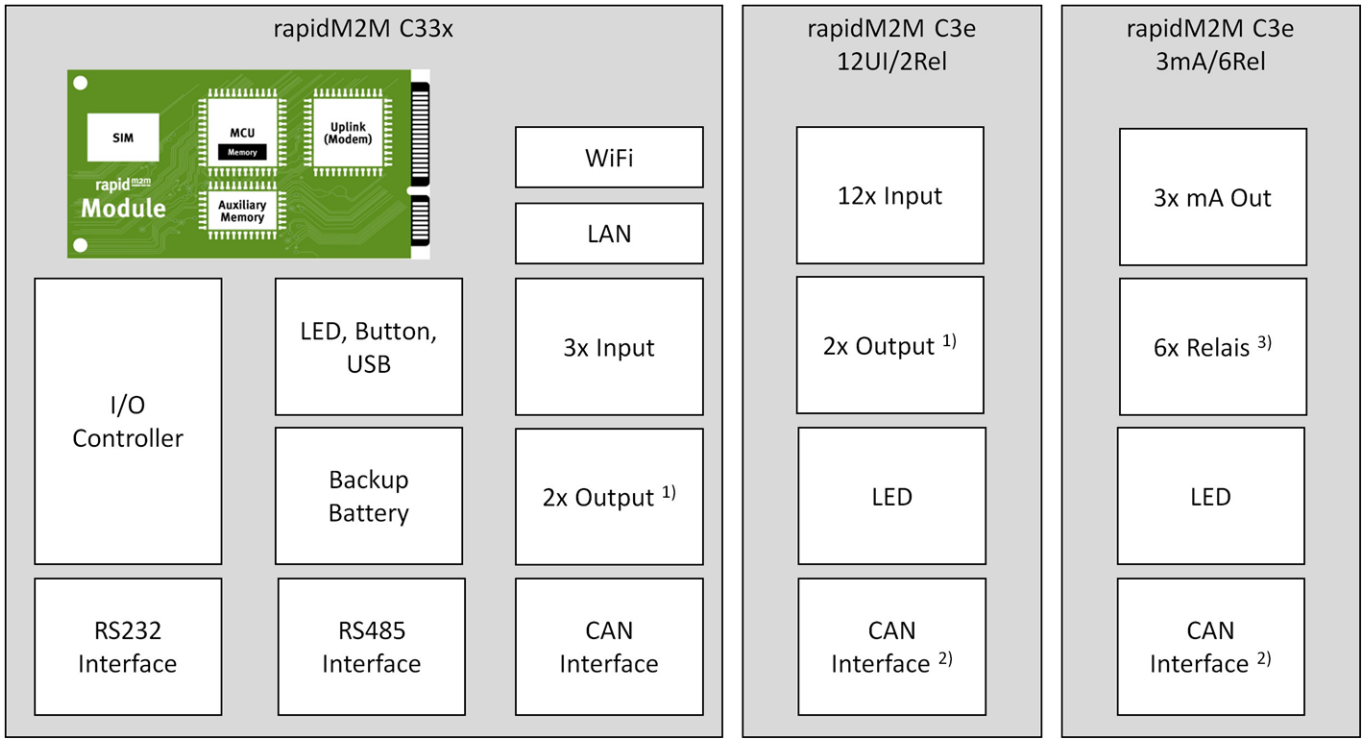
Vorderseite des myDatalogC33x



Rückseite des myDatalogC33x

1	Versorgung (V IN, GND)	**8**	Com2 (RS232)
2	Relais 1-2	**9**	MDN Taste (frei verwendbar, Auswertung durch die Device Logic)
3	Anschluss für die WiFi-Antenne	**10**	CAN / CAN FD
4	Reset Taster	**11**	Com1 (RS485)
5	Mini-B USB (nur für Debug und Device Logic Update)	**12**	Universaleingang 1-3 (inkl. GND)
6	Anschluss für die Mobilfunkantenne	**13**	LAN-Schnittstelle
7	RGB-LED (frei verwendbar, gesteuert durch die Device Logic)	**14**	Dip-Switch zum Ein-/Ausschalten der Abschlusswiderstände für die CAN-Schnittstelle

4.5.1 Systemarchitektur



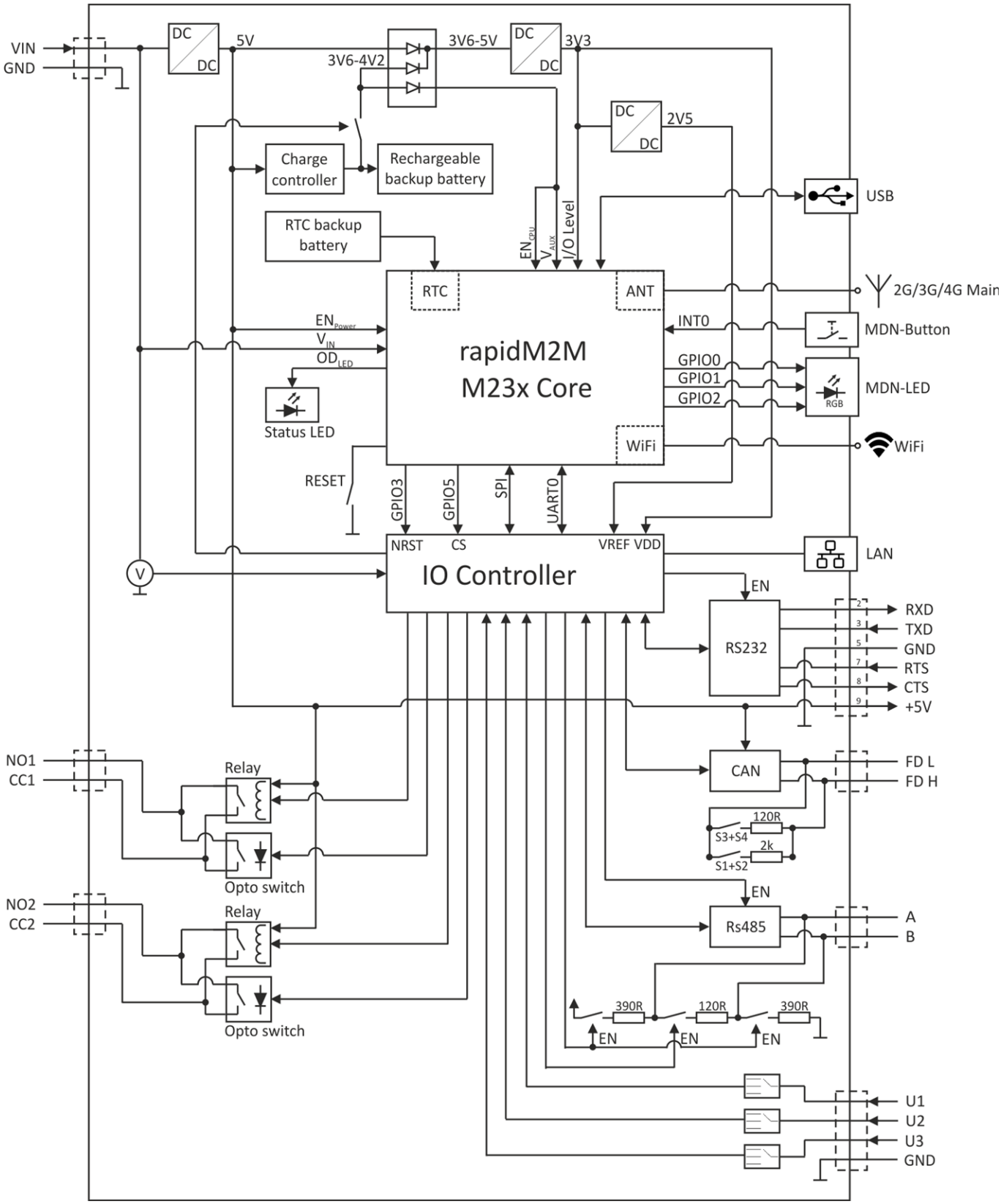
Systemarchitektur des myDatalogC33x und der Erweiterungsmodule

1) Parallelschaltung eines Relais (hohe Schaltströme) und eines Optoswitchs (hohe Schaltfrequenzen)

2) Ausschließlich für die Kommunikation mit dem myDatalogC33x bestimmt

3) Je 3 der Relais sind zu einer Gruppe mit gemeinsamer Wurzel zusammengefasst

4.5.2 Blockschaltbild



Blockschaltbild des myDatalogC33x

4.6 Bestimmungsgemäße Verwendung

Das stationäre, kompakte, frei programmierbare Messgerät dient der Ermittlung, Verarbeitung und Übertragung von Messdaten, die über diverse Industrieschnittstellen erfasst werden. Das Gerät benötigt eine permanente Energieversorgung. Die Speicherung der gemessenen und erfassten Daten erfolgt auf einem nicht flüchtigen Speichermedium. Diese gespeicherten Daten werden über das Mobilfunknetz, WiFi oder LAN an einen zentralen Server zur Weiterverarbeitung gesendet. Für die Herstellung der Mobilfunkverbindung ist das Gerät mit einem integrierten SIM-Chip versehen. Es sind die zulässigen maximalen Grenzwerte, aufgeführt im Kapitel "Technische Daten" auf Seite 13, unbedingt zu beachten. Sämtliche von diesen Grenzwerten abweichende Einsatzfälle, die nicht vom Hersteller in schriftlicher Form freigegeben sind, entfallen aus der Haftung des Herstellers.

> ***Hinweis:*** *Das Gerät ist ausschließlich zum vorab angeführten Zweck bestimmt. Eine andere, darüber hinausgehende Benutzung oder ein Umbau des Geräts ohne schriftliche Absprache mit dem Hersteller, gilt als nicht bestimmungsgemäß. Für jegliche hieraus resultierende Schäden und daraus resultierende Folgeschäden haftet der Hersteller nicht. Das Risiko trägt allein der Betreiber.*
>
> ***Hinweis:*** *Der Hersteller haftet nicht für Datenverluste jeglicher Art.*
>
> ***Hinweis:*** *Der integrierte SIM-Chip gewährleistet eine Mobilfunkverbindung über eine Vielzahl internationaler Serviceprovider. Um alle Funktionen des Geräts nutzen zu können, muss gewährleistet sein, dass es sich im Versorgungsbereich eines dieser Anbieter befindet. Eine Liste aller unterstützten Länder und dazugehörigen Serviceprovider finden Sie unter www.microtronics.com/footprint. Für die Nutzung der mobilen Datenübertragung ist ein "Managed Service"-Vertrag mit der Firma Microtronics Engineering GmbH erforderlich (siehe www.microtronics.com/managedservice). Dieser beinhaltet die Bereitstellung der Mobilfunkverbindung über die Netze der in der oben genannten Liste enthaltenen Serviceprovider.*

4.7 Allgemeine Produktinformationen

Es handelt sich um ein kompaktes, stationäres, frei programmierbares Gerät, das neben der Ermittlung, Verarbeitung und Übertragung von Messdaten, auch noch für verschiedenste Steuer- und Regelaufgaben verwendet werden kann.

Für die Messdatenerfassung stehen folgende Schnittstellen zur Verfügung:

- 3 x Universaleingänge, die in verschiedenen Analog- oder Digitalmodi betrieben werden können
- 1 x RS485 Schnittstelle
- 1 x CAN / CAN FD Schnittstelle (nicht galvanisch getrennt)
- 1 x RS232 Schnittstelle
- WiFi-Schnittstelle (sofern nicht als Uplink eingesetzt)

Des Weiteren stehen für die Ausgabe von Regel- und Steuerbefehlen 2 potentialfreie Schaltkontakte zur Verfügung. Bei den potentialfreien Schaltkontakten handelt es sich um eine Parallelschaltung eines Relais (hohe Schaltströme) und eines Optoswitchs (hohe Schaltfrequenzen). Der integrierte Pufferakku gestattet es bei Ausfall der Versorgung eine angemessene applikative Reaktion zu realisieren bzw. ermöglicht ein korrektes Ausbuchen aus dem Mobilfunknetz. Dank der Hardware Real-Time Clock mit eigenständiger Pufferbatterie läuft die Systemzeit auch im ausgeschalteten Zustand weiter, wodurch bei der Wiederinbetriebnahme sofort eine gültige Zeitbasis zur Verfügung steht.

Der Benutzer kann mittels rapidM2M Studio (siehe "rapidM2M Studio " auf Seite 101) seine eigene Applikation erstellen. Während der Entwicklung wird jener Teil der Applikation, der am Gerät installiert

werden muss (d.h. die Device Logic), über die USB-Schnittstelle in den myDatalogC33x geladen. Bei Applikationen, die über den rapidM2M Store angeboten werden, erfolgt die Installation der Device Logic über die Mobilfunk-, WiFi- oder LAN-Verbindung im Zuge der Verknüpfung der Site mit dem myDatalogC33x . Die Device Logic ermöglicht den Zugriff auf die seriellen Schnittstellen (RS232, CAN / CAN FD und RS485), wodurch der Benutzer die Möglichkeit erhält, nahezu alle mit diesen Schnittstellen kompatiblen Geräte und Sensoren anzubinden und die entsprechenden Kommunikationsprotokolle zu implementieren.

Der myDatalogC33x bietet dem Benutzer einen Speicherbereich für seine Daten (3 MB) sowie 10 voneinander unabhängige Speicherblöcke für Konfigurationsdaten mit je 4000 Bytes . Zudem verfügt der myDatalogC33x neben 4 Registrierungsspeicherblöcken mit je 1kB , die im Flash gespeichert werden, noch über einen weiteren der mittels der Funktion "rM2M_RegInit()" optional initialisiert werden kann und im RAM abgelegt wird. Dessen Größe kann bei der Initialisierung angegeben werden, ist allerdings auf maximal 1kB begrenzt. Die Registrierungsspeicherblöcke sind vordefinierten Verwendungszwecken zugewiesen und dienen zur Ablage gerätespezifischer Daten (siehe "Registrierungsspeicherblöcke" auf Seite 36).

Erfasste Daten können über das Mobilfunknetz, WiFi oder LAN an einen zentralen myDatanet-Server zur Weiterverarbeitung gesendet werden. Für die Herstellung der Mobilfunkverbindung ist das Gerät mit einem integrierten SIM-Chip versehen. Für das Anstoßen des Verbindungsaufbaus zum zentralen myDatanet-Server ist der Benutzer selbst verantwortlich (siehe "rM2M_TxStart()"). Die Synchronisation der Konfigurations-, Registrierungs- und Messdaten mit dem Server hingegen wird vom System selbständig durchgeführt.

4.8 Gerätekennzeichnung

Die Angaben in diesem Handbuch gelten ausschließlich für den Gerätetyp myDatalogC33x . Das Typenschild befindet sich auf der rechten Seite des Geräts und beinhaltet folgende Angaben:

- Typenbezeichnung
- Seriennummer
- MAC-Adresse der LAN-Schnittstelle
- MAC-Adresse der WiFi-Schnittstelle
- Artikelnummer
- Angaben zur Spannungsversorgung
- Produktionswoche und Produktionsjahr
- Länderlisten-Profil des SIM-Chips
- Umgebungsbedingungen im Betrieb
- Schutzart
- Hardwarerevision
- Name und Anschrift des Herstellers
- Logo zur WEEE-Direktive der EU
- CE-Kennzeichnung

Wichtig für alle Rückfragen und Ersatzteilbestellungen ist die richtige Angabe der Typenbezeichnung und der Seriennummer. Nur so ist eine einwandfreie und schnelle Bearbeitung möglich.

Typenschild myDatalogC33x

Hinweis: *Dieses Symbol gibt das Länderlisten-Profil (siehe www.microtronics.com/footprint) des im Gerät verbauten SIM-Chips an.*

Hinweis: *Diese Betriebsanleitung ist Bestandteil des Gerätes und muss für den Benutzer jederzeit zur Verfügung stehen. Die darin enthaltenen Sicherheitshinweise sind zu beachten.*

WARNUNG:
Es ist strengstens untersagt, die Sicherheitseinrichtungen außer Kraft zu setzen oder in ihrer Wirkungsweise zu verändern.

4.9 Einbau von Ersatz- und Verschleißteilen

Es wird an dieser Stelle ausdrücklich darauf aufmerksam gemacht, dass Ersatz- und Zubehörteile, die nicht vom Hersteller geliefert wurden, auch nicht vom Hersteller geprüft und freigegeben wurden. Der Einbau und/oder die Verwendung solcher Produkte können u. U. konstruktiv vorgegebene Eigenschaften des Geräts negativ verändern. Für sämtliche Schäden, die durch die Verwendung von Nicht-Originalteilen und Nicht-Original-Zubehörteilen entstehen, ist die Haftung des Herstellers ausgeschlossen.

4.10 Aufbewahrung des Produkts

Zur Aufbewahrung des myDatalogC33x stellen Sie sicher, dass alle relevanten Daten zum myDatanet-Server übertragen wurden. Gegebenenfalls lösen Sie dazu direkt am Gerät, falls Sie dies in Ihrer Device Logic vorgesehen haben, mittels MDN-Taste eine Übertragung aus und kontrollieren Sie anschließend erneut, ob nun alle relevanten Daten übertragen wurden. Haben Sie an Ihrem Gerät keine Möglichkeit zum Auslösen einer Übertragung der zwischengespeicherten Messdaten vorgesehen, müssen Sie unter Umständen bis zur nächsten planmäßigen Datenübertragung warten, bis alle Daten zum myDatanet–Server gesendet wurden. Dies gilt speziell für die Verbindungsart „Intervall“ (siehe "rM2M_TxSetMode()"). Wurde die Verbindungsart „Intervall & Wakeup“ ausgewählt, können Sie die Übertragung über den myDatanet–Server auslösen. Kontrollieren Sie anschließend erneut, ob nun alle relevanten Daten übertragen wurden. Bei der Verbindungsart „online“ werden die ermittelten Messdaten sofort zum myDatanet–Server übertragen. Die Daten am Server sind immer aktuell und das Gerät kann jederzeit abgeschaltet werden. Als nächstes trennen Sie das Gerät von der Versorgungsspannung. Wenn möglich, sollte dabei die Versorgungsspannung zuerst abgeschaltet werden bevor Sie die Kabel an den Klemmen V IN und GND (siehe "Anschluss der Sensoren, der Aktoren und der Versorgung" auf Seite 46) lösen. Anschließend können die restliche Verkabelung und die Antenne entfernt werden. Stellen Sie sicher, dass der myDatalogC33x vollständig deaktiviert ist, bevor Sie ihn in der Originalverpackung aufbewahren. Drücken Sie dazu die Reset-Taste direkt am Gerät, falls Sie in Ihrer Device Logic keine Routine für das kontrollierte Herunterfahren des Systems nach dem Trennen der Versorgungsspannung vorgesehen haben.

Die Konfiguration und die zuletzt ermittelten Daten bleiben erhalten. Auch die Systemzeit läuft dank der Hardware Real-Time Clock mit eigenständiger Pufferbatterie weiter. Bei der Wiederinbetriebnahme steht somit sofort eine gültige Zeitbasis zur Verfügung (siehe "Technische Details zur Systemzeit" auf Seite 67).

4.11 Gewährleistung

Das Gerät wurde vor Auslieferung funktional geprüft. Bei bestimmungsgemäßer Verwendung (siehe "Bestimmungsgemäße Verwendung" auf Seite 23) und Beachtung der Bedienungsanleitung, der mitgeltenden Unterlagen (siehe "Mitgeltende Unterlagen" auf Seite 69) und der darin enthaltenen Sicherheitshinweise und Anweisungen sind keine funktionalen Einschränkungen zu erwarten und ein einwandfreier Betrieb sollte möglich sein.

Hinweis: *Beachten Sie hierzu auch das nachfolgende Kapitel "Haftungsausschluss" auf Seite 27.*

Hinweis: Einschränkung der Gewährleistung

Bei Nichtbeachtung der Sicherheitshinweise und Anweisungen in dieser Unterlage behält sich der Hersteller eine Einschränkung der Gewährleistung vor.

4.12 Haftungsausschluss

Der Hersteller übernimmt keine Haftung

- für Folgeschäden, die auf **eine Änderung** dieses Dokumentes zurückzuführen sind. Der Hersteller behält sich das Recht vor, den Inhalt des Dokuments einschließlich dieses Haftungsausschlusses unangekündigt zu ändern.
- für Personen- oder Sachschäden, die auf eine **Missachtung** der gültigen Vorschriften zurückzuführen sind. Für Anschluss, Inbetriebnahme und Betrieb der Geräte/Sensoren sind alle Informationen und übergeordneten gesetzlichen Bestimmungen des Landes (in Österreich z. B. die ÖVE-Richtlienien), wie gültige Ex-Vorschriften sowie die für den jeweiligen Einzelfall geltenden Sicherheits- und Unfallverhütungsvorschriften zu beachten.
- für Personen- oder Sachschäden, die auf **unsachgemäße Handhabung** zurückzuführen sind. Sämtliche Handhabungen am Gerät, welche über die montage- und anschlussbedingten Maßnahmen hinausgehen, dürfen aus Sicherheits- und Gewährleistungsgründen prinzipiell nur von Microtronics - Personal bzw. durch Microtronics autorisierte Personen oder Firmen vorgenommen werden.
- für Personen- oder Sachschäden, die auf den Betrieb des Geräts in technisch **nicht einwandfreiem** Zustand zurückzuführen sind.
- für Personen- oder Sachschäden, die auf eine **nicht bestimmungsgemäße Verwendung** zurückzuführen sind.
- für Personen- oder Sachschäden, die auf eine **Missachtung** der **Sicherheitshinweise** in dieser Anleitung zurückzuführen sind.
- für fehlende oder falsche Messwerte, die auf **unsachgemäße Installation** zurückzuführen sind und für die daraus resultierenden Folgeschäden.

4.13 Pflichten des Betreibers

WARNUNG:
Im EWR (Europäischer Wirtschaftsraum) sind die nationale Umsetzung der Rahmenrichtlinie (89/391/EWG) sowie die dazugehörigen Einzelrichtlinien und davon besonders die Richtlinie (2009/104/EG) über die Mindestvorschriften für Sicherheit und Gesundheitsschutz bei Benutzung von Arbeitsmitteln durch Arbeitnehmer bei der Arbeit, jeweils in der gültigen Fassung, zu beachten und einzuhalten.

Der Betreiber muss die örtliche Betriebserlaubnis einholen und die damit verbundenen Auflagen beachten.

Zusätzlich muss er die örtlichen gesetzlichen Bestimmungen für

- die Sicherheit des Personals (Unfallverhütungsvorschriften),
- die Sicherheit der Arbeitsmittel (Schutzausrüstung und Wartung),
- die Produktentsorgung (Abfallgesetz),
- die Materialentsorgung (Abfallgesetz),
- die Reinigung (Reinigungsmittel und Entsorgung) und
- die Umweltschutzauflagen einhalten.

Vor dem Betreiben des Messgeräts ist vom Betreiber sicherzustellen, dass bei der Montage und Inbetriebnahme, wenn diese vom Betreiber selbst durchgeführt werden, die örtlichen Vorschriften beachtet werden.

4.14 Anforderungen an das Personal

Die Installation, Inbetriebnahme und Wartung dürfen nur durch Personal durchgeführt werden, das die folgenden Bedingungen erfüllt:

- Qualifiziertes Fachpersonal mit entsprechender Ausbildung
- Autorisierung durch den Anlagenbetreiber

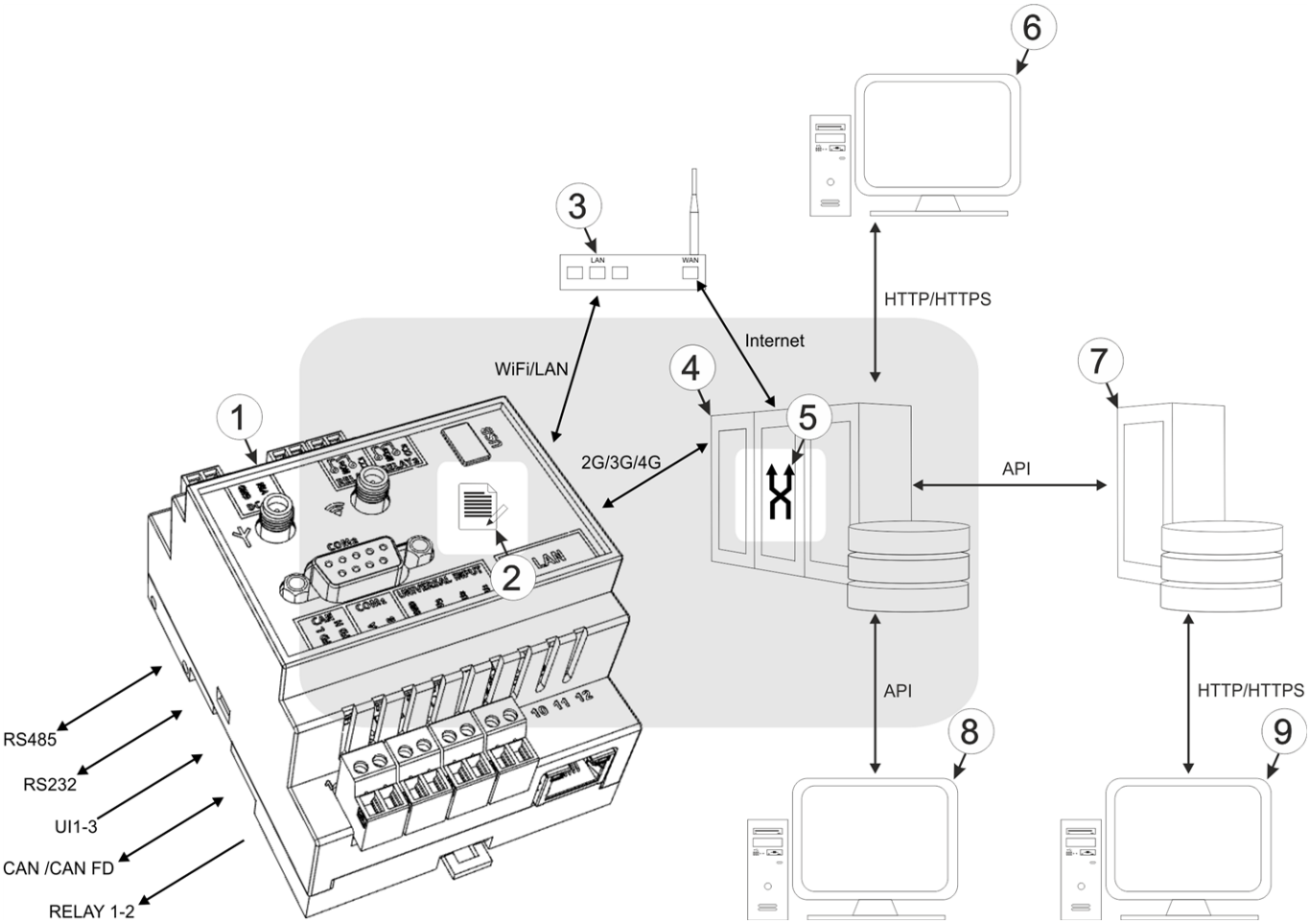
Hinweis: Qualifiziertes Fachpersonal

Im Sinne dieser Anleitung bzw. Warnhinweise auf dem Produkt selbst sind dies Personen, die mit Aufstellung, Montage, Inbetriebnahme und Betrieb des Produktes vertraut sind und über die ihrer Tätigkeit entsprechenden Qualifikationen verfügen, wie z.B.

- *Ausbildung und Unterweisung bzw. Berechtigung, Stromkreise und Geräte/Systeme gemäß den Standards der Sicherheitstechnik ein- und auszuschalten, zu erden und zu kennzeichnen*
- *Ausbildung oder Unterweisungen gemäß den Standards der Sicherheitstechnik in Pflege und Gebrauch angemessener Sicherheitsausrüstung*
- *Schulung in Erster Hilfe*

Kapitel 5 Funktionsprinzip

In der unten abgebildeten Grafik sind alle Komponenten, die Teil des myDatanet sind, grau hinterlegt. Alle anderen Komponenten müssen vom Kunden bereitgestellt/erstellt werden.



Funktionsprinzip

1	myDatalogC33x mit integriertem Managed Service SIM-Chip (Datenübertragung inkludiert)
2	durch den Kunden erstellte Applikation (Device Logic), die sich um die Datenerfassung und -aufzeichnung kümmert (siehe"Device Logic" auf Seite 107)
4	myDatanet-Server, zu dem die Daten übertragen werden
5	Durch den Kunden definierter Data Descriptor , der die Nutzung der durch die Applikation (Device Logic) erzeugten Messdaten und Konfigurationen in Verbindung mit der Oberfläche des myDatanet-Servers ermöglicht (siehe "Data Descriptor " auf Seite 213)
6	Client, der mittels Web-Browser auf die Oberfläche des myDatanet-Servers zugreift
7	kundenspezifischer Server, der den Clients eine eigene Oberfläche zur Verfügung stellt. Die Daten bezieht der kundenspezifische Server über die API-Schnittstelle des myDatanet-Servers (siehe "API" auf Seite 225).
8	Client, auf dem ein PC-Programm läuft, das seine Daten über die API-Schnittstelle des myDatanet-Servers (siehe "API" auf Seite 225) bezieht
9	Client, der mittels Web-Browser auf die Oberfläche des kundenspezifischen Servers zugreift

Funktionen und Komponenten, die durch myDatanet bereitgestellt werden:

- myDatalogC33x

  Programmierbares (siehe "Device Logic" auf Seite 107), stationäres Gerät mit integriertem Speicher und standardisierten Industrieschnittstellen (UI1-3, RS485, CAN / CAN FD, RS232, potentialfreier Schaltkontakt 1-2) zur Anbindung von Maschinen, Sensoren und Aktoren an den myDatanet-Server (2G/3G/4G/WiFi/LAN)

- Managed Service

  Das Managed Service ist die Basis für den Betrieb der Geräte und bietet eine breite Palette an Services. Managed Service inkludiert Updates für Geräte-Firmware, mobile Datenübertragung auf globaler Ebene sowie gebührenfreien Support - ein Ansprechpartner für die gesamte Lösung.

- myDatanet-Server

  Datenbank für die Speicherung der Messdaten und Konfigurationen. Der Zugriff auf die Daten erfolgt entweder über die API (siehe "API" auf Seite 225) oder die Web-Oberfläche des Servers.

Funktionen und Komponenten, die durch den Kunden bereitgestellt werden

- Maschinen, Sensoren oder Aktoren

  Maschinen, Sensoren oder Aktoren die über Schnittstellen verfügen, die mit dem im Kapitel "Technische Daten" (siehe "Technische Daten" auf Seite 13) aufgelisteten Spezifikationen kompatibel sind

- Applikation (Device Logic)

  Die Firmeware des myDatalogC33x kümmert sich lediglich um die Synchronisation der Messdaten und Konfigurationen zwischen myDatalogC33x und myDatanet-Server. Die vom Kunden erstellte Applikation muss sich um die Messwerterfassung und den Aufbau der zu speichernden Datenblöcke kümmern. Die Ablage der Datenblöcke erfolgt wiederum von der Firmware des myDatalogC33x (siehe "rM2M_RecData()"). Der Zeitpunkt der Synchronisation und die Art der Verbindung müssen ebenfalls durch die vom Kunden erstellte Applikation festgelegt werden. Dazu stehen Ihnen die beiden API-Funktionen "rM2M_TxStart()" und "rM2M_TxSetMode()" zur Verfügung.

- Data Descriptor

  Die Basisfunktion des myDatanet-Servers beschränkt sich auf die Synchronisation der Messdatenkanäle ("histdata0" - "histdata9") und Konfigurationsblöcke ("config0" - "config9") zwischen myDatalogC33x und Server. Der vom Kunden definierte Data Descriptor muss sich um die Aufgliederung der Messdatenkanäle und Konfigurationsblöcke in die einzelnen Datenfelder kümmern. Eine Erklärung dazu finden Sie im Kapitel "Datenstruktur" auf Seite 213.

- Router zum Anbinden des myDatalogC33x an das Internet (optional)

  Dadurch steht für die Verbindung zum myDatanet-Server neben der Mobilfunkverbindung auch die WiFi- oder LAN-Verbindung als Kommunikationskanal zur Verfügung.

- Kundenspezifischer Server mit Web-Oberfläche für die Clients (optional)

  Mit dessen Hilfe ist es möglich eine eigene Web-Oberfläche für die Clients zu erstellen. Die Daten werden dabei vom kundenspezifischen Server über die API-Schnittstelle (siehe "API" auf Seite 225) vom myDatanet-Server gelesen.

5.1 Empfohlene Vorgehensweise

5.1.1 Entwicklung einer M2M / IoT Anwendung

Es wird empfohlen bei der Entwicklung einer M2M / IoT Anwendung mit der Definition des Data Descriptor (siehe "Data Descriptor " auf Seite 213) zu beginnen. Durch ihn werden die verschiedenen Datenstrukturen (Messdaten, Konfigurationen usw.) festgelegt, die sowohl für die Device Logic als auch für den myDatanet-Server gültig sind. Für den Zugriff auf die Daten des myDatanet-Servers über die API gelten ebenfalls die Definitionen des Data Descriptor .

Bei der Zuweisung der Daten zu den jeweiligen Containern ("histdata0" - "histdata9" bzw. "config0" - "config9") sollte der Informationstyp berücksichtigt werden. Für Zeitreihen sollten die Container "histdata0" - "histdata9" verwendet werden. Sollte es sowohl Messdaten geben, die häufig erzeugt werden als auch welche, die nur selten generiert werden, empfiehlt es sich zwei unterschiedliche Container (z.B. "histdata0" für die häufig erzeugten und "histdata1" für die selten generierten) zu verwenden. Ähnliches gilt auch für die Konfigurationsdaten, für die die Container "config0" - "config9" vorgesehen sind. Bei den Konfigurationsdaten empfiehlt sich neben der Berücksichtigung der Häufigkeit der Änderung auch noch eine Gruppierung nach logischen Zusammenhängen.

> ***Hinweis:*** *Eine wohl durchdachte Wahl der Container hilft dabei das benötigte Datenvolumen und die damit verbundenen Kosten gering zu halten.*

5.2 Funktionsweise des internen Datenspeichers

Struktur	Ringspeicher
Gesamtgröße	3 MB
Anzahl der Sektoren	8
Sektorgröße	393.216 Byte

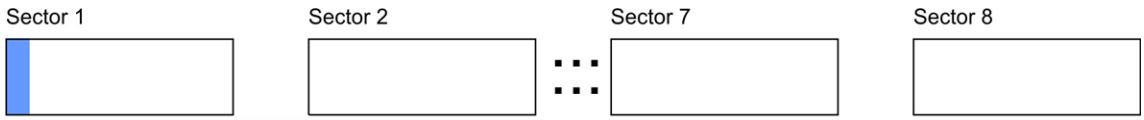
Der interne Datenspeicher des myDatalogC33x ist als Ringspeicher mit 8 Sektoren aufgebaut. Wurde der gesamte Speicher (3 MB) zur Gänze beschrieben, wird der Sektor mit den ältesten Daten vollständig gelöscht, bevor wieder neue Daten in diesen Sektor gespeichert werden können. D.h. der interne Datenspeicher umfasst zumindest 2,625 MB an gültigen Daten, maximal aber 3 MB .

Aus diesem Grund empfiehlt es sich die Datenübertragung und -aufzeichnung so aufeinander abzustimmen, dass zwischen zwei Übertragungen maximal 2,625 MB aufgezeichnet werden müssen. Ist zu erwarten, dass aufgrund einer schlechten Netzabdeckung einzelne Übertragungen ausfallen, muss auch dies bei der Berechnung der zu speichernden Datenmenge berücksichtigt werden. Des Weiteren gilt es auch zu beachten, dass der systembedingte Overhead pro Datensatz 11 Byte beträgt und die ersten 8 Byte eines jeden Sektors für die interne Verwaltung reserviert sind. Die 11 Byte Overhead beinhalten bereits den Zeitstempel, sodass dieser bei der Berechnung der Größe des kompletten Datensatzes nicht mehr berücksichtigt werden muss. Reicht der freie Platz in einem Sektor nicht mehr aus, um den vollständigen Datensatz zu speichern, wird der Datensatz in den nächsten Sektor geschrieben. D.h. ein Datensatz wird nicht über Sektorgrenzen hinweg geschrieben.

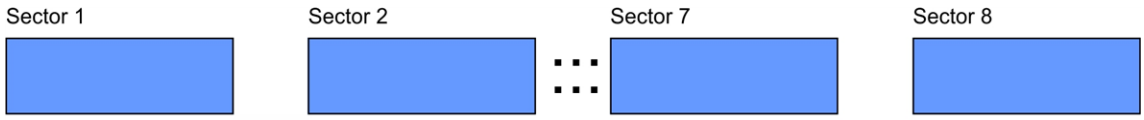
Hinweis:

Ergänzende Erklärung zur Funktionsweise des Ringspeichers

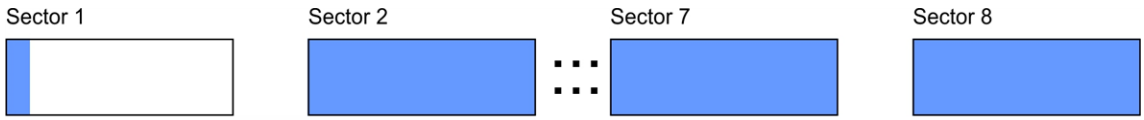
Datenspeicher nach der ersten Datenaufzeichnung:



Datenspeicher nachdem 3 MB aufgezeichnet wurden



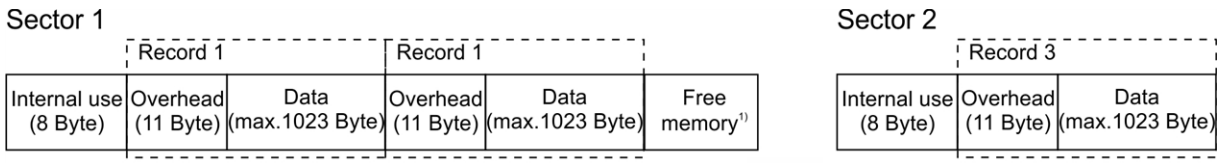
Datenspeicher, wenn nachdem bereits 3 MB aufgezeichnet wurden eine weitere Datenaufzeichnung erfolgte



Hinweis:

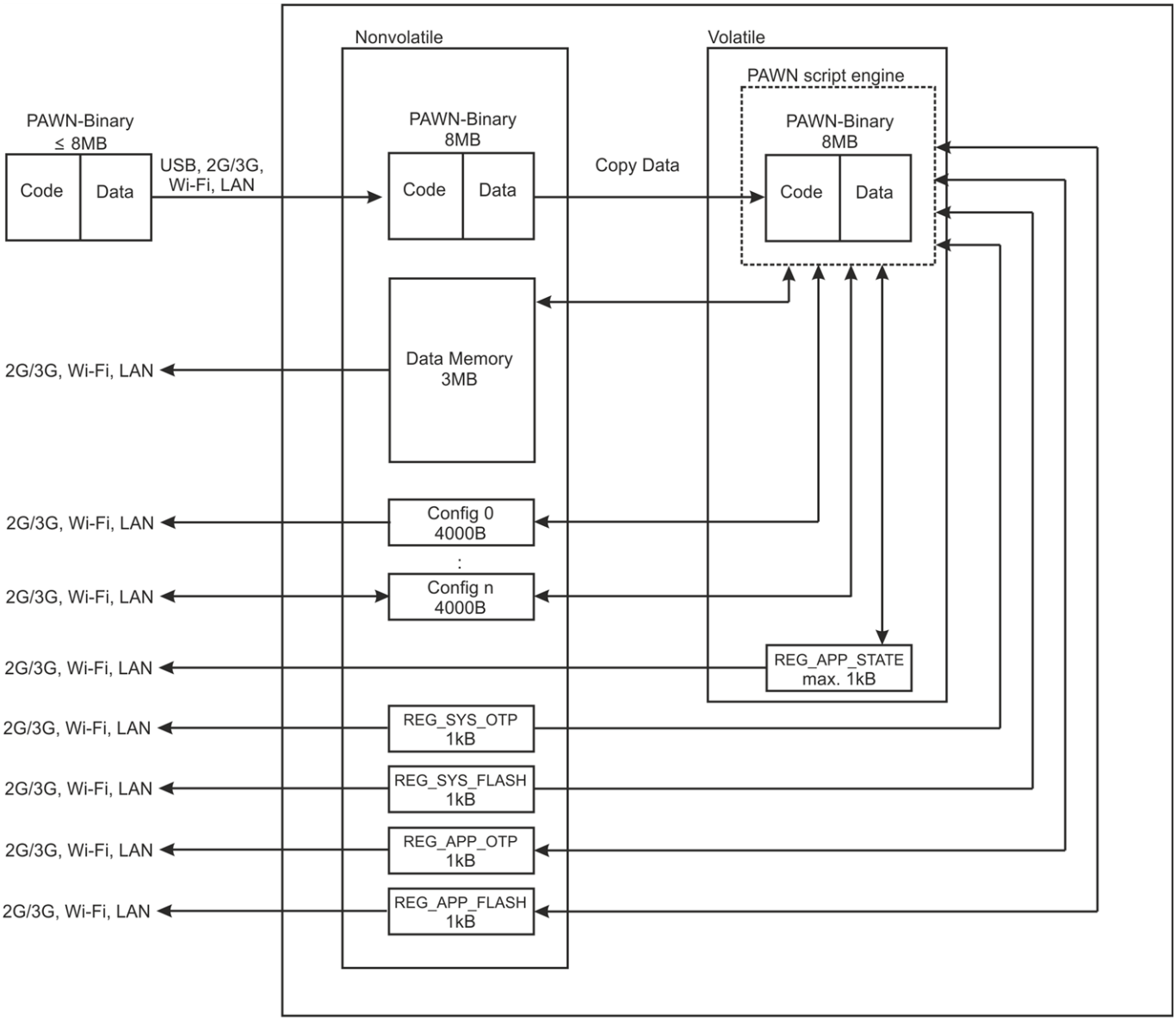
Ergänzende Erklärung zur Berechnung der zu speichernden Datenmenge:

In dem folgenden Beispiel wurde zur Vereinfachung und Verdeutlichung der Darstellung angenommen, dass die Sektoren nur 2 vollständige Datensätze aufnehmen können.



1) Freier Speicher im Sektor 1, der nicht mehr ausreicht, um einen vollständigen Datensatz (Overhead + Daten) aufzunehmen

5.3 Speicherorganisation



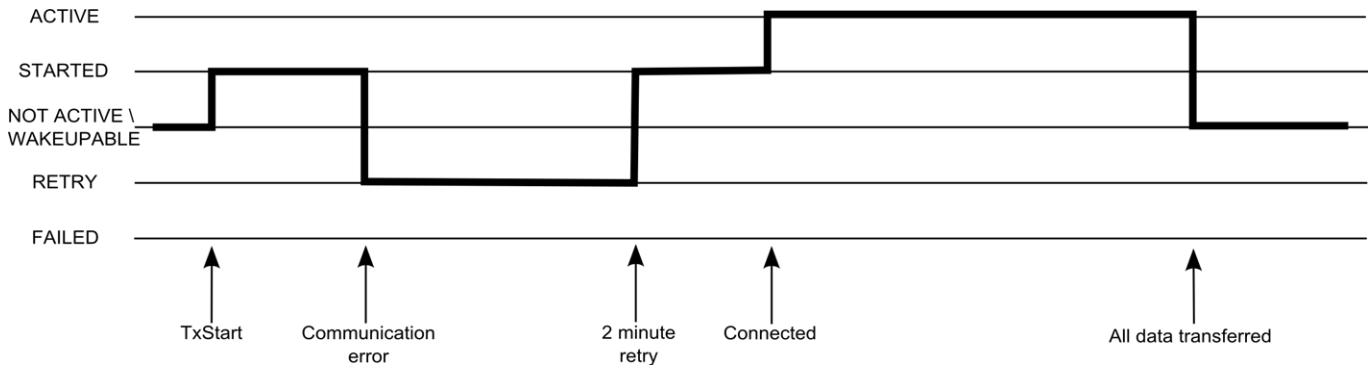
Organisation des Speichers des myDatalogC33x

[1] Dieser Speicherblock ist nur verfügbar, wenn er mittels der Funktion "rM2M_RegInit()" initialisiert wurde.

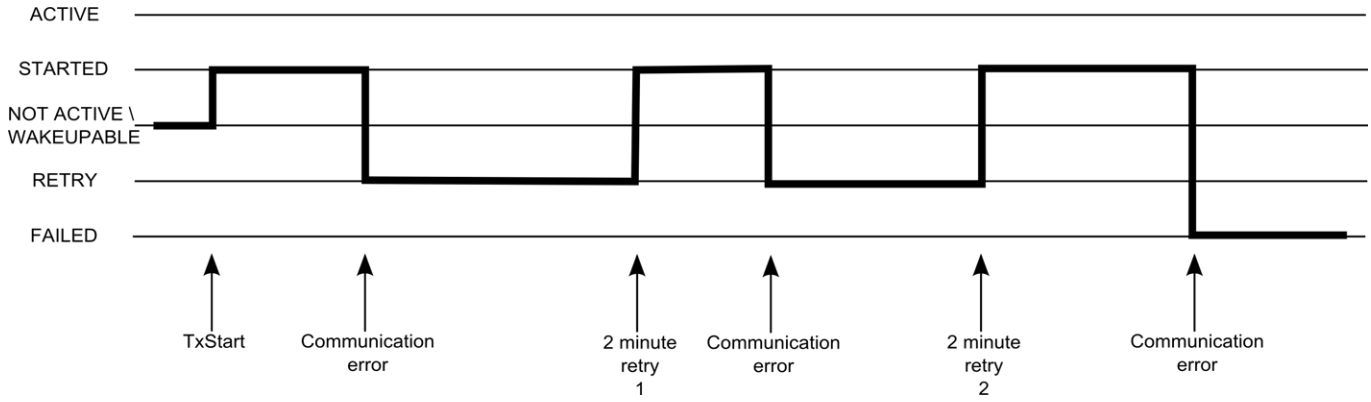
Die dekomprimierte Größe des PAWN-Binarys, das in ein myDatalogC33x geladen werden soll, darf 8MB nicht überschreiten. Um die Datenmenge bei der Übertragung des PAWN-Binarys ins myDatalogC33x zu reduzieren, kann das PAWN-Binary mittels Compiler-Anweisung (#pragma amxcompress <0-3>) komprimiert werden. Im Flash Speicher des myDatalogC33x sind das PAWN-Binary (bis zu 8MB), die 10 Konfigurationsblöcke (je 4000 Byte), die 4 Registrierungsspeicherblöcke (je 1kB) und die Messdaten (3 MB) abgelegt. Für die Ausführung durch die PAWN Script-Engine wird das PAWN-Binary ins RAM kopiert und falls nötig dekomprimiert. Im RAM werden auch die optional mittels der Funktion "rM2M_RegInit()" initialisierbaren Registrierungsspeicherblöcke (z.B. REG_APP_STATE) abgelegt.

5.4 Vorgehensweise bei Verbindungsabbrüchen

Für alle Verbindungen mit Ausnahme des "online"-Modus gilt, dass bei einem Abbruch der Verbindung nach 2min. erneut versucht wird, die Verbindung herzustellen. Der erneute Verbindungsaufbau erfolgt bis zu 2 Mal.



Verbindung konnte beim ersten Retry aufgebaut werden.



Verbindung konnte trotz 2 Retries nicht aufgebaut werden.

ACTIVE

Verbindung zum myDatanet-Server wurde aufgebaut. Die Daten werden übertragen.

STARTED

Verbindungsaufbau wurde ausgelöst.

NOT ACTIVE

Das System wartet auf das Auslösen des nächsten Verbindungsaufbaus. Der letzte Verbindungsversuch war erfolgreich und es konnten alle Daten übertragen werden.

WAKEUPABLE

Das Modem ist ins GSM-Netz eingebucht und das System wartet auf das Auslösen des nächsten Verbindungsaufbaus. Der letzte Verbindungsversuch war erfolgreich und es konnten alle Daten übertragen werden. Da das Modem ins GSM-Netz eingebucht ist, kann der Verbindungsaufbau auch durch den myDatanet-Server ausgelöst werden (siehe "Benutzerhandbuch für myDatanet-Server " 206.886).

RETRY

Das System warten 2min. bis zum erneuten Verbindungsaufbauversuch.

FAILED

Das System wartet auf das Auslösen des nächsten Verbindungsaufbaus. Beim letzten Verbindungsversuch konnten keine Daten bzw. nicht alle Daten übertragen werden.

> ***Hinweis:*** *Abhängig von der Art des Kommunikationsfehlers wird unter Umständen das Gerät neu gestartet (z.B. zum neu Initialisieren des SIM-Chips) bevor der Status "FAILED" gesetzt wird.*

Der aktuelle Verbindungsstatus kann jederzeit durch die Funktion "rM2M_TxGetStatus()" ausgelesen werden.

5.4.1 Verbindungsabbruch im "online"-Modus

Bricht die Verbindung im "online"-Modus ab, erfolgt einmalig ein unmittelbarer Verbindungsaufbauversuch. Ist es dabei nicht möglich eine Verbindung herzustellen, folgt die Standard Retry-Sequenz mit 2 weiteren Versuchen im Abstand von 2min. . Konnte die Verbindung auch beim letzten Retry nicht hergestellt werden, bleibt das Gerät bis zum nächsten Auslösen des Verbindungsaufbaus mittels der Funktion "rM2M_TxStart()" offline.

5.4.2 Verbindungsabbruch während eines Device Logic Downloads

Auf einen Verbindungsabbruch während des Device Logic Downloads reagiert das Gerät mit der Standard Retry-Sequenz (2 Verbindungsaufbauversuche im Abstand von 2min.). Zusätzlich dazu wird, sobald das Gerät den Verbindungsabbruch erkannt hat, der Log-Eintrag "SCRIPT_ERR, SCRIPT DOWNLOAD ERROR" ins Gerätelog eingetragen. Die bestehende Device Logic kann dadurch, dass sie sich im RAM befindet, bis zum nächsten PowerOn weiterhin ausgeführt werden. Nach einem PowerOn ist die Ausführung nicht mehr möglich und es wird der Fehler "SCRIPT_ERR, NO SCRIPT" ins Gerätelog eingetragen. Der Grund dafür ist, dass zu Beginn des Device Logic Downloads der Bereich im Flash-Speicher, in dem sich die Device Logic befindet, gelöscht wird.

5.5 Timeout-Überwachung im Online-Modus

Im Online-Modus sendet der myDatalogC33x standardmäßig in einem Intervall von 15min. und 3sec. (d.h. alle 903sec.) einen Keep Alive Ping an den myDatanet-Server. Dies ermöglicht es dem Server zu erkennen, ob die Verbindung zum myDatalogC33x noch besteht. Um Unterbrechungen der Verbindung zeitnaher erkennen zu können, kann das Standardintervall für den Keep Alive Ping mittels der Funktion "rM2M_ SetTCPKeepAlive()" angepasst werden.

Damit auch der myDatalogC33x eine Unterbrechung der Verbindung zeitnahe erkennen kann, lässt sich am Server der sogenannte "Bidirektionale Alive Ping" aktivieren. Dieser kann global für den kompletten Server, für einen speziellen Kunden oder für eine einzelne Site aktiviert werden (siehe "Benutzerhandbuch für myDatanet-Server " 206.886). Ist der "Bidirektionale Alive Ping" aktiviert, sendet der myDatanet-Server auf jeden Keep Alive Ping des Geräts eine entsprechende Antwort (Keep Alive Response). Ab dem Empfang der ersten Keep Alive Response erwartet der myDatalogC33x eine regelmäßige Keep Alive Response innerhalb von 10sec. als Antwort auf den Keep Alive Ping. Bleibt die Keep Alive Response drei Mal hintereinander aus, wird zunächst versucht, die Kommunikation ohne vollständige Trennung der Verbindung (d.h. nur mittels Neuinitialisierung der Verbindung nur auf TCP-Ebene) wiederherzustellen. Erst wenn dies nicht funktioniert, trennt der myDatalogC33x die Verbindung zum myDatanet-Server vollständig und baut sie unmittelbar wieder erneut auf. Mit dem Empfang der ersten Keep Alive Response wird auch die Erfassung der Round Trip Time [ms] aktiviert. D.h. bei jedem weiteren Keep Alive Ping wird die Zeit gemessen, bis die Keep Alive Response vom Server erhalten wurde. Mittels der Funktion "rM2M_TxItfGetStats()" kann die zuletzt ermittelte "Round Trip Time" vom System gelesen werden.

5.6 Automatische Auswahl des GSM-Netzes

Da der myDatalogC33x mit einem SIM-Chip ausgestattet ist, der eine Mobilfunkverbindung über eine Vielzahl internationaler Serviceprovider gewährleistet (siehe www.microtronics.com/footprint), ist eine Auswahl des GSM-Netzes, in das sich das Gerät einbuchen soll, erforderlich. Diese erfolgt automatisch vom Gerät.

5.7 Ermittlung der GSM/UMTS/LTE-Signalstärke

Die interne Aktualisierungsrate des Messwertes für die GSM/UMTS/LTE-Signalstärke ist abhängig von der Verbindungsart, die mittels der Funktion "rM2M_TxSetMode()" ausgewählt wird:

- Intervall: Aktualisierung jeweils beim Verbindungsaufbau
- Intervall & Wakeup: Aktualisierung alle 30sec.
- Online: Aktualisierung alle 5sec.

Über die Funktion "rM2M_GSMGetRSSI()" kann der zuletzt ermittelte Wert vom System gelesen werden.

5.8 Ermittlung der GSM-Positionsdaten

Alle 24h wird durch die Firmware ein internes Flag gesetzt welches bewirkt, dass beim nächsten Aufruf der Funktion "rM2M_TxStart()" auch eine Ermittlung der GSM-Positionsdaten erfolgt. Durch Setzen des Flags "RM2M_TX_SUPPRESS_ POSUPDATE" beim Aufruf der Funktion kann die Positionsbestimmung jedoch auch unterdrückt werden. Ebenso ist es möglich die Ermittlung der GSM-Positionsdaten durch Setzen des Flags "RM2M_TX_POSUPDATE" beim Aufrufen von "rM2M_TxStart()" gezielt auszulösen. Wurde am myDatalogC33x der Verbindungsmodus "Intervall & Wakeup" aktiviert, das zuvor beschriebene interne Flag von der Firmware gesetzt und der Verbindungsaufbau durch den Empfang einer Wakeup SMS ausgelöst (d.h. über den myDatanet-Server), wird die Ermittlung der GSM-Positionsdaten auf jeden Fall durchgeführt und kann nicht unterdrückt werden. Ist der Verbindungsmodus "online" aktiv, können die GSM-Positionsdaten nicht erzeugt werden.

5.9 Errorhandling

Um zu gewährleisten, dass bei Problemen mit der Device Logic eine Diagnose bzw. Behebung aus der Ferne möglich ist, wurden folgende Übertragungmechanismen in die Firmware integriert. Für den Fall, dass keine Device Logic vorhanden ist, erfolgt die Verbindung zum myDatanet-Server alle 24h . Sollte eine vorhandene Device Logic aufgrund von vom System erkannten Fehlern deaktiviert worden sein, wird dieses Backup Intervall auf 1h gesetzt. In beiden Fällen wird die Verbindugsart „Intervall & Wakeup“ aktiviert, wodurch das Auslösen einer Verbindung über die Oberfläche des myDatanet-Servers möglich ist (siehe "Benutzerhandbuch für myDatanet-Server " 206.886).

5.10 Registrierungsspeicherblöcke

Neben 4 Blöcken mit je 1kB , die im Flash gespeichert sind, kann noch ein weiterer mittels der Funktion "rM2M_RegInit()" optional initialisiert werden, der im RAM abgelegt wird. Dessen Größe kann bei der Initialisierung angegeben werden, ist allerdings auf maximal 1kB begrenzt. Die Registrierungsspeicherblöcke bieten die Möglichkeit gerätespezifische Daten abzulegen und mit dem myDatanet-Server zu synchronisieren. Die Blöcke unterscheiden sich nur hinsichtlich ihrer Zugriffsmöglichkeiten und des

Speicherortes. Daraus ergeben sich vordefinierte Verwendungszwecke, die in der folgenden Tabelle beschrieben werden:

Speicherblock	Zugriff	Speicher	Verwendungszweck
Systemspezifische Daten			
REG_SYS_OTP [1)]	**lesbar:** Device Logic, myDatanet-Server	FLASH	Systeminformationen, die einmalig im Zuge der Produktion geschrieben werden
REG_SYS_FLASH [1)]	**lesbar:** Device Logic, myDatanet-Server	FLASH	Systeminformationen, die im Betrieb veränderbar sein müssen
Applikationsspezifische Daten			
REG_APP_OTP	**lesbar:** Device Logic, myDatanet-Server **schreibbar:** Device Logic	FLASH	applikationsspezifische Informationen, die einmalig im Zuge der Produktion geschrieben werden (Empfehlung, ein mehrfaches Schreiben wird von der Firmware nicht verhindert)
REG_APP_FLASH	**lesbar:** Device Logic, myDatanet-Server **schreibbar:** Device Logic	FLASH	applikationsspezifische Informationen, die im Betrieb veränderbar sein müssen
Applikationsspezifische, flüchtige Daten			
REG_APP_STATE [2)]	**lesbar:** Device Logic, myDatanet-Server **schreibbar:** Device Logic	RAM	applikationsspezifische Informationen, die im Betrieb veränderbar sein müssen und kein nicht flüchtiges Speichern im Flash erfordern (z.B. aktueller Gerätestatus).

1) Das Schreiben von Daten in diese beiden Speicherblöcke ist dem Hersteller vorbehalten.

2) Dieser Speicherblock ist nur verfügbar, wenn er mittels der Funktion "rM2M_RegInit()" initialisiert wurde.

> ***Hinweis:*** *Es ist auch möglich, die Speicherblöcke im Zuge des Produktionsprozesses über die lokalen Schnittstellen (USB und beide UART) zu beschreiben. Um Informationen darüber zu erhalten, ist jedoch eine Vereinbarung mit dem Hersteller (siehe "Kontaktinformationen" auf Seite 253) erforderlich.*

Für den Zugriff auf die Registrierungsspeicherblöcke stehen die Funktionen "rM2M_RegGetString()", "rM2M_RegGetValue()", "rM2M_RegSetString()", "rM2M_RegSetValue()", "rM2M_RegDelValue()" und "rM2M_RegDelKey()" zur Verfügung.

5.10.1 REG_APP_OTP

Durch Speichern des "Product Identity Profile" (PIP) in diesem Registrierungsspeicherblock können am myDatanet-Server die im Folgenden beschriebenen Funktionen ausgelöst werden. Das PIP besteht aus folgenden Feldern:

pipCustomer
Name des Kunde, dem die Messstelle zugeordnet werden soll [2-50 Zeichen].

pipCtx
Name der Messstelle, die angelegt/verwendet werden soll [2-50 Zeichen].

pipAppId
ID der IoT Applikation auf Basis der die Messstelle erstellt werden soll [max. 50 Zeichen].

pipAppVer (optional)
Momentan im Gerät installierte Version der Device Logic (z.B. 7) [Integer].

pipCtxAutocreate (optional)
gibt an, ob die Messstelle (falls sie noch nicht existiert) angelegt werden soll ("0" oder "1" sind als String zu speichern)

- *"0": anlegen einer neuen Messstelle ist nicht zulässig*
- *"1": neue Messstelle darf angelegt werden (default)*

Empfängt der myDatanet-Server einen PIP, wird zwischen zwei grundsätzlichen Szenarien unterschieden:

- Es existiert noch keine Messstelle mit dem im Feld "pipCtx" angegebenen Namen:

  Nur wenn sowohl der Kunde als auch die Applikations-Vorlage am myDatanet-Server gefunden wurden und "pipCtxAutocreate=1" bzw. das Feld nicht vorhanden ist, wird die Messstelle neu angelegt.

- Eine Messstelle mit dem im Feld "pipCtx" angegebenen Namen wurde am Server gefunden:

  In diesem Fall haben die Felder "pipCtxAutocreate" und "pipCustomer" keine Relevanz. Stimmen die im Feld "pipAppId" angegebene Applikations-ID und die der gefundenen Messstelle überein, wird das Gerät der Messstelle, selbst wenn es sich in einem anderen Kunden befindet, zugewiesen. Dazu wird das Gerät in den entsprechenden Kunden verschoben. Ist der Messstelle bereits ein Gerät zugewiesen, wird die Zuweisung des bestehenden Geräts aufgehoben. Das bestehende Gerät wird in den Pool des Kunden verschoben.

5.11 File Transfer

Es ist möglich bis zu 60 Dateien für den File Transfer zu registrieren (siehe "FT_Register()"). Dabei muss der Funktion "FT_Register()" eine Callback Funktion (siehe "Callback Funktionen" auf Seite 180) übergeben werden, die beim Empfang eines File Transfer Kommandos aufgerufen werden soll. Die Callback Funktion muss in der Lage sein, alle File Transfer Kommandos (siehe "File Transfer Kommandos" im Kapitel "Konstanten" auf Seite 180) zu behandeln. Im Zuge der Registrierung müssen auch noch mittels der Funktion "FT_SetProps()" die Dateieigenschaften gesetzt werden. Soll eine Datei nicht mehr für den File Transfer verfügbar sein, kann sie durch die Funktion "FT_Unregister()" aus der Registrierung entfernt werden.

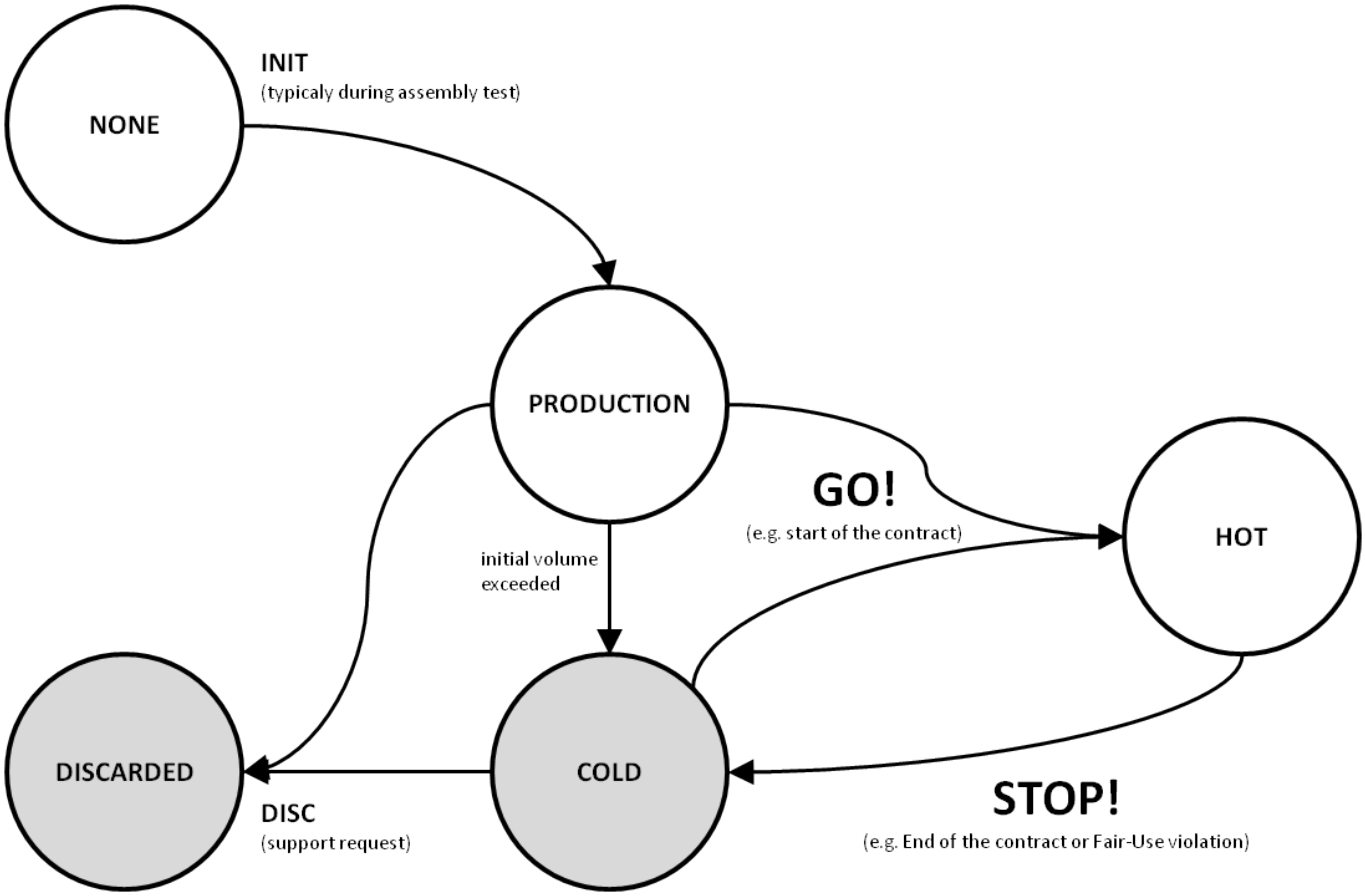
Beim Empfang eines File Transfer Kommandos wird eine Sitzung gestartet, welche nach 15sec. automatisch beendet wird, wenn die Device Logic das Kommando nicht korrekt behandelt. Um Konflikte zu vermeiden, kann immer nur eine Sitzung nach der anderen aktiv sein.

5.12 Bedeutung des SIM-Status

Das Gerät erhält vom myDatanet-Server Informationen über die zulässige Verwendung des SIM-Chips. Für diesen SIM-Status sind folgende Zustände definiert:

SIM-Status	Übertragung per Device Logic	Erklärung
RM2M_SIM_STATE_NONE	ja	Initialzustand
RM2M_SIM_STATE_PRODUCTION	ja	Neu produziertes Gerät liegt auf Lager
RM2M_SIM_STATE_HOT	ja	Gültiger Vertrag
RM2M_SIM_STATE_COLD	nein	Vertragsende oder Fair-Use Verletzung
RM2M_SIM_STATE_DISCARDED	nein	Gerät wurde außer Dienst gestellt

Wie in der vorangehenden Tabelle ersichtlich ist das Auslösen der Verbindung per Device Logic für die Zustände "RM2M_SIM_STATE_COLD" und "RM2M_SIM_STATE_DISCARDED" nicht möglich. Die Funktionen "rM2M_TxStart()" und "rM2M_TxSetMode()" liefern in diesem Fall den Fehlercode "ERROR_SIM_STATE". Um ein Gerät das sich im Zustand "RM2M_SIM_STATE_COLD" befindet wieder in den Zustand "RM2M_SIM_STATE_HOT" zu versetzen, setzen Sie sich mit dem Hersteller (siehe "Kontaktinformationen" auf Seite 253) in Verbindung. Der SIM-Status kann jeder Zeit mittels der Funktion "rM2M_GSMGetInfo()" ausgelesen werden. Bei jeder Änderung des SIM-Status erfolgt zudem ein Eintrag (z.B. SIM_STATE, HOT) ins Gerätelog.



Statediagramm der SIM-Zustände

Kapitel 6 Lagerung, Lieferung und Transport

6.1 Eingangskontrolle

Kontrollieren Sie den Lieferumfang sofort nach Eingang auf Vollständigkeit und augenscheinliche Unversehrtheit. Melden Sie eventuell festgestellte Transportschäden unverzüglich an den anliefernden Frachtführer. Senden Sie ebenfalls unverzüglich eine schriftliche Meldung an Microtronics Engineering GmbH. Unvollständigkeiten der Lieferung müssen innerhalb von 2 Wochen schriftlich an Ihre zuständige Vertretung oder direkt an die Firmenzentrale des Herstellers (siehe "Kontaktinformationen" auf Seite 253) gerichtet werden.

Hinweis: *Später eingehende Reklamationen werden nicht anerkannt!*

6.2 Lieferumfang

Hinweis: *Eine für den Betrieb zwingend erforderliche Antenne (siehe "Antennen" auf Seite 241) ist nicht im Standardlieferumfang enthalten und muss gesondert geordert werden.*

Zum Standardlieferumfang des myDatalogC33x WIFI/2G/3G/4G World (301331) gehören:

- rapidM2M M23x WIFI/2G/3G/4G World [1)]

[1)] Das rapidM2M Modul (rapidM2M M23x WIFI/2G/3G/4G World) wurde bereits während der Produktion in den myDatalogC33x WIFI/2G/3G/4G World eingesetzt.

Weiteres Zubehör wie Montagesets, Antennen, Netzteil usw. je nach Bestellung. Diese bitte anhand des Lieferscheins prüfen.

6.3 Lagerung

Folgende Lagerbedingungen sind unbedingt einzuhalten:

myDatalogC33x	Lagertemperatur	-40...+85°C
	Feuchte	15...90%rH

Die Messtechnik ist vor korrosiven und organischen Lösungsmitteldämpfen, radioaktiver Strahlung sowie starker elektromagnetischer Strahlung geschützt aufzubewahren.

6.4 Transport

Schützen Sie den myDatalogC33x vor starken Stößen, Schlägen, Erschütterungen oder Vibrationen. Der Transport muss in der Originalverpackung erfolgen.

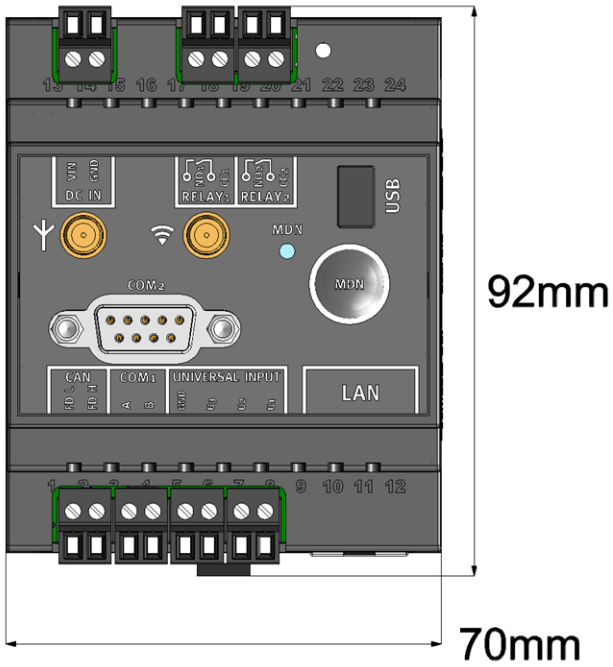
6.5 Rücksendung

Jeder Rücksendung muss ein vollständig ausgefülltes Retourenformular, welches im Servicebereich des myDatanet-Servers erhältlich ist, beigelegt werden. Die unbedingt erforderliche "RMA Nr" erhalten Sie vom Support & Service-Center (siehe "Kontaktinformationen" auf Seite 253). Die Rücksendung des myDatalogC33x muss in der Originalverpackung frachtfrei zu Microtronics Engineering GmbH (siehe "Kontaktinformationen" auf Seite 253) erfolgen. Nicht ausreichend frei gemachte Sendungen werden nicht angenommen!

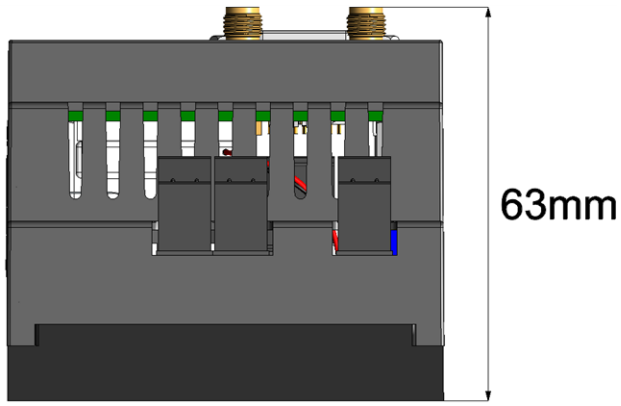
Kapitel 7 Installation

Wichtiger Hinweis: *Um Schäden am Gerät zu vermeiden, dürfen die in diesem Abschnitt der Anleitung beschriebenen Arbeiten nur von qualifiziertem Personal ausgeführt werden.*

7.1 Abmessungen



Abmessungen: Breite und Höhe



Abmessungen: Tiefe

7.2 Montage des myDatalogC33x

Wichtiger Hinweis:

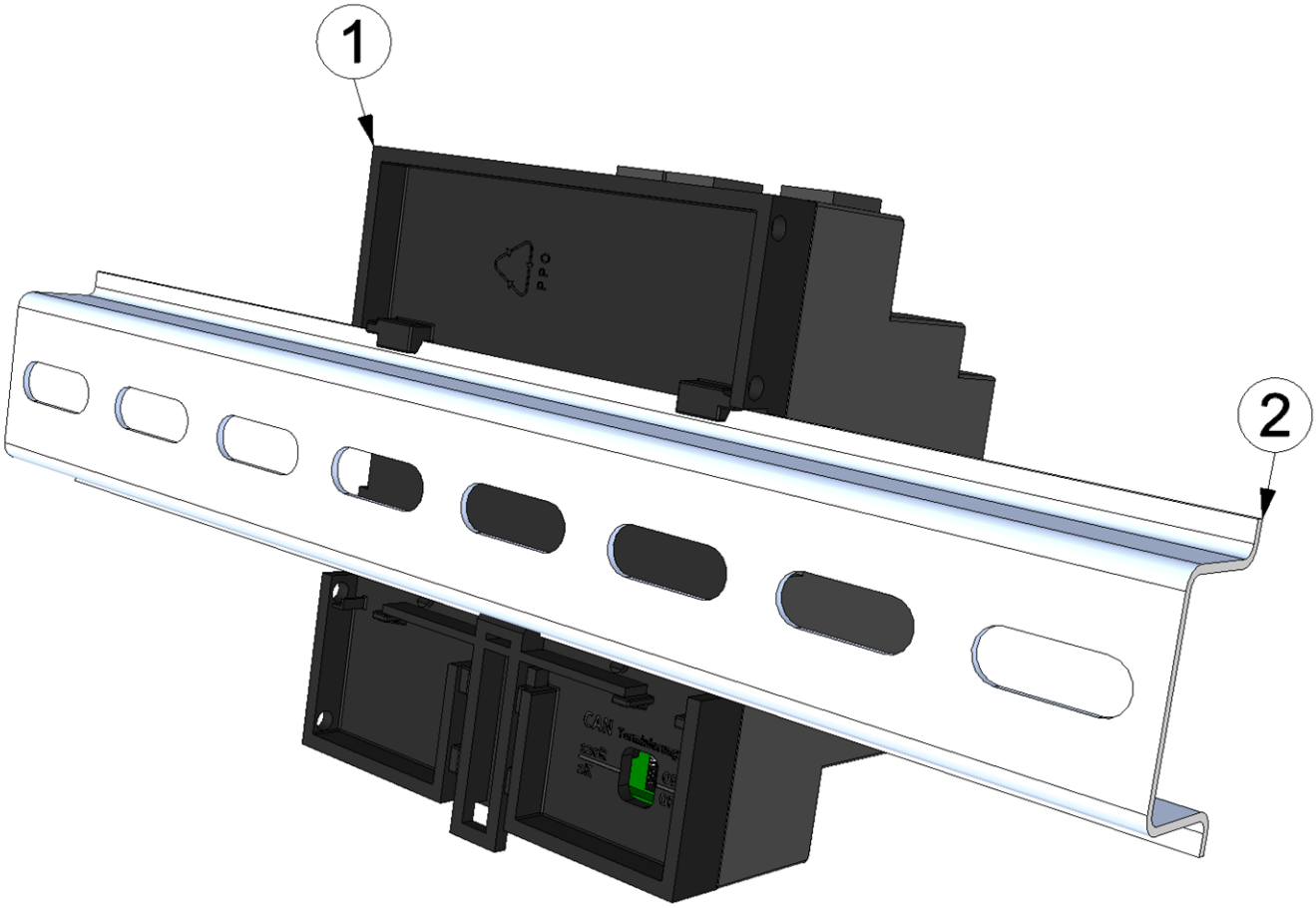
- *Achten Sie auf eine sachgemäße Montage!*
- *Befolgen Sie bestehende gesetzliche bzw. betriebliche Richtlinien!*
- *Unsachgemäße Handhabung kann zu Verletzungen und/oder Beschädigungen an den Instrumenten führen!*
- *Der myDatalogC33x ist nicht für den Einsatz in geschlossenen Kanälen zugelassen.*

Der Platz für die Montage muss nach bestimmten Kriterien ausgewählt werden. Vermeiden Sie unbedingt die folgenden Gegebenheiten:

- direkte Sonneneinstrahlung
- direkte Witterungseinflüsse (Regen, Schnee, ...)
- Gegenstände, die starke Hitze ausstrahlen (maximale Umgebungstemperatur: -20...+60°C)
- Objekte mit starkem elektromagnetischem Feld (Frequenzumrichter o.ä.)
- korrodierende Chemikalien oder Gase
- mechanische Stöße
- direkte Installation an Geh- oder Fahrwegen
- Vibrationen
- radioaktive Strahlung

Hinweis: *Unter und über dem Gerät sollten Sie ca. 2-5cm Abstand für die Kabelanschlüsse vorsehen. Die Antennenanschlüsse befinden sich auf der Vorderseite des Geräts. Der benötigte Platz richtet sich nach den verwendeten Antennen. Weitere Informationen zu den Abmessungen für die Montage entnehmen Sie dem jeweiligen Unterkapitel.*

7.2.1 Hutschienenmontage



Hutschienenmontage

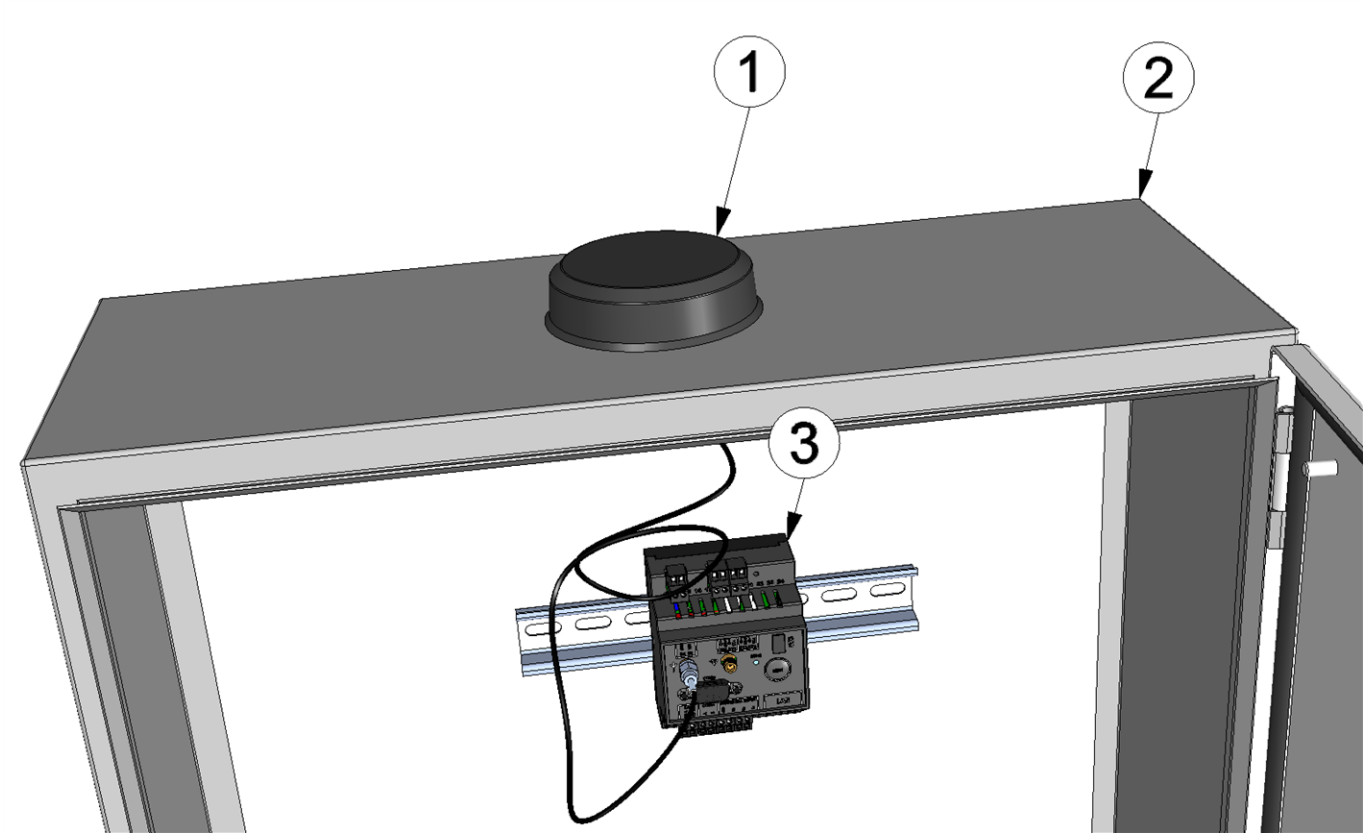
1 myDatalogC33x	**2** Hutschiene

1. Setzen Sie das myDatalogC33x auf der Oberkante der Hutschiene auf. Durch eine leichte Drehung um die Horizontalachse rastet das myDatalogC33x auf der Hutschiene ein (siehe Abbildung "Hutschienenmontage" auf Seite 44).

7.2.2 Montage in einem Schaltschrank

Die Stabantenne Multiband 2G/3G SMA-M (301075) eignet sich nicht für die Montage innerhalb eines Schaltschranks, da das Mobilfunksignal durch das Metall des Schranks abgeschirmt wird. Der Hersteller empfiehlt in diesem Fall die Verwendung der als Zubehör erhältlichen Flachantenne Disc SMA-M 2,5m (206.816) bzw. der Flachantenne Disc US SMA-M 2,5m (206.818).

> ***Wichtiger Hinweis:*** *Die Montage der Flachantenne Disc SMA-M 2,5m bzw. der Flachantenne Disc US SMA-M 2,5m ist nur bis zu einer Wandstärke von 3mm möglich.*



Schaltschrank mit montierter Flachantenne Disc SMA-M 2,5m (206.816)

1 Flachantenne Disc SMA-M 2,5m (206.816)	**3** myDatalogC33x
2 Schaltschrank	

7.3 Sicherheitshinweise zur Verkabelung

Wichtiger Hinweis: *Um Schäden zu vermeiden, stellen Sie stets die Spannungsversorgung am Gerät ab, wenn elektrische Anschlüsse durchgeführt werden.*

Wenn Anschlüsse an den myDatalogC33x gelegt werden, müssen die folgenden Warnungen und Hinweise ebenso beachtet werden, wie Warnungen und Hinweise, die in den einzelnen Kapiteln zum Einbau zu finden sind. Weitere Sicherheitsinformationen finden Sie unter "Sicherheitshinweise" auf Seite 17.

7.3.1 Hinweise zur Vermeidung elektrostatischer Entladungen (ESD)

Wichtiger Hinweis: *Um Gefahren und ESD-Risiken zu minimieren, sollten Wartungsprozeduren, für die keine Stromversorgung des Geräts erforderlich ist, nur nach Trennung vom Stromnetz ausgeführt werden.*

Die empfindlichen elektronischen Komponenten im Geräteinneren können durch statische Elektrizität beschädigt werden, was zur Beeinträchtigung der Geräteleistung bis hin zum Ausfall des Geräts führen kann. Der Hersteller empfiehlt die folgenden Schritte zur Vermeidung von Beschädigungen des Geräts durch elektrostatische Entladungen:

- Leiten Sie eventuell auf Ihrem Körper vorhandene statische Elektrizität ab, bevor Sie elektronische Komponenten des Geräts (wie z.B. Leiterplatten und die Komponenten darauf) berühren. Hierzu können Sie eine geerdete metallische Oberfläche berühren, wie etwa den Gehäuserahmen eines Geräts oder ein Metallrohr.
- Vermeiden Sie unnötige Bewegungen, um den Aufbau statischer Ladungen zu vermindern.
- Transportieren Sie statisch-empfindliche Komponenten in antistatischen Behältnissen oder Verpackungen.
- Tragen Sie ein Antistatikarmband, das über ein Kabel geerdet ist, um Ihren Körper zu entladen und von statischer Elektrizität freizuhalten.
- Fassen Sie Komponenten, die gegen Aufladungen empfindlich sind, nur in einem Antistatik-Arbeitsbereich an. Verwenden Sie, falls möglich, antistatische Fußbodenbeläge und Arbeitsunterlagen.

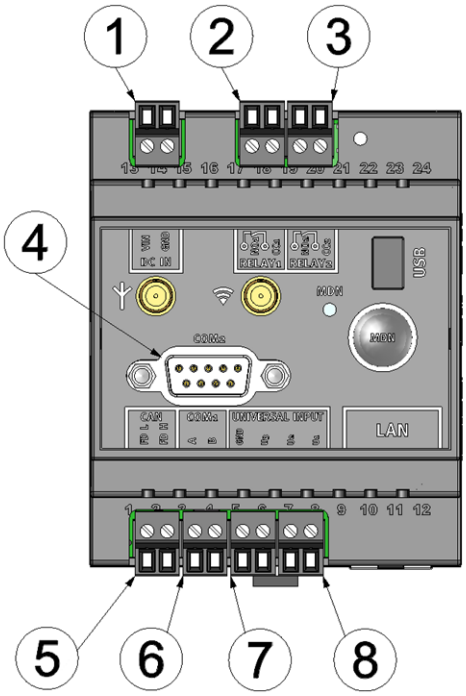
7.4 Elektrische Installation

Wichtiger Hinweis: *Um Schäden am Gerät zu vermeiden, darf nur qualifiziertes Personal die in diesem Kapitel der Bedienungsanleitung beschriebene Installation durchführen.*

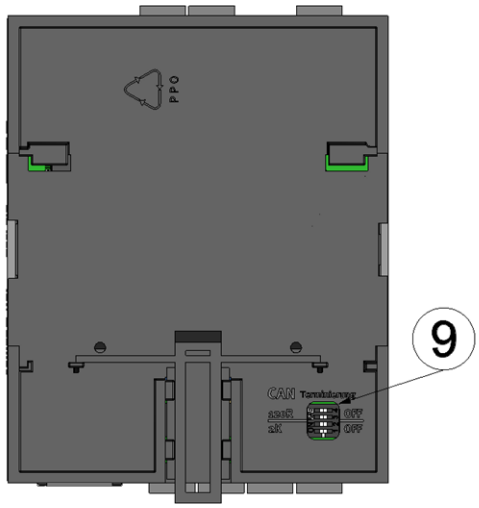
7.4.1 Anschluss der Sensoren, der Aktoren und der Versorgung

Wichtiger Hinweis:

- *Alle Verkabelungsarbeiten sollten im stromlosen Zustand erfolgen!*
- *Achten Sie auf eine sachgemäße Montage!*
- *Befolgen Sie bestehende gesetzliche bzw. betriebliche Richtlinien!*
- *Unsachgemäße Handhabung kann zu Verletzungen und/oder Beschädigungen an den Instrumenten führen!*
- *Verlegen Sie alle Daten- und Stromkabel so, dass sie keine Stolpergefahr darstellen und die Kabel keine scharfen Krümmungen aufweisen.*



Anschluss der Sensoren, der Aktoren und der Versorgung (Vorderseite)



Anschluss der Sensoren, der Aktoren und der Versorgung (Rückseite)

1 Versorgung (V IN, GND)	**6** Com1 (RS485)
2 Relais 1	**7** Masse und Universaleingang 1
3 Relais 2	**8** Universaleingang 2-3
4 Com2 (RS232)	**9** Dip-Switch zum Ein-/Ausschalten der Abschlusswiderstände für die CAN-Schnittstelle
5 CAN / CAN FD	

DC IN

VIN	Versorgungsspannung: 9...32VDC (+/-10%), max. 9W
GND	Masse

RELAY1[1)]

NO1	Arbeitskontakt des potentialfreien Schaltkontakts 1 (Normally Open)
CC1	Wurzel des potentialfreien Schaltkontakts 1

[1)] Parallelschaltung eines PhotoMOS-Relais (hohe Schaltfrequenzen) und eines mechanischen Relais (hohe Schaltströme)

RELAY2[1)]

NO2	Arbeitskontakt des potentialfreien Schaltkontakts 2 (Normally Open)
CC2	Wurzel des potentialfreien Schaltkontakts 2

[1)] Parallelschaltung eines PhotoMOS-Relais (hohe Schaltfrequenzen) und eines mechanischen Relais (hohe Schaltströme)

COM2

1	NC
2	RXD Leitung der RS232-Schnittstelle
3	TXD Leitung der RS232-Schnittstelle
4	NC
5	Masse
6	NC
7	RTS Leitung der RS232-Schnittstelle
8	CTS Leitung der RS232-Schnittstelle
9	5V [1], max. 750mA

[1] Versorgungsspannung (reserviert für Erweiterungen)

CAN

FD L	CAN Low Leitung der CAN-Schnittstelle
FD H	CAN High Leitung der CAN-Schnittstelle

COM1

A	RS485 A
B	RS485 B

UNIVERSAL INPUT

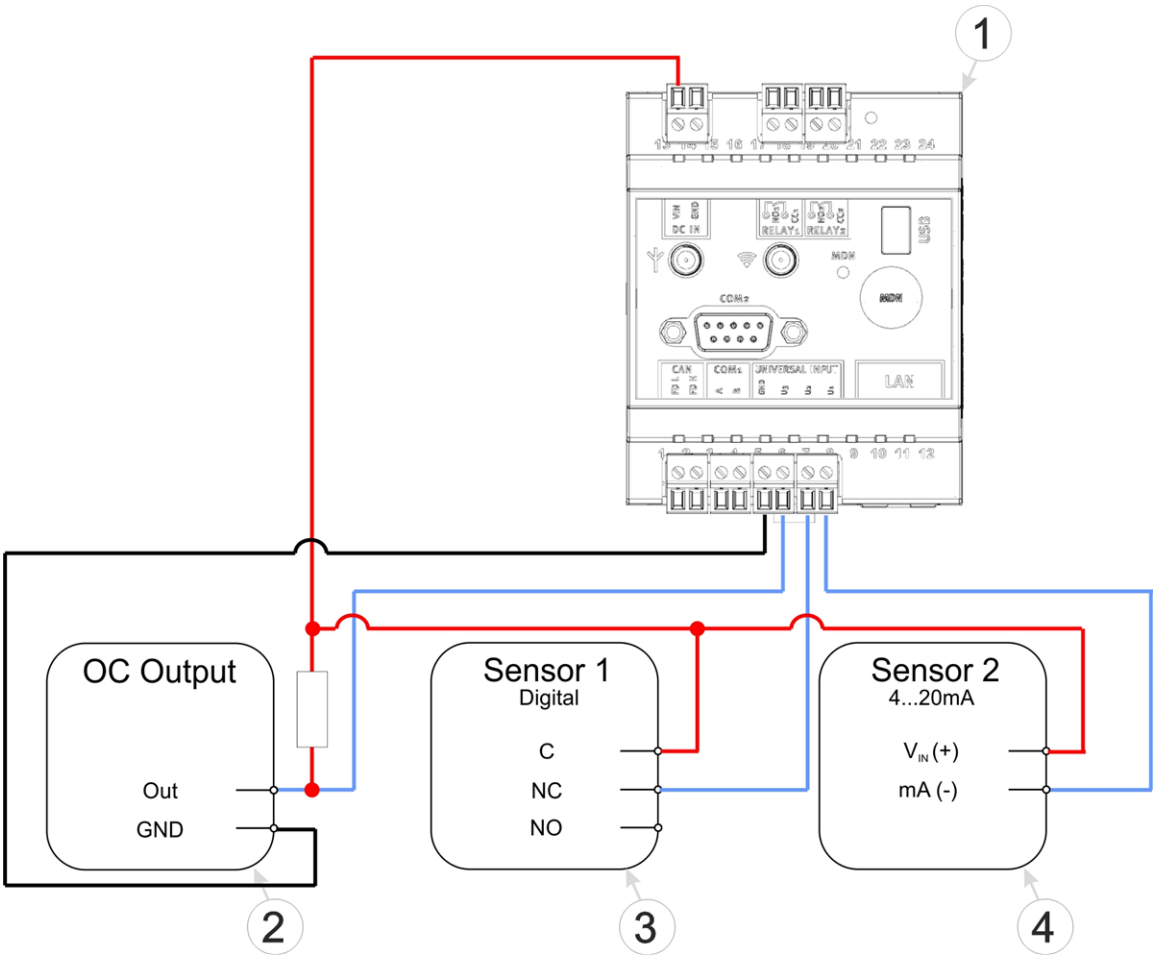
GND	Masse
UI3	Universaleingang 3
UI2	Universaleingang 2
UI1	Universaleingang 1

1. Verbinden Sie Ihre Sensoren und Aktoren mit den Eingängen und Ausgängen. Achten Sie dabei auf Stromlosigkeit! Die Kabel zur Versorgung des myDatalogC33x sollten im stromlosen Zustand mit den Versorgungsklemmen verbunden werden.
2. Schließen Sie die Antenne an (siehe "Anschluss der GSM-Antenne" auf Seite 50).
3. Schalten Sie die 9...32VDC Versorgungsspannung des myDatalogC33x ein.

Der folgende Schritt ist nicht zwingend erforderlich.

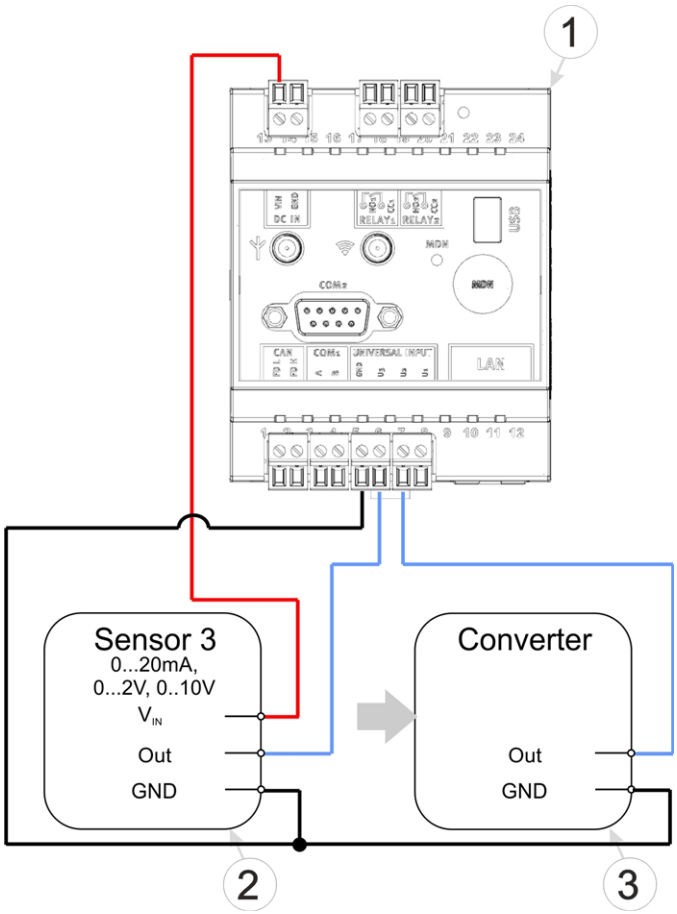
4. Überprüfen Sie, ob die Verbindung zum myDatanet korrekt funktioniert hat (siehe "Kommunikation mit dem Gerät testen" auf Seite 70).

7.4.1.1 Anschlussbeispiele



Anschlussbeispiele (OC Output, Digital, 4…20mA)

1 myDatalogC33x	**4** potentialfreier Relaiskontakt
2 Sensor mit Open Collector Ausgang	**4** 2-Leiter mA-Sensor



Anschlussbeispiele (0…20mA, 0…2V, 0…10V, Converter)

1 myDatalogC33x	**3** Signalwandler, Trennwandler
2 3-Leiter mA-Sensor oder 3-Leiter U-Sensor	

7.4.2 Anschluss der GSM-Antenne

> ***Wichtiger Hinweis:*** *Um eine korrekte Funktion zu gewährleisten, benutzen Sie nur Antennen, die vom Hersteller geliefert werden.*

Die Standardantenne wird direkt am Antennenstecker (siehe "Übersicht" auf Seite 20) des myDatalogC33x angebracht. Im Falle einer niedrigen Funksignalstärke können Sie die Kuppelantenne Multiband SMA-M 3m (301212) verwenden.

Wenn die Entfernung zwischen der Lage der Antenne und dem myDatalogC33x zu groß ist, können Sie eine 2,5m Antennenverlängerung SMA-M/SMA-F 2,5m (206.807) verwenden.

1. Stellen Sie sicher, dass das myDatalogC33x spannungslos ist.
2. Verbinden Sie, falls Sie eine Antennenverlängerung benötigen, diese zuerst mit der Antenne.
3. Verbinden Sie die Antennenverlängerung bzw. die Antenne direkt mit dem Antennenanschluss des myDatalogC33x (siehe "Übersicht" auf Seite 20).

   > ***Wichtiger Hinweis:*** *Vermeiden Sie zu starke Krafteinwirkung beim Festziehen der Antenne. Benutzen Sie kein Werkzeug zum Festziehen der Antennen bzw. der Antennenverlängerung, sondern ziehen Sie die Antenne mit der Hand fest.*

4. Stellen Sie die Spannungsversorgung des myDatalogC33x wieder her.

Der folgende Schritt ist nicht zwingend erforderlich.

5. Überprüfen Sie, ob die Verbindung zum myDatanet korrekt funktioniert hat (siehe "Kommunikation mit dem Gerät testen" auf Seite 70).

7.4.3 Anschluss der Erweiterungsmodule

Wichtiger Hinweis:

- *Alle Verkabelungsarbeiten sollten im stromlosen Zustand erfolgen!*
- *Achten Sie auf eine sachgemäße Montage!*
- *Befolgen Sie bestehende gesetzliche bzw. betriebliche Richtlinien!*
- *Unsachgemäße Handhabung kann zu Verletzungen und/oder Beschädigungen an den Instrumenten führen!*
- *Verlegen Sie alle Daten- und Stromkabel so, dass sie keine Stolpergefahr darstellen und die Kabel keine scharfen Krümmungen aufweisen.*

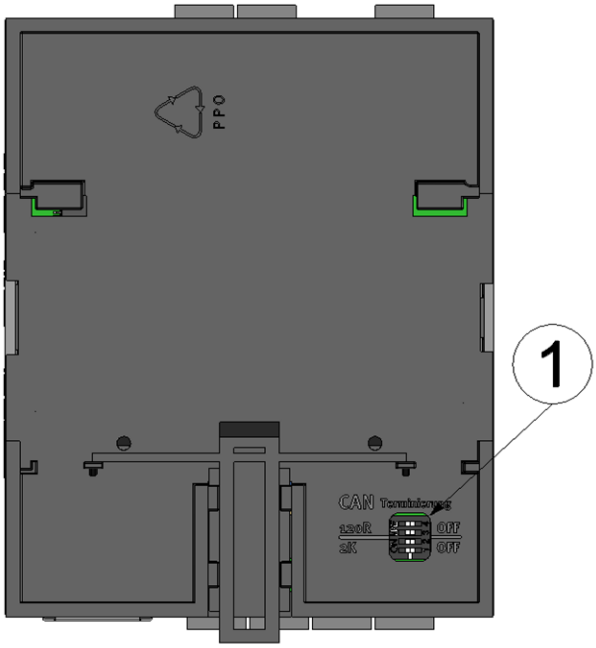
Eine Auflistung kompatibler Erweiterungsmodule finden Sie im Kapitel "Erweiterungsmodule" auf Seite 242. Es ist möglich, bis zu 10 Erweiterungsmodule mittels CAN-Bus mit dem myDatalogC33x zu verbinden. Die Gesamtlänge des CAN-Bus darf dabei 20m nicht überschreiten. Wird die CAN-Schnittstelle des myDatalogC33x für die Anbindung von Erweiterungsmodulen verwendet, steht sie nicht mehr für Mess- und Steueraufgaben zur Verfügung (d.h. für den direkten Anschluss von Sensoren und Aktoren). In diesem Anwendungsfall sollen keine anderen Busteilnehmer, außer den Erweiterungsmodulen und dem myDatalogC33x , mit dem CAN-Bus verbunden werden (d.h. es wird ein eigener abgekapselter CAN-Bus für die Anbindung von Erweiterungsmodulen benötigt).

7.4.3.1 CAN-Bus ohne Stichleitungen

Wichtiger Hinweis: *Alle Verkabelungsarbeiten sollten im stromlosen Zustand erfolgen!*

1. Setzen Sie Abschlusswiderstände für die CAN-Schnittstelle bei allen Busteilnehmern (Erweiterungsmodulen und myDatalogC33x) entsprechend der Position, die sie am Bus einnehmen werden. Beim ersten und beim letzen Busteilnehmer müssen die 120Ω Abschlusswiderstände (S3 und S4 des Dip-Switch) eingeschaltet werden.

   Hinweis: *Da sich die Dip-Switch zum Ein-/Ausschalten der Abschlusswiderstände für die CAN-Schnittstelle auf der Rückseite der Geräte befinden, empfiehlt es sich, die Einstellung der Abschlusswiderstände vor der Montage der Geräte vorzunehmen.*

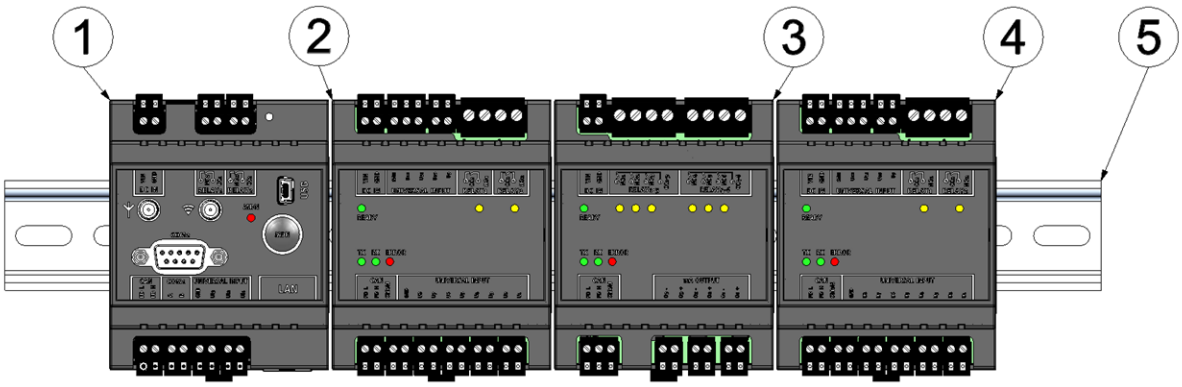


Rückseite des myDatalogC33x oder eines Erweiterungsmoduls

1	Dip-Switch zum Ein-/Ausschalten der Abschlusswiderstände für die CAN-Schnittstelle

2. Bringen Sie das myDatalogC33x und die Erweiterungsmodule in die endgültige Montageposition (z.B. nebeneinander auf einer Hutschiene). Informationen zur sachgemäßen Montage des myDatalogC33x finden Sie im Kapitel "Montage des myDatalogC33x " auf Seite 43. Erläuterungen zur Montage der Erweiterungsmodule finden Sie im Benutzerhandbuch des jeweiligen Erweiterungsmoduls.

   Hinweis: *Beachten Sie bei der Auswahl der Montagepositon der einzelnen Geräte, dass die Gesamtlänge des CAN-Bus 20m nicht überschreiten darf*



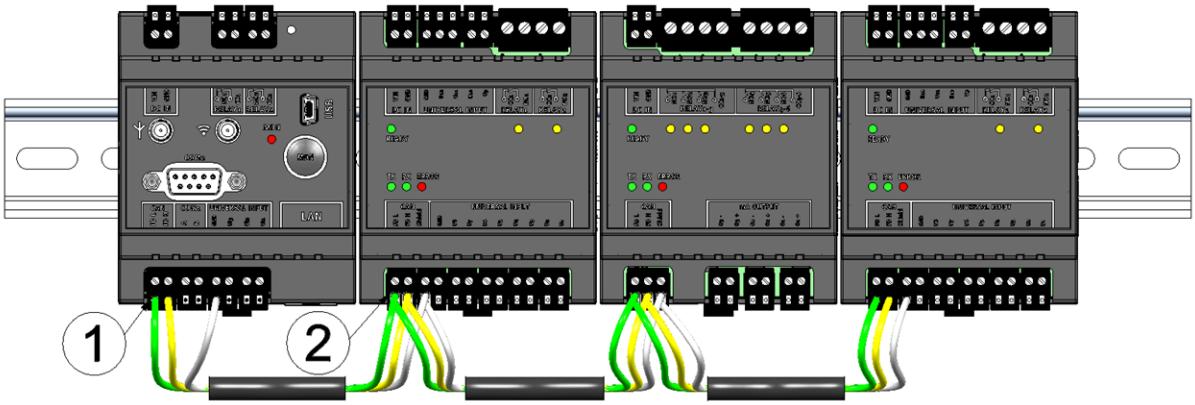
Hutschiene mit montiertem myDatalogC33x und Erweiterungsmodulen

1	myDatalogC33x (120Ω Abschlusswiderstand eingeschaltet)	**4**	myDatalogC3e 12UI/2Rel (120Ω Abschlusswiderstand eingeschaltet)
2	myDatalogC3e 12UI/2Rel	**5**	Hutschiene
3	myDatalogC3e 3mA/6Rel		

3. Verbinden Sie die CAN-Schnittstelle des myDatalogC33x mit jenen der Erweiterungsmodule. Dabei sind jeweils alle "FD L" Klemmen, sowie alle "FD H" Klemmen miteinander zu verbinden. Achten Sie dabei auf Stromlosigkeit!

   Hinweis: *Verwenden Sie für die Verbindung der CAN-Schittstellen ein geschirmtes Kabel. An den Erweiterungsmodulen steht für den Anschluss des Kabelschirms die Klemme "Shield" zur Verfügung. Am myDatalogC33x muss der Kabelschirm mit der GND Klemme verbunden werden. Verwenden Sie eine Twin-Aderendhülsen bzw. Verteilerklemmen, falls Sie mehrere Kabel mit der GND Klemme verbinden wollen.*

   Hinweis: *Da sowohl am myDatalogC33x als auch an den Erweiterungsmodulen nur eine Klemme für jedes der Signale bereitsteht, sollten Sie, falls Sie mehr als ein Erweiterungsmodul verwenden wollen, Twin-Aderendhülsen einsetzen.*

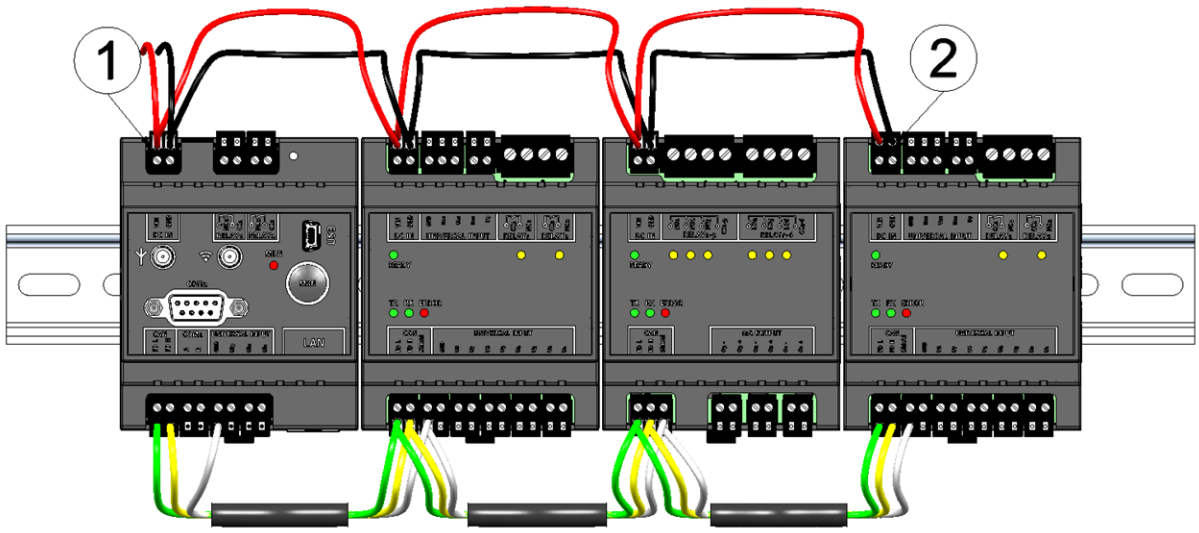


Verkabeln der CAN-Schnittstelle

1 Aderendhülse	**2** Twin-Aderendhülsen (Aderendhülse für 2 Kabel)

4. Verbinden Sie Ihre Sensoren und Aktoren mit den Eingängen und Ausgängen des myDatalogC33x und der Erweiterungsmodule. Achten Sie dabei auf Stromlosigkeit!

5. Verbinden Sie die Kabel zur Versorgung des myDatalogC33x und der Erweiterungsmodule mit den VIN und GND Klemmen. Achten Sie beim Verbinden auf Stromlosigkeit! Verwenden Sie auch hier Twin-Aderendhülsen falls mehr als ein Kabel pro Klemme verbunden werden soll.



Anschließen der Versorgung

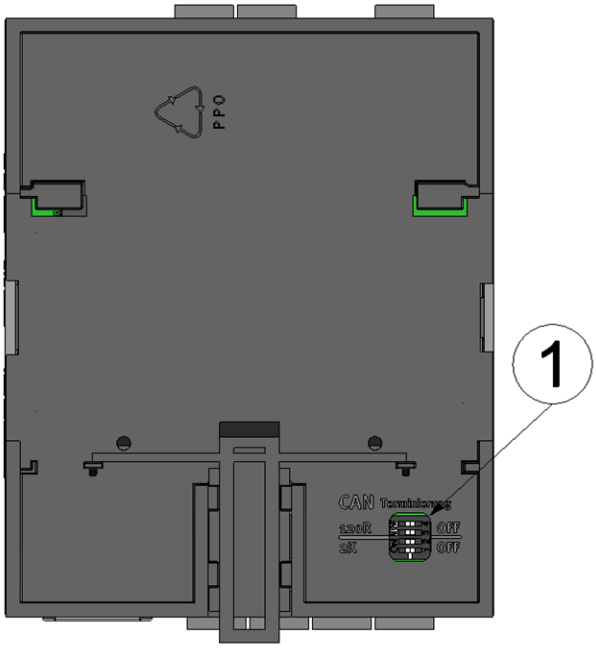
1 Twin-Aderendhülse	**2** Aderendhülse

7.4.3.2 CAN-Bus mit Stichleitung

Wichtiger Hinweis: *Alle Verkabelungsarbeiten sollten im stromlosen Zustand erfolgen!*

1. Setzen Sie Abschlusswiderstände für die CAN-Schnittstelle bei allen Busteilnehmern (Erweiterungsmodulen und myDatalogC33x) entsprechend der Position, die sie am Bus einnehmen werden. Beim ersten und beim letzen Busteilnehmer müssen die 120Ω Abschlusswiderstände (S3 und S4 des Dip-Switch) eingeschaltet werden. Bei Busteilnehmern, die per Stichleitung mit dem Bus verbunden werden, muss der 2k Abschlusswiderstand (S1 und S1 des Dip-Switch) eingeschaltet werden, falls die Stichleitung länger als 0,3m ist.

   Hinweis: *Da sich die Dip-Switch zum Ein-/Ausschalten der Abschlusswiderstände für die CAN-Schnittstelle auf der Rückseite der Geräte befinden, empfiehlt es sich, die Einstellung der Abschlusswiderstände vor der Montage der Geräte vorzunehmen.*

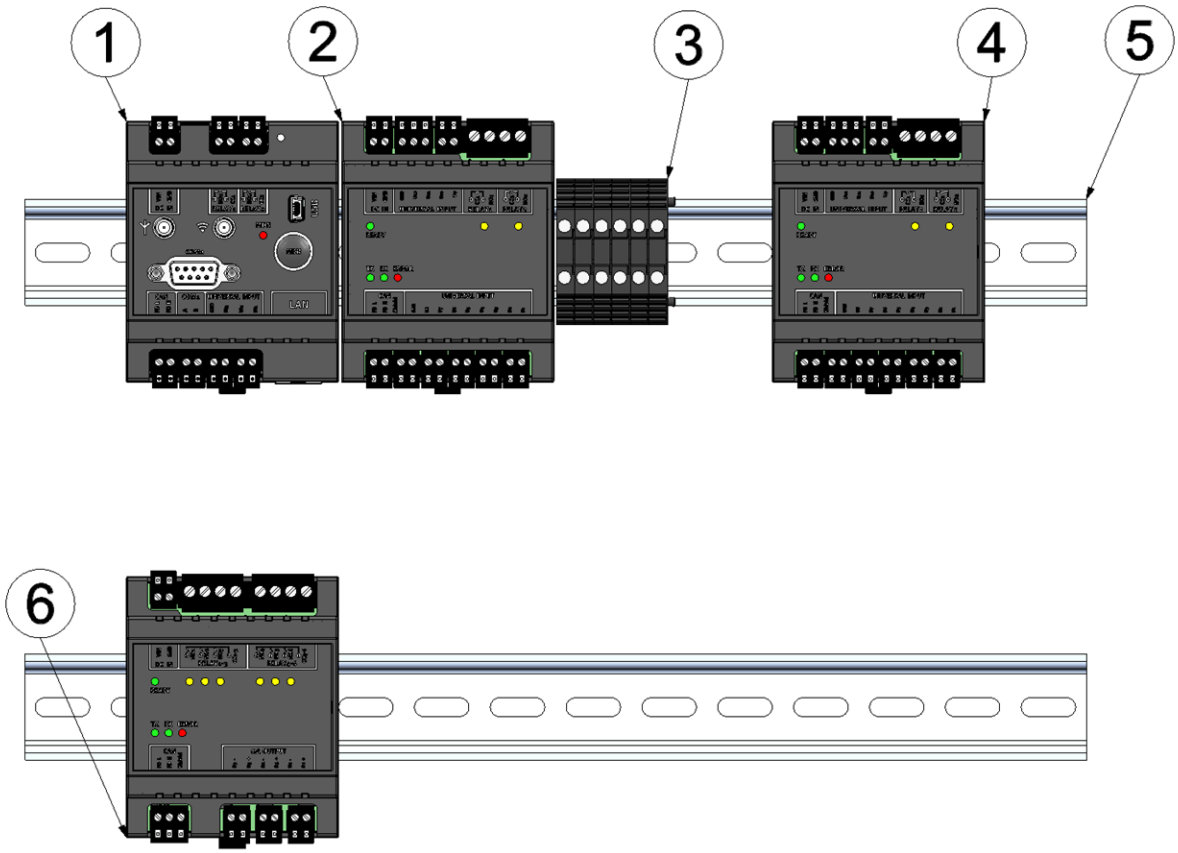


Rückseite des myDatalogC33x oder eines Erweiterungsmoduls

1	Dip-Switch zum Ein-/Ausschalten der Abschlusswiderstände für die CAN-Schnittstelle

2. Bringen Sie den myDatalogC33x und die Erweiterungsmodule in die endgültige Montageposition (z.B. montiert auf mehreren Hutschienen). Informationen zur sachgemäßen Montage des myDatalogC33x finden Sie im Kapitel "Montage des myDatalogC33x " auf Seite 43. Erläuterungen zur Montage der Erweiterungsmodule finden Sie im Benutzerhandbuch des jeweiligen Erweiterungsmoduls.

   Hinweis: *Beachten Sie bei der Auswahl der Montagepositon der einzelnen Geräte, dass die Gesamtlänge des CAN-Bus 20m nicht überschreiten darf*



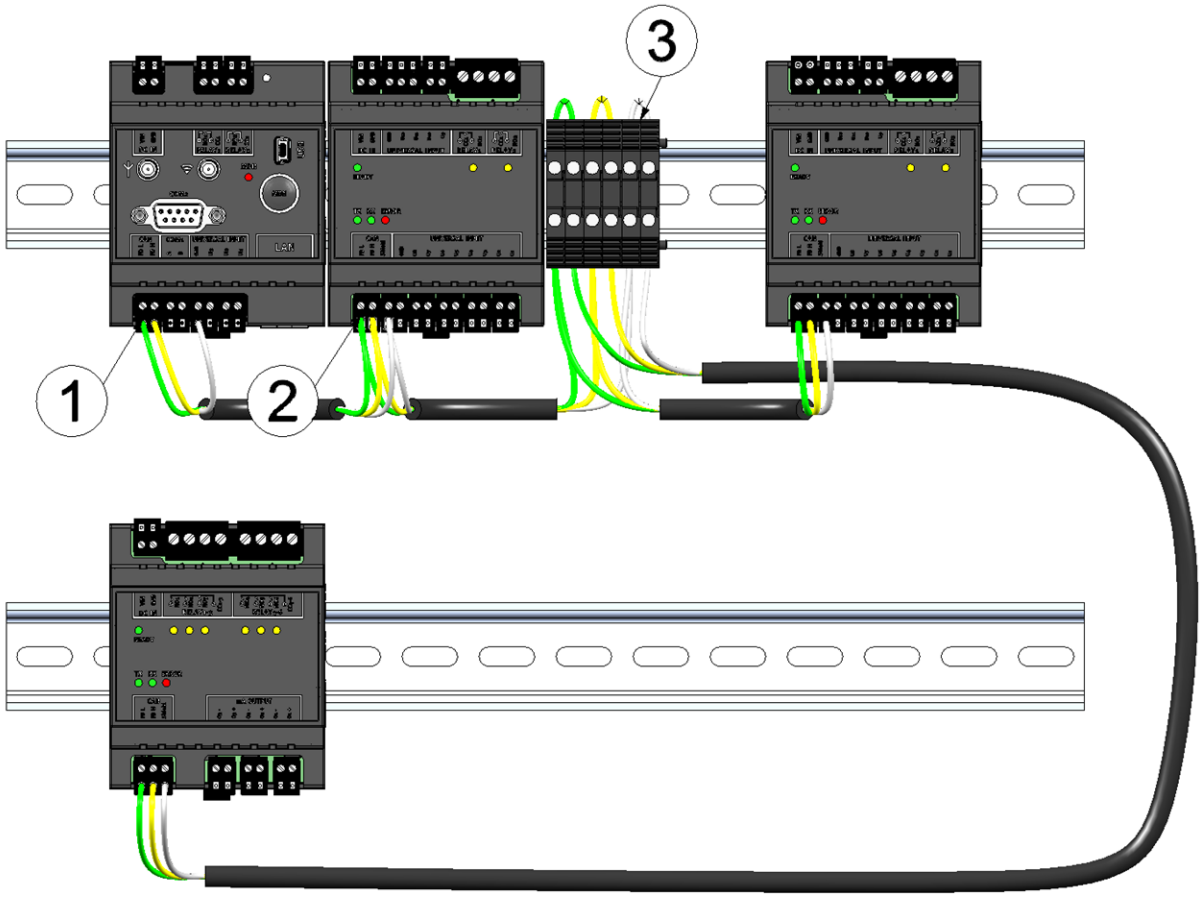
Hutschienen mit montiertem myDatalogC33x und Erweiterungsmodulen

1	myDatalogC33x (120Ω Abschlusswiderstand eingeschaltet)	**4**	myDatalogC3e 3mA/6Rel (2k Abschlusswiderstand eingeschaltet)
2	myDatalogC3e 12UI/2Rel	**5**	Hutschiene
3	Verteilerklemme	**6**	myDatalogC3e 12UI/2Rel (120Ω Abschlusswiderstand eingeschaltet)

3. Verbinden Sie die CAN-Schnittstelle des myDatalogC33x mit jenen der Erweiterungsmodule. Dabei sind jeweils alle "FD L" Klemmen, sowie alle "FD H" Klemmen miteinander zu verbinden. Achten Sie dabei auf Stromlosigkeit!

   Hinweis: *Verwenden Sie für die Verbindung der CAN-Schittstellen ein geschirmtes Kabel. An den Erweiterungsmodulen steht für den Anschluss des Kabelschirms die Klemme "Shield" zur Verfügung. Am myDatalogC33x muss der Kabelschirm mit der GND Klemme verbunden werden. Verwenden Sie eine Twin-Aderendhülsen bzw. Verteilerklemmen, falls Sie mehrere Kabel mit der GND Klemme verbinden wollen.*

   Hinweis: *Da sowohl ammyDatalogC33x als auch an den Erweiterungsmodulen nur eine Klemme für jedes der Signale bereitsteht, sollten Sie, falls Sie mehr als ein Erweiterungsmodul verwenden wollen, Twin-Aderendhülsen bzw. Verteilerklemmen einsetzen.*

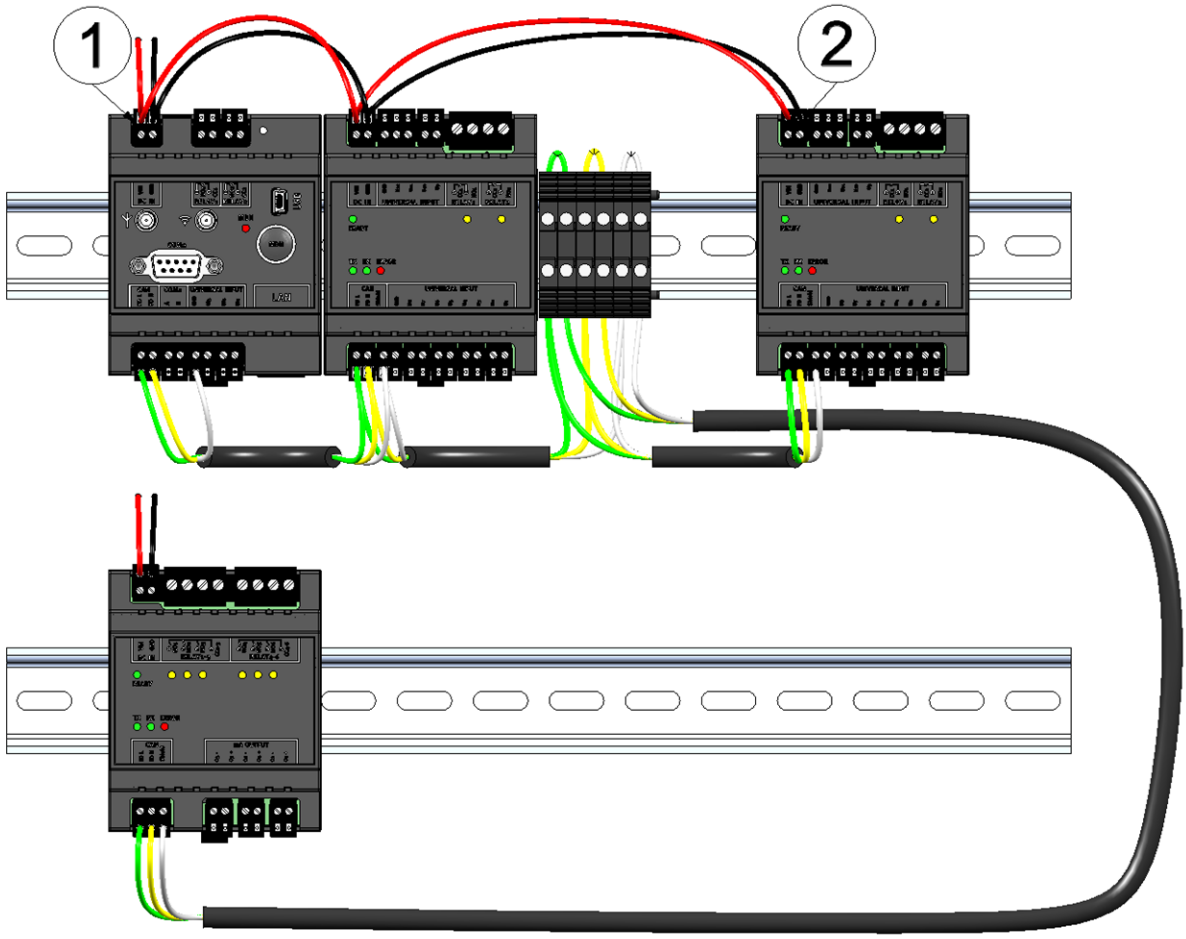


Verkabeln der CAN-Schnittstelle

1 Aderendhülse	**3** Twin-Aderendhülsen (Aderendhülse für 2 Kabel)
2 Verteilerklemme	

4. Verbinden Sie Ihre Sensoren und Aktoren mit den Eingängen und Ausgängen des myDatalogC33x und der Erweiterungsmodule. Achten Sie dabei auf Stromlosigkeit!

5. Verbinden Sie die Kabel zur Versorgung des myDatalogC33x und der Erweiterungsmodule mit den VIN und GND Klemmen. Achten Sie beim Verbinden auf Stromlosigkeit! Verwenden Sie auch hier Twin-Aderendhülsen oder Verteilerklemmen, falls mehr als ein Kabel pro Klemme verbunden werden soll.



Anschließen der Versorgung

1 Twin-Aderendhülse	**2** Aderendhülse

7.4.4 Technische Details zu den Universaleingängen

Hinweis: *Die Universaleingänge sind galvanisch nicht getrennt.*

7.4.4.1 0/4...20mA Modus

Hinweis: *Über 23,96mA wird der betroffene Eingang hochohmig (Sicherheitsabschaltung, um Schäden am Universaleingang zu vermeiden).*

Auflösung	6,36µA
I_{max}	23,96mA
Bürde	96Ω

7.4.4.2 0...2V Modus

Auflösung	610µV
U_{max}	2,5V
Bürde	10k086

7.4.4.3 0...10V Modus

Auflösung	7,97mV
U_{max}	32V
Bürde	4k7

7.4.4.4 Standard Digitalmodi (PWM, Frequenz, Digital, Zähler)

Allgemein	U_{max}	32V
	Low	<0,99V
	High	>2,31V
	Bürde	4k7
PWM	Messbereich	1...99%
	f_{max}	100Hz
	Impulslänge min.	1ms
Frequenz	Messbereich	1...1000Hz
Zähler	Impulslänge min.	1ms

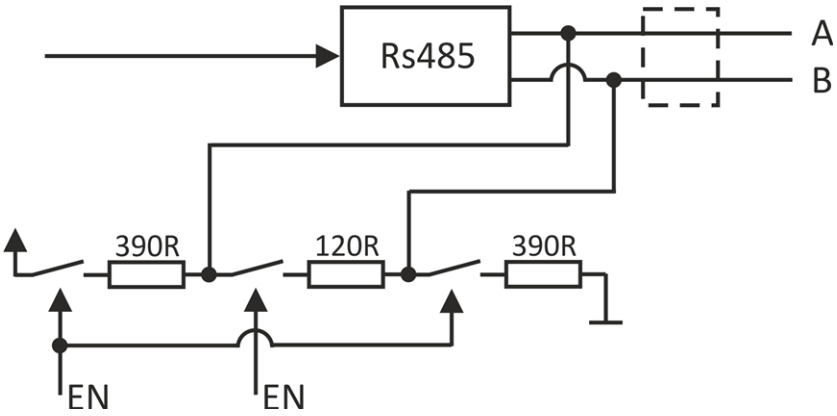
7.4.5 Technische Details zur RS485-Schnittstelle

Hinweis: *Die RS485-Schnittstelle entspricht der Norm EIA-485.*

Die RS485-Schnittstelle des myDatalogC33x verfügt über einen Eingangsgleichtaktbereich, der den gesamten für die RS485 spezifizierten Bereich (-7V...+12V) abdeckt. Höhere Spannungen führen zur Beschädigung der Schnittstelle. Differentielle Signale von mehr als +/-200mV innerhalb des spezifizierten Eingangsgleichtaktbereiches werden korrekt erkannt. Im Sendemodus liegt das Ausgangssignal im Bereich von 1,5...3,3V .

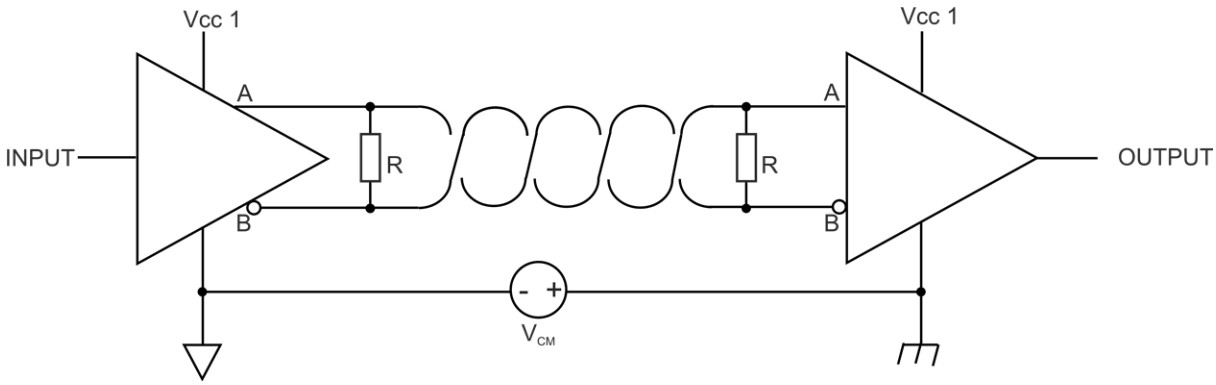
Baudrate	2400-115200
Stopbits	1, 2
Parität	N, E, O
Datenbits	7, 8
Abschlusswiderstand	Aus 120Ω
Klemmwiderstände	Aus 390Ω

Per Device Logic kann der 120Ω Abschlusswiderstand zwischen RS485 A und B zugeschaltet werden. Auch die Klemmwiderstände (Pull up auf RS485 A und Pull down auf RS485 B) können per Device Logic zugeschaltet werden.



Prinzipschaltbild zu den zuschaltbaren Widerständen

Hinweis: *Ergänzende Erklärung zur Verbindung zweier RS485 Busteilnehmer*



Prinzipschaltbild: Verbindung zweier RS485 Busteilnehmer

Ein Problem entsteht, wenn keine Verbindung zwischen den GND-Potentialen von Sender und Empfänger besteht. In diesem Fall entsteht eine Gleichtaktspannung (V_{CM}). Der GND-Potentialunterschied darf max. +/- 7V betragen. Bei höheren Spannungen kommt es zur Beschädigung der Schnittstelle. Kurzzeitige Überspannungen (ESD, EFT und Surge) werden jedoch durch Schutzschaltungen abgefangen.

Anmerkung: Der für RS485 spezifizierte Gleichtakteingangsspannungsbereich von -7V…+12V ergibt sich aus dem max. zulässigen GND-Potentialunterschied (+/- 7V) und dem für RS485 max. zulässigen Ausgangsspannungsbereich von 0…5V.

7.4.6 Technische Details zur CAN-Schnittstelle

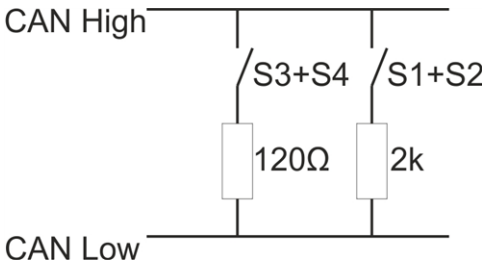
Hinweis: *Die CAN-Schnittstelle des myDatalogC33x ist kompatibel zur Norm ISO-11898, einschließlich der Anforderungen für 24V.*

unterstützte CAN Spezifikation	CAN: V2.0B CAN FD: V1.0
Datenübertragungsrate	CAN: max. 1Mbit/s/ CAN FD: max. 8Mbit/s

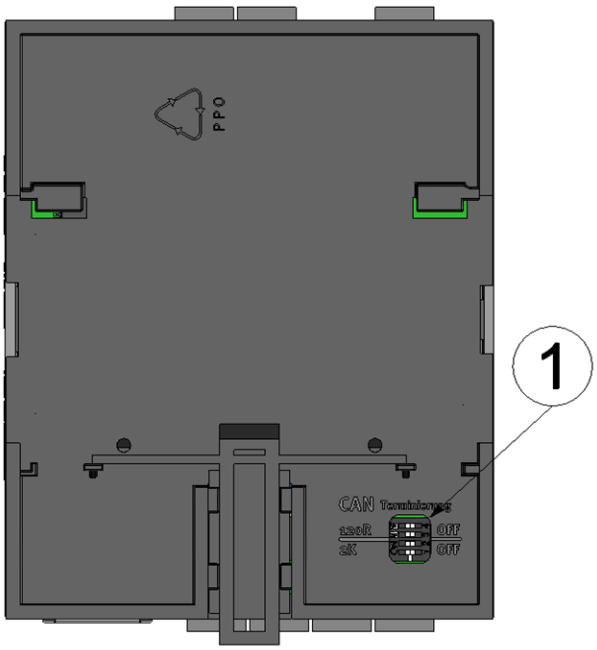
Die CAN-Schnittstelle verfügt über keine galvanische Trennung zwischen dem CAN-Bus und dem CAN-Controller. Die Ausgangstreiber der CAN-Schnittstelle sind allerdings gegen Überlastung geschützt und werden durch einen Kurzschluss nicht beschädigt. Es ist möglich, das myDatalogC33x sowohl am Ende als auch mittels Stichleitungen an den CAN-Bus anzubinden. Wird das myDatalogC33x am Ende des Busses anstelle des vorgeschriebenen 120Ω Abschlusswiderstandes angeschlossen, muss mittels S3 und S4 des Dip-Switch der im Gerät integrierte 120Ω Widerstand zugeschaltet werden. Bei Anbindung via Stichleitungen kann es abhängig von der Länge der Leitung und der gewählten Datenübertragungsrate erforderlich werden, den 2k Widerstand mittels S1 und S2 des Dip-Switch zu aktivieren. Im Auslieferungszustand ist keiner der beiden Widerstände aktiv. Der Dip-Switch zum Ein-/Ausschalten der Abschlusswiderstände befindet sich auf der Rückseite des myDatalogC33x .

Dip-Switches SW1

S1 und S2	2k Abschlusswiderstand zw. CAN High und CAN Low
S3 und S4	120Ω Abschlusswiderstand zw. CAN High und CAN Low



Prinzipschaltbild zu den zuschaltbaren Widerständen



Position des Dip-Switches

1	Dip-Switch zum Ein-/Ausschalten der Abschlusswiderstände für die CAN-Schnittstelle

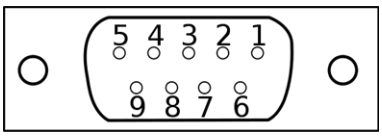
7.4.7 Technische Details zur RS232-Schnittstelle

Hinweis: *Die RS232-Schnittstelle des myDatalogC33x ist kompatibel zur Norm TIA/EIA-232-F.*

Die Ausgangstreiber sind gegen Überlastung geschützt und werden durch einen Kurzschluss auf GND oder +/-15V nicht beschädigt. Die Eingänge sind mit einem 5kΩ Abschlusswiderstand versehen.

Baudrate	2400-115200
Stopbits	1, 2
Parität	N, E, O
Datenbits	7, 8
Fluss-kontrolle	Aus RTS/CTS

Die Richtung der Signale entspricht jener eines DCE (z.B. Modem).



9pol. Sub-D(f)

Belegung des Sub-D Steckers

Pin	Signal	Typ
1	NC	
2	RXD	O (Low: -5,4V ; High: 5,4V)
3	TXD	I (Low: <0,8V; High: >2V)
4	NC	
5	GND	
6	NC	
7	RTS	I (Low: <0,8V; High: >2V)
8	CTS	O (Low: -5,4V ; High:5,4V)
9	5V [1)] max. 750mA	

[1)] Versorgungsspannung (reserviert für Erweiterungen)

Sollte sich an Ihrem Sensor ebenfalls eine SUB-D(f) Buchse befinden, können Sie den als Zubehör erhältlichen Gender changer 9pol. D-Sub male/male (206.684) verwenden. Für den Fall, dass die Anschlusseigenschaften (Übertragungsrichtung der einzelnen Signalleitungen) Ihres Sensors ebenfalls jener eines DCE (z.B. Modem) entsprechen, können Sie den als Zubehör erhältlichen Nullmodemadapter 9pol. D-Sub female/male (206.686) verwenden.

7.4.8 Technische Details zur USB-Schnittstelle

Über die USB-Slave-Schnittstelle wird die Verbindung zu einem PC hergestellt. Dabei ist ausschließlich die Kommunikation mit der webbasierten Entwicklungsumgebung rapidM2M Studio oder dem Konfigurationsprogramm DeviceConfig vorgesehen. Es ist nicht möglich mittels Device Logic auf die USB-Schnittstelle zuzugreifen. Eine detaillierte Beschreibung der webbasierte Entwicklungsumgebung rapidM2M Studio finden Sie im Kapitel "rapidM2M Studio " auf Seite 101. Erläuterungen der Funktionsweise des Konfigurationsprogramms DeviceConfig finden Sie im Kapitel "DeviceConfig " auf Seite 81.

Der Zugang zur webbasierten Entwicklungsumgebung rapidM2M Studio ist im Microtronics Partner Programm, für das Sie sich unter folgender Adresse kostenlos anmelden können, enthalten:

https://partner.microtronics.com

Das Konfigurationsprogramm DeviceConfig steht unter folgender Adresse gratis zum Download bereit:

www.microtronics.com/deviceconfig

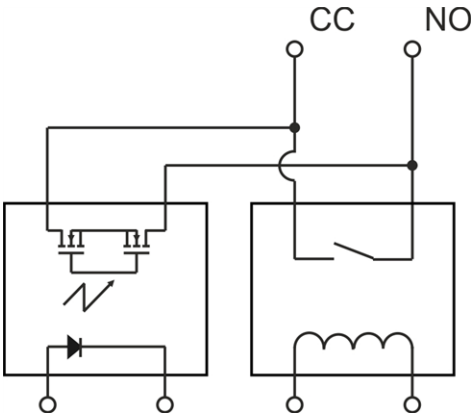
Wichtiger Hinweis: *Sollte die Antenne des Geräts geerdet sein oder mit dem Massepotential eines anderen Objektes verbunden sein (z.B. Montage an einem Schaltschrank), entfernen Sie die Antennen bevor Sie das Gerät mit der USB-Schnittstelle eines PCs verbinden. Andernfalls kann es zu einer Potenzialverschiebung zwischen der Masse der Antenne und der Masse des PCs kommen, wodurch die USB-Schnittstelle des Geräts beschädigt werden kann.*

7.4.9 Technische Details zu den Ausgängen

7.4.9.1 Potentialfreier Schaltkontakt (NO, CC)

> ***Wichtiger Hinweis:*** *Der Benutzer muss dafür sorgen, dass der Strom über den potentialfreien Schaltkontakt2A(Relais-Mode) bzw. 130mA (Optoswitchs-Mode) nicht überschreitet.*

Im Ruhezustand ist der Arbeitskontakt des Relais bzw. des Optoswitchs geöffnet (Normally Open).



Ersatzschaltbild für den potentialfreien Schaltkontakt

Allgemein	Spannungsfestigkeit (CC, NO)	32V AC
	galv. Trennung	1,5kV AC
Relais	U_{max}	32V AC und DC
	I	max. 2A
	P_{max}	64VA , 60W
Optoswitch	U_{max}	32V AC und DC
	$I_{max\ cont.}$	130mA
	$I_{max\ peak}$	400mA (100ms (1 shot), DC)
	P_{max}	500mW
	R_{on}	35Ω
	f_{max}	1000Hz

7.4.10 Technische Details zum integrierten Pufferakku

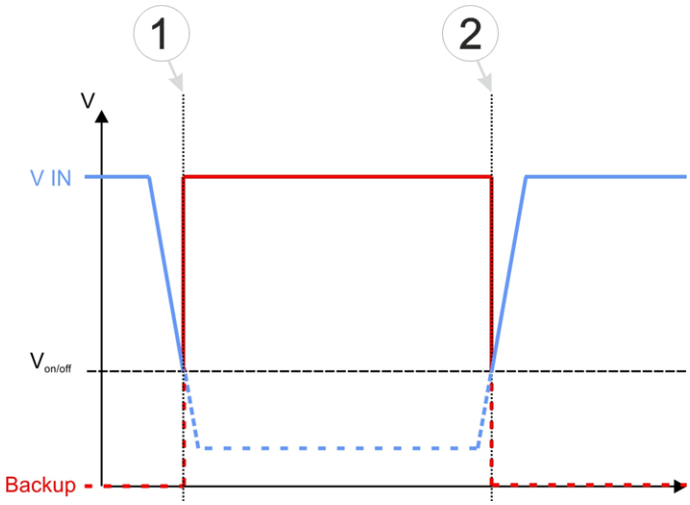
Der integrierte Pufferakku ermöglicht eine applikative Reaktion (z.B. Absetzen einer Meldung und Ausbuchen aus dem Mobilfunknetz) auf den Ausfall der Versorgung. Sobald die Versorgungsspannung unter 8,65V fällt, wird die Versorgung auf den Pufferakku umgeschaltet. Steigt die Versorgungsspannung wieder über 8,65V , wird wieder vom Pufferakku auf die Versorgungsspannung umgeschaltet.

Sobald das myDatalogC33x ausschließlich vom Pufferakku versorgt wird, fallen folgende Baugruppen aus:

- CAN-Schnittstelle (Dadurch ist auch keine Kommunikation mit den Erweiterungsmodulen mehr möglich.)
- Ansteuerung der Relais (d.h. die Arbeitskontakte gehen in den Ruhezustand "NO" über). Die Ansteuerung des Optoswitchs ist davon nicht betroffen.
- 5V Versorgung an COM2 (RS232-Schnittstelle)

Ein durch Hardware realisierter Regler sorgt dafür, dass der Pufferakku nur geladen wird, wenn die Umgebungstemperatur den zulässigen Bereich (0 ...+45°C) nicht verletzt.

V IN	Versorgungsspannung: 9...32VDC (+/-10%)
$V_{on/off}$	Schwelle für das Umschalten zwischen Normalbetrieb und Versorgung über den Pufferakku: 8,65V
Backup	Spannung des Pufferakkus

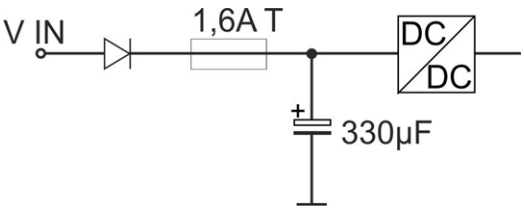


Versorgung des myDatalogC33x

1	• Die Versorgung wird auf den Pufferakku umgeschaltet. • Das Gerät arbeitet normal weiter (abgesehen von den Baugruppen, die bei Versorgung über den Pufferakku ausfallen).
2	• Die Versorgungsspannung überschreitet wieder die Schwelle für das Umschalten auf den Normalbetrieb. • Das myDatalogC33x nimmt den normalen Betrieb wieder auf.

Wichtiger Hinweis: *Wenn der myDatalogC33x nach Ausführen der applikativen Reaktion (z.B. Absetzen einer Meldung und Ausbuchen aus dem Mobilfunknetz) auf den Ausfall der Versorgung nicht über die Device Logic kontrolliert heruntergefahren wird, wird die Device Logic weiter ausgeführt, bis der integrierte Pufferakku tiefentladen ist. Da der integrierte Pufferakku durch die Tiefentladung beschädigt werden könnte, sollte dieser Betriebszustand vermieden werden.*

7.4.11 Technische Details zur Energieversorgung



Prinzipschaltbild der Energieversorgung

V IN	9...32VDC (+/-10%)
Leistungsaufnahme [1)] (ohne Sensoren)	typ. 5W max. 9W
Eingangskapazität	330µF
Sicherung	1,6A T
Verpolungsschutz	Ja

[1)] gilt für den laufenden Betrieb. Durch die Eingangskapazität tritt zum Einschaltzeitpunkt eine Stromspitze auf.

Um im Falle eines Versorgungsspannungsausfalls zuverlässig auf den integrierten Pufferakku umschalten zu können, ist das myDatalogC33x mit einer relativ großen Eingangskapazität (330µF) ausgestattet. Bitte beachten Sie bei der Auswahl des Netzteils, dass dieses in der Lage ist, den nötigen Anlaufstrom zu liefern. Eine Auswahl an kompatiblen Netzteilen finden Sie im Kapitel "Versorgung" auf Seite 241. Der Versorgungsspannungseingang ist des weiteren mit einer Diode zum Schutz vor Verpolung sowie einer 1,6A T Sicherung ausgestattet.

7.4.12 Technische Details zur Systemzeit

Das myDatalogC33x verfügt über eine Hardware Real-Time Clock mit eigenständiger Pufferbatterie, deren zu erwartende Lebensdauer>10 Jahre ist. Die Systemzeit läuft somit weiter, auch wenn die Power Supply Unit entnommen wird. Das bedeutet, dass nach der Wiederinbetriebnahme sofort gültige Zeitstempel für die Mess- und Log-Daten erzeugt werden können. Bei jeder Verbindung zum myDatanet-Server erfolgt zudem eine automatische Synchronisation der Systemzeit mit dem Server.

Kapitel 8 Inbetriebnahme

8.1 Hinweise an den Benutzer

Bevor Sie den myDatalogC33x anschließen und in Betrieb nehmen, sind die folgenden Benutzerhinweise unbedingt zu beachten!

Dieses Handbuch enthält alle Informationen, die zum Gebrauch des Gerätes erforderlich sind.

Es wendet sich an technisch qualifiziertes Personal, welches über einschlägiges Wissen im Bereich der Messtechnik verfügt.

Um die einwandfreie Funktion des myDatalogC33x zu gewährleisten, muss dieses Handbuch sorgfältig gelesen werden.

Bei eventuellen Unklarheiten oder Schwierigkeiten in Bezug auf Montage, Anschluss oder Konfiguration wenden Sie sich an Microtronics Engineering GmbH (siehe "Kontaktinformationen" auf Seite 253).

8.2 Mitgeltende Unterlagen

Für die Installation, Inbetriebnahme und den Betrieb des Gesamtsystems werden neben dieser Bedienungsanleitung möglicherweise zusätzliche Anleitungen oder technische Beschreibungen benötigt.

Diese Anleitungen liegen den jeweiligen Zusatzgeräten oder Sensoren bei bzw. stehen auf der Microtronics - Webseite zum Download bereit.

8.3 Allgemeine Grundsätze

Die Inbetriebnahme des gesamten Messsystems darf erst nach Fertigstellung und Prüfung der Installation erfolgen. Vor der Inbetriebnahme ist das Studium des Handbuches erforderlich, um fehlerhafte oder falsche Konfiguration auszuschließen.

Machen Sie sich mit Hilfe des Handbuches mit der Bedienung des myDatalogC33x und den Eingabemasken des myDatanet-Servers vertraut, bevor Sie mit der Konfiguration beginnen.

8.4 Inbetriebnahme des Systems

> ***Hinweis:*** *Es empfiehlt sich, den myDatalogC33x zuerst im Büro in Betrieb zu nehmen bevor Sie das Gerät am Einsatzort fix montieren. Dabei sollten Sie gleich eine Site für den späteren Betrieb am myDatanet-Server anlegen (siehe "Anlegen der Site" auf Seite 97) und eine Konfiguration für die Site (inkl. Data Descriptor und Device Logic) festlegen (siehe "Messstellenkonfiguration" auf Seite 74). Wenn Sie die Site auf Basis einer IoT Applikation (siehe "Benutzerhandbuch für myDatanet-Server " 206.886) erstellen, werden der Data Descriptor und die Device Logic aus der IoT Applikation übernommen und müssen nicht gesondert angegeben werden. Nutzen Sie die Gelegenheit sich in geordneter Umgebung mit den Funktionen des Geräts vertraut zu machen. Sie können auch geeignete Testsignale zum Simulieren der Sensoren verwenden, um die Konfiguration des myDatalogC33x bereits vor der eigentlichen Inbetriebnahme optimal fest zu legen. Dadurch reduzieren Sie den Zeitaufwand bei der Installation vor Ort auf das Minimum.*

Folgende Arbeiten sollten Sie im Büro erledigen bevor Sie sich zum Einsatzort des Geräts begeben:

1. Legen Sie, falls erforderlich, einen Kunden am myDatanet-Server an (siehe "Benutzerhandbuch für myDatanet-Server " 206.886).
2. Legen Sie innerhalb des gewünschten Kunden eine Site / Applikation für den Betrieb am myDatanet-Server an (siehe "Anlegen der Site" auf Seite 97).

   Hinweis: *Für den Betrieb des myDatalogC33x muss eine Site des Typs "rapidM2M " oder eine Site auf Basis einer zum Site Typ "rapidM2M " kompatiblen IoT Applikation erstellt werden.*

3. Konfigurieren Sie die erstellte Site / Applikation entsprechend Ihren Anforderungen (siehe "Messstellenkonfiguration" auf Seite 74). Wurde die Site nicht auf Basis einer IoT Applikation erstellt, müssen Sie auch noch den Data Descriptor und die Device Logic über den Konfigurationsabschnitt "Steuerung" festlegen (siehe "Steuerung" auf Seite 75).
4. Schließen Sie die Antenne an (siehe "Anschluss der GSM-Antenne" auf Seite 50).
5. Lösen Sie einen Verbindungsaufbau aus, damit die Konfiguration der Site / Applikation zum myDatalogC33x übertragen wird. Falls noch keine Device Logic ins Gerät geladen wurde, erreichen Sie dies durch Herstellen der Spannungsversorgung (siehe "Anschluss der Sensoren, der Aktoren und der Versorgung" auf Seite 46). Wurde bereits eine Device Logic ins Gerät geladen, führen Sie die in der Device Logic vorgesehenen Operationen zum Auslösen des Verbindungsaufbaus aus.

   Hinweis: *Diesen Schritt können Sie auch überspringen, da bei der Installation vor Ort ebenfalls eine Verbindung ausgelöst werden muss, wodurch die Konfiguration dann zu diesem Zeitpunkt zum myDatalogC33x übertragen wird.*

6. Trennen Sie anschließend die Verkabelung der Versorgungsspannung, möglichst im stromlosen Zustand, vom Gerät. Stellen Sie sicher, dass der myDatalogC33x vollständig deaktiviert ist. Drücken Sie dazu die Reset-Taste direkt am Gerät, falls Sie in Ihrer Device Logic keine Routine für das kontrollierte Herunterfahren des Systems nach dem Trennen der Versorgungsspannung vorgesehen haben.
7. Entfernen Sie die Antenne wieder.

Folgende Arbeiten werden direkt am Einsatzort des Geräts durchgeführt:

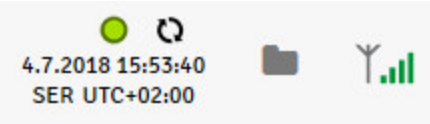
8. Führen Sie alle im Kapitel "Anschluss der Sensoren, der Aktoren und der Versorgung" auf Seite 46 beschriebenen Schritte durch.
9. Überprüfen Sie, ob die Verbindung zum myDatanet-Server korrekt funktioniert (siehe "Kommunikation mit dem Gerät testen" auf Seite 70).

8.5 Kommunikation mit dem Gerät testen

1. Legen Sie eine Site / Applikation für den Betrieb am myDatanet-Server an (siehe "Anlegen der Site" auf Seite 97).

   Hinweis: *Für den Betrieb des myDatalogC33x muss eine Site des Typs "rapidM2M " oder eine Site auf Basis einer zum Site Typ "rapidM2M " kompatiblen IoT Applikation erstellt werden.*

2. Konfigurieren Sie die erstellte Site / Applikation entsprechend Ihren Anforderungen (siehe "Messstellenkonfiguration" auf Seite 74). Wurde die Site nicht auf Basis einer IoT Applikation erstellt, müssen Sie auch noch den Data Descriptor und die Device Logic über den Konfigurationsabschnitt "Steuerung" festlegen (siehe "Steuerung" auf Seite 75).
3. Schließen Sie die Antenne an (siehe "Anschluss der GSM-Antenne" auf Seite 50).
4. Lösen Sie einen Verbindungsaufbau aus. Falls noch keine Device Logic ins Gerät geladen wurde, erreichen Sie dies durch Herstellen der Spannungsversorgung (siehe "Anschluss der Sensoren, der Aktoren und der Versorgung" auf Seite 46). Wurde bereits eine Device Logic ins Gerät geladen, führen Sie die in der Device Logic vorgesehenen Operationen zum Auslösen des Verbindungsaufbaus aus.
5. Warten Sie bis in der Messgeräteliste angezeigt wird, dass das Gerät mit dem Server verbunden ist (rotierende Pfeile).



   Mit Ausnahme der Verbindungsart „online“ (siehe "rM2M_TxSetMode()") ist die Zeit während der den myDatalogC33x mit dem Server verbunden ist sehr kurz. Daher kann auch geprüft werden, ob der Zeitstempel der letzten Verbindung (unter dem grünen Statussymbol) aktualisiert wurde.

Die folgenden Schritte sind nur erforderlich, wenn Sie auch gleich die Messwerterfassung und die Datenübertragung testen wollen.

6. Führen Sie alle im Kapitel "Anschluss der Sensoren, der Aktoren und der Versorgung" auf Seite 46 beschriebenen Schritte durch. Dazu gehört auch das Verdrahten der Sensoren.

   Wichtiger Hinweis: *Alle Verkabelungsarbeiten sollten im stromlosen Zustand erfolgen!*

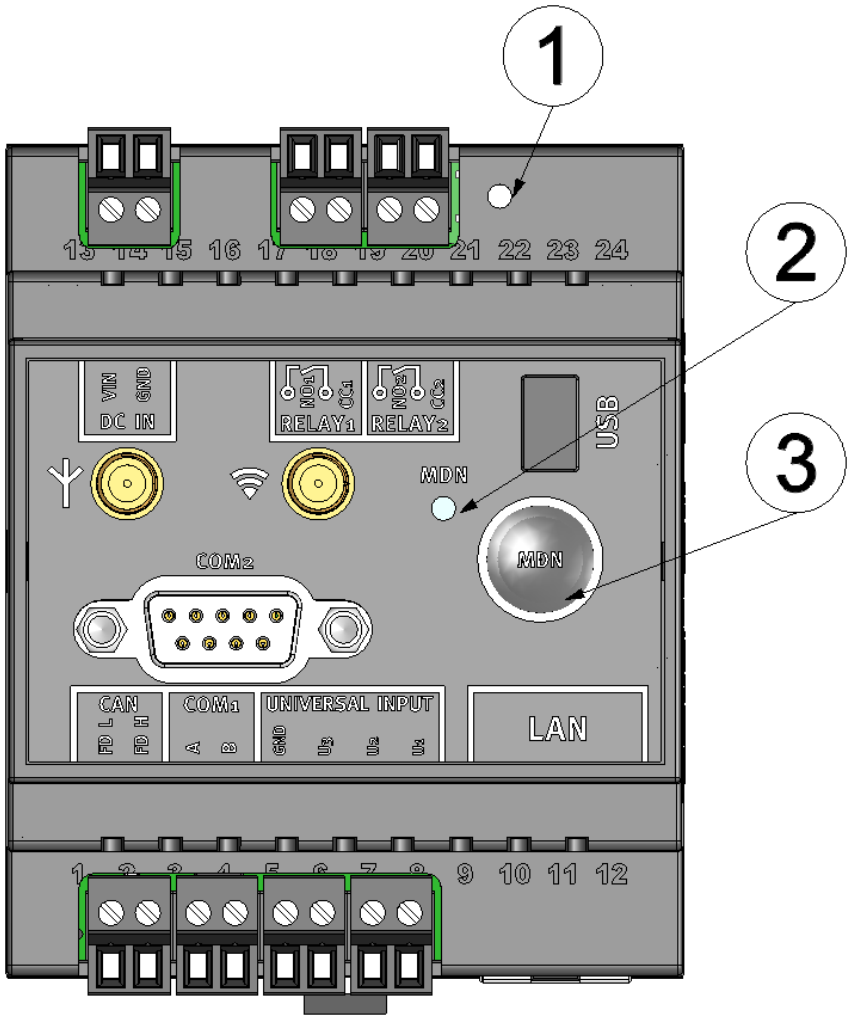
7. Für die Überprüfung der Datenübertragung können Sie die "Auswertungen" des myDatanet-Server verwenden (siehe"Benutzerhandbuch für myDatanet-Server " 206.886). Dazu ist allerdings die Konfiguration des Data Descriptor (siehe "Data Descriptor " auf Seite 213) erforderlich.
8. Nachdem Sie die erforderlichen Vorbereitungen getroffen haben, lösen Sie, falls Sie dies in Ihrer Device Logic vorgesehen haben, direkt am Gerät eine Übertragung aus. Haben Sie keine Möglichkeit zum Auslösen einer Übertragung vorgesehen, warten Sie bis zur nächsten planmäßigen Datenübertragung.
9. Werten Sie die ankommenden Daten aus.

Kapitel 9 Benutzerschnittstellen

Die Konfiguration des myDatalogC33x erfolgt über das Web-Interface am myDatanet-Server (siehe "Benutzerschnittstelle am myDatanet-Server" auf Seite 74), dessen Web-Adresse Sie von Ihrem zuständigen Vertriebspartner erhalten.

9.1 Benutzerschnittstelle am myDatalogC33x

9.1.1 Bedienelemente



Bedienelemente

1	Reset Taster	**3**	MDN Taste (frei verwendbar, Auswertung durch die Device Logic)
2	RGB-LED (frei verwendbar, gesteuert durch die Device Logic)		

9.2 Benutzerschnittstelle am myDatanet-Server

9.2.1 Messstellenkonfiguration

> ***Hinweis:*** *Abhängig vom jeweiligen Benutzerlevel sind einige der in den folgenden Unterkapiteln erwähnten Konfigurationsfelder unter Umständen ausgeblendet. Wenden Sie sich in diesem Fall an den Administrator des myDatanet-Servers.*

Die Eingabemaske zur Konfiguration der Messstelle erreichen Sie durch Klicken auf den Messstellennamen in der Messstellenliste (siehe "Benutzerhandbuch für myDatanet-Server " 206.886).

9.2.1.1 Site

Kunde
gibt an, welchem Kunden die Site zugeordnet ist

 **-Symbol**
Site einem anderen Kunden zuweisen

Name
Bezeichnung der Site (nicht relevant für die Geräte- oder Datenzuordnung) [2-50 Zeichen]

Gerät S/N
Seriennummer des Geräts, das mit der Site verknüpft ist (Gerätezuordnung!)

Applikation
Name der IoT Applikation, auf deren Basis die Site erstellt wurde

Applikation Version
Versionsnummer der IoT Applikation, die aktuell auf der Site installiert ist. Stimmen die Versionsnummern der Site und jene der im Gerät installierten Device Logic nicht überein, wird die Versionsnummer der im Gerät installierten Device Logic zusätzlich zur Versionsnummer der Site angezeigt.

Tags
Liste der Tags, die der Site bereits zugewiesen sind. Durch einen Klick auf das Kreuz neben der Bezeichnung des Tags kann diese Zuweisung wieder aufgehoben werden. Die Eingabemaske zur Zuweisung der Tags kann durch Klicken auf das Plus-Symbol geöffnet werden.

9.2.1.2 Kommentar

Kommentar
freies Kommentarfeld (wird auch unterhalb des Gerätetyps in der Liste der Sites/Applikationen angezeigt)

9.2.1.3 Steuerung

Hinweis: *Wurde die Messstelle auf Basis einer IoT Applikation (siehe "Benutzerhandbuch für myDatanet-Server " 206.886) erstellt, ist dieser Konfigurationsabschnitt nicht sichtbar.*

Device Logic Typ	aus	Device Logic deaktiviert	
	Pawn	Aktiviert die Abarbeitung der Device Logic	
Device Logic Quelle	Pawn source code	Device Logic	Eingabefenster zum Editieren der Device Logic, die in das myDatalogC33x geladen wird (siehe "Device Logic" auf Seite 107)
	Hochladen eines kompilierten Scripts	Datei hochladen	Auswahl des Device Logic Binary-Files (*.amx), das auf den myDatanet-Server hochgeladen und bei der nächsten Verbindung in das myDatalogC33x geladen wird. Der Dateipfad wird nur solange angezeigt, solange die Eingabemaske zur Konfiguration der Messstelle nicht geschlossen wurde.
Data Descriptor	Eingabefenster zum Konfigurieren des Data Descriptor (siehe "Data Descriptor " auf Seite 213)		

9.2.1.4 Konfiguration 0 - Konfiguration 9

Hinweis: *Diese Konfigurationsabschnitte sind nur sichtbar, wenn mittels des Data Descriptor (siehe "Data Descriptor " auf Seite 213) der logische Aufbau des entsprechenden Konfiguratiosdatenblocks definiert wurde. Auch die Bezeichnung des Konfigurationsabschnitts wird über den Data Descriptor festgelegt.*

Diese Konfigurationsabschnitte ermöglichen es die Parameter aus den vom Kunden frei definierbaren, voneinander unabhängigen Speicherblöcken über die Oberfläche des myDatanet-Servers verändern bzw. anzeigen zu können. Dazu muss der logische Aufbau der Konfiguratiosdatenblöcke mit Hilfe des Data Descriptor (siehe "Data Descriptor " auf Seite 213) definiert werden.

9.2.1.5 Alarmierung

Quittierung	Standard	Für die Entscheidung, ob die Alarme automatisch oder manuell quittiert werden müssen, wird die globale Servereinstellung herangezogen.
	automatisch	Alarme werden automatisch quittiert, sobald alle Benachrichtigungen versendet wurden. Wurden auch SMS versendet, die einen Tarif mit Sendebestätigungsfunktion haben, so wird mit der Quittierung auf die Sendebestätigung gewartet.
	manuell	Alarme müssen durch den Anwender quittiert werden.
Transfervolumen	Standard	Die Einstellung für den Transfervolumenalarm wird von der globalen Servereinstellung übernommen.
	aus	Der Transfervolumenalarm ist deaktiviert.
	individuell	Die Schwelle, bei der der Transfervolumenalarm ausgelöst werden soll, kann in das nebenstehende Feld in KiB eingegeben werden.
Offline Alarm nach	Alarmierung, falls sich das Instrument länger als die eingestellte Zeit nicht meldet (00:00 Alarmierung deaktiviert).	
Bezeichnung Benutzeralarm 1	frei wählbare Bezeichnung für den benutzerdefinierten Alarm 1. Wird von einem mit der Site verknüpften Gerät der benutzerdefinierte Alarm 1 gemeldet, nutzt der Server diesen Text zur Signalisierung des Alarms. Selbiges gilt für den benutzerdefinierten Alarm 2 und 3.	
Bezeichnung Benutzeralarm 2	frei wählbare Bezeichnung für den benutzerdefinierten Alarm 2	
Bezeichnung Benutzeralarm 3	frei wählbare Bezeichnung für den benutzerdefinierten Alarm 3	

9.2.1.6 Grundeinstellungen

Zeitzone	Regionseinstellungen (nicht relevant für Rohmessdaten, da diese in UTC gespeichert werden)	
Sommerzeit	Konfiguration für die automatische Zeitumstellung	
	standard	Die Konfiguration für die Zeitumstellung wird von der globalen Servereinstellung übernommen.
	aus	automatische Zeitumstellung deaktiviert
	USA	vordefinierte Einstellung für den amerikanischen Raum
	EU	vordefinierte Einstellung für den europäischen Raum
Standard Auswertung	Auswahl der Auswertung, die durch einen Klick auf den Gerätelink in den Karten geladen wird	
	aus	Es wird keine Auswertung geladen.
	"Name einer Auswertung"	Die ausgewählte Auswertung wird geladen.
Auswertungs-Vorlage	Auswahl der Auswertungs-Vorlage, die zur Darstellung der Daten verwendet wird, wenn auf das Symbol zur Anzeige der Messdaten, das sich in der Liste der Sites/Applikationen befindet, geklickt wird. In der Dropdown-Liste werden nur jene Auswertungs-Vorlagen angezeigt, bei denen der Site-/Applikationstyp der ersten Wildcard kompatibel zur Site/Applikation ist, die aktuell bearbeitet wird. Das Symbol zur Anzeige der Messdaten wird in der Liste der Sites/Applikationen nur dann angezeigt, wenn eine Auswertungs-Vorlage ausgewählt wurde.	
	(nicht zugeordnet)	Das Symbol zur Anzeige der Messdaten wird in der Liste der Sites/Applikationen nicht angezeigt.
	"Name einer Auswertungs-Vorlage"	Name der Auswertungs-Vorlage, die zur Darstellung der Messdaten verwendet wird
Änderungsprotokoll Konfiguration	Auswahl, welche Änderungen an den Konfigurationen protokolliert werden sollen	
	web api	Änderungen, die über die Serveroberfläche oder die REST-API vorgenommen wurden, werden protokolliert.
	web device api	Änderungen, die über die Serveroberfläche, vom Gerät selbst oder über die REST-API vorgenommen wurden, werden protokolliert.

9.2.1.7 FTP-Export Einstellungen

Hinweis: *Dieser Konfigurationsabschnitt ist nur sichtbar, wenn die Lizenz "FTP Agent Extended" für den myDatanet-Server freigeschaltet wurde.*

FTP Export Profil	aus	FTP Export deaktiviert
	"Name eines FTP Export Profils"	Liste mit den FTP-Export-Profilen, die am Server angelegt wurden (zum Anlegen eines FTP-Export-Profils siehe "Benutzerhandbuch für myDatanet-Server " 206.886).
Einstellungen des gewählten Profils	zeigt eine Übersicht der wichtigsten Parameter des ausgewählten FTP-Export-Profils an	
FTP Verzeichnis	ermöglicht es, das Standardverzeichnis des ausgewählten FTP-Export-Profils zu überschreiben [0-100 Zeichen]	
letzter Export	Zeitstempel des letzten FTP Exportes	

9.2.2 Gerätekonfiguration

Hinweis: *Abhängig vom jeweiligen Benutzerlevel sind einige der in den folgenden Unterkapiteln erwähnten Konfigurationsfelder unter Umständen ausgeblendet. Wenden Sie sich in diesem Fall an den Administrator des myDatanet-Servers.*

Die Eingabemaske zur Konfiguration des Geräts erreichen Sie durch Klicken auf die Seriennummer in der Liste der Sites/Applikationen (siehe "Benutzerhandbuch für myDatanet-Server " 206.886) oder durch Klicken auf den Gerätenamen in der Messgeräteliste (siehe "Benutzerhandbuch für myDatanet-Server " 206.886).

9.2.2.1 Kommentar

Kommentar

freies Kommentarfeld (wird auch unterhalb des Namens der Site in der Liste der Sites/Applikationen angezeigt)

9.2.2.2 Messgerät

Kunde	Name des Kunden, dem das Messgerät zugeordnet ist
Tags	Liste der Tags, die dem Messgerät bereits zugewiesen sind. Durch einen Klick auf das Kreuz neben der Bezeichnung des Tags kann diese Zuweisung wieder aufgehoben werden. Durch Klicken auf das Plus-Symbol wird die Eingabemaske zur Zuweisung der Tags geöffnet. Diese ermöglicht sowohl die Zuweisung vorhandener als auch die Erstellung neuer Tags.
Seriennummer	Seriennummer des Geräts
Geräteklasse	Damit ein Gerät mit einer Site verbunden werden kann, müssen die Geräteklasse der Site und die des Geräts übereinstimmen. Die Geräteklasse kann nach dem Anlegen des Geräts über die Serveroberfläche nur bis zur ersten Verbindung des Geräts mit dem Server verändert werden. Sollte beim Anlegen des Geräts eine Geräteklasse eingestellt werden, die nicht mit der tatsächlichen Geräteklasse des Geräts übereinstimmt, wird diese bei der ersten Verbindung automatisch korrigiert.

Telefonnummer	Telefonnummer der SIM-Karte. An diese Nummer werden die Steuer-SMS (z.B. Wakeup) gesendet. Format: +43555837465	
Geräte Flags	zusätzliche Information zur Geräteklasse (für interne Verwendung)	
Firmware Version	aktuell installierte Softwareversion des Messcontrollers	
Letzter Verbindungsaufbau	jeweils der letzte Zeitstempel der betreffenden Operation	
Letzter Wakeup		
Letzter Verbindungsabbau		
Letzter Übertragungsfehler		
Letzte Aloha Verbindung		
Wakeup SMS Anzahl	Anzahl der seit der letzten Verbindung an dieses Gerät gesendeten Wakeup-SMS. Bei jeder erfolgreich hergestellten Verbindung wird dieser Zähler zurückgesetzt.	
Device Logic Sync	Produktiv	Stimmen die im Gerät installierte und die am Server gespeicherte Device Logic nicht überein, wird die am Server gespeicherte Device Logic in das Gerät geladen.
	Entwicklung (sync)	Es erfolgt eine Synchronisation der Device Logic zwischen dem Gerät und dem Server. Dabei wird jenes mit dem aktuellsten Zeitstempel zur jeweils anderen Stelle übertragen.
	Entwicklung (no sync)	Es erfolgt keine Synchronisation der Device Logic zwischen dem Gerät und dem Server
Firmware Update	aus	Firmware Update ist deaktiviert
	ein	Sobald eine neue Version des ausgewählten Firmware-Typs vorhanden ist, wird diese sofort installiert.
	auch wenn tag nicht vorhanden	Firmware wird auch ans Gerät übertragen, wenn das Gerät den aktuellen Firmwarestand nicht an den Server übermittelt hat (NICHT EMPFOHLEN!).
	Downgrade erlauben	ermöglicht es, eine ältere Firmwareversion als die im Gerät vorhandene zu installieren (NICHT EMPFOHLEN!)
	einmalig	Führt einmalig ein Firmware Update durch. Ist keine neue Firmware verfügbar oder wurde die Firmware erfolgreich installiert, wird das Firmware Update automatisch auf "aus" geschaltet.
	ignorieren	Das Firmware Update ist deaktiviert und auf verfügbare Firmware Updates wird nicht hingewiesen.

Firmware Typ	Released	Nur Firmwareversionen bei denen sowohl interner Test als auch Feldtest erfolgreich waren, werden installiert (Fehlfunktionen nahezu ausgeschlossen).
	Release Candidate	Nur Firmwareversionen bei denen der interne Test erfolgreich war, werden installiert (Fehlfunktionen nicht ausgeschlossen).
	Beta Release	Auch Firmwareversionen bei denen noch nicht alle internen Tests erfolgreich abgeschlossen sind, werden installiert (Fehlfunktionen durchaus möglich).
Identifikation	String, der die im Gerät verbaute Hardwareplattform und die dazugehörige Hardwareversion angibt (d.h. die rapidM2M Modulidentifikation)	
Prod. Rev.	Produktrevision des myDatalogC33x	

9.2.2.3 GPRS

SIM Tarif
ausgewählter SIM-Tarif

Kapitel 10 DeviceConfig

10.1 Allgemein

Das Konfigurationsprogramm DeviceConfig steht unter folgender Adresse gratis zum Download bereit:

www.microtronics.com/deviceconfig

Es handelt sich um ein Tool zur Konfiguration, Wartung, Fehleranalyse und Synchronisation. Es ist mit allen myDatanet Geräten, die über eine USB-Schnittstelle, eine Wireless M-Bus-Schnittstelle oder eine Bluetooth Low Energy Schnittstelle verfügen, kompatibel.

Die Anforderungen bezüglich Konfiguration und Wartung variieren je nach Typ des Geräts. Um eine einfache und intuitive Bedienung zu ermöglichen, passt sich die Benutzeroberfläche des DeviceConfig daher automatisch an das jeweilig verbundene Gerät an. Neben den Standardfunktionen bietet das Tool auch Unterstützung für gerätespezifische Prozesse (bspw. Kalibrierung, 0-Punktabgleich).
Das DeviceConfig ermöglicht es Ihnen folgende Aufgaben durchzuführen:

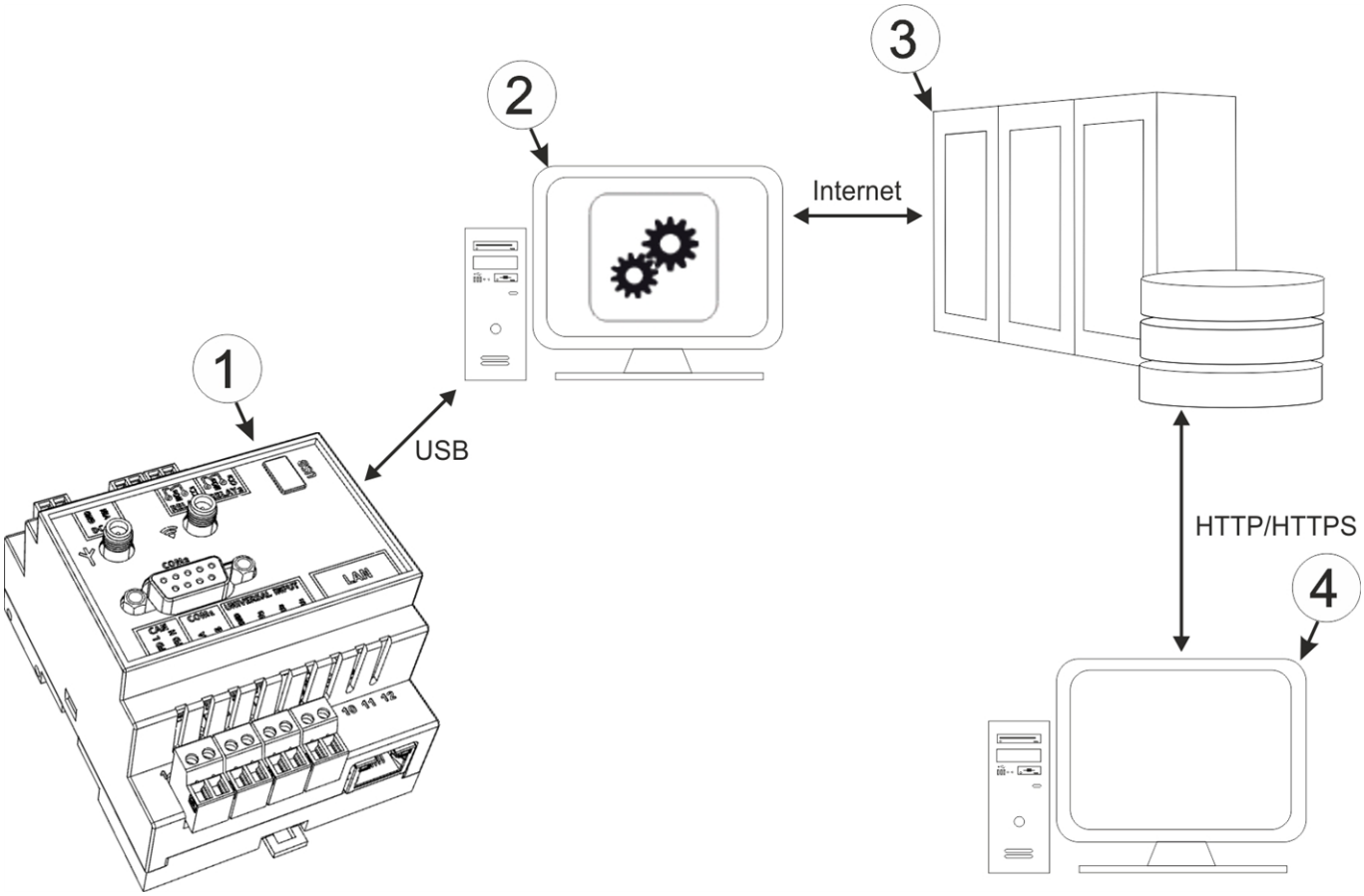
- Synchronisation von Messdaten und Konfiguration zwischen Gerät und Server (speziell für Geräte ohne GSM/GPRS Modem)
- Grundlegende Konfiguration des Geräts (z.B. Mess- und Übertragungsintervall)
- Auslesen und Analyse des Gerätelogs
- Kalibrierung, Trimmung und 0-Punktabgleich (spezielle Kenntnisse bzw. Passwort erforderlich)
- Aktualisierung der Firmware

10.2 Voraussetzungen

Schnittstellen	1 x USB
Betriebssystem	Win XP Windows Vista Windows 7 Windows 8 Windwos 10
Internetverbindung	empfohlen
Benötigter Speicherplatz	ca. 50MB

10.3 Funktionsprinzip

Die folgende Beschreibung bezieht sich speziell auf die Verwendung des Konfigurationsprogramms DeviceConfig in Verbindung mit dem myDatalogC33x .



Funktionsprinzip

1	myDatalogC33x	**3**	myDatanet-Server
2	PC mit installiertem Konfigurationsprogramm DeviceConfig	**4**	Client, der mittels Web-Browser auf die Oberfläche des myDatanet-Servers zugreift

> ***Wichtiger Hinweis:*** *Bei der USB-Schnittstelle handelt es sich um eine Serviceschnittstelle, die durch den im Lieferumfang enthaltenen Dichtstopfen vor Verschmutzung geschützt werden muss, wenn sie nicht verwendet wird.*

Das Konfigurationsprogramm DeviceConfig kommuniziert per USB-Verbindung direkt mit dem myDatalogC33x . Zu den vom Konfigurationsprogramm DeviceConfig bereitgestellten Funktionen gehören:

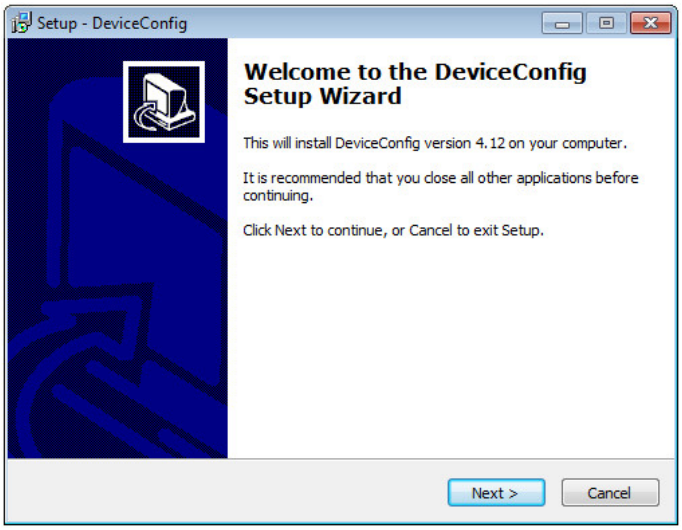
- Auslesen und Analyse des Gerätelogs (siehe "Karteireiter "Log"" auf Seite 89)
- Aktualisierung der Firmware (siehe "Karteireiter "Firmware"" auf Seite 91)

10.4 Installation

Das folgende Kapitel beschreibt den Installationsprozess unter Windows 7.

1. Führen Sie die Datei *"InstDeviceConfig.exe"* aus, um den Installationsprozess zu starten.

   Hinweis: *Verbinden Sie das Gerät bzw. den USB BLE-Adapter (300685) erst nach Abschluss des Installationsprozesses mit Ihrem PC, da die benötigten Treiber erst während dieses Vorgangs installiert werden.*

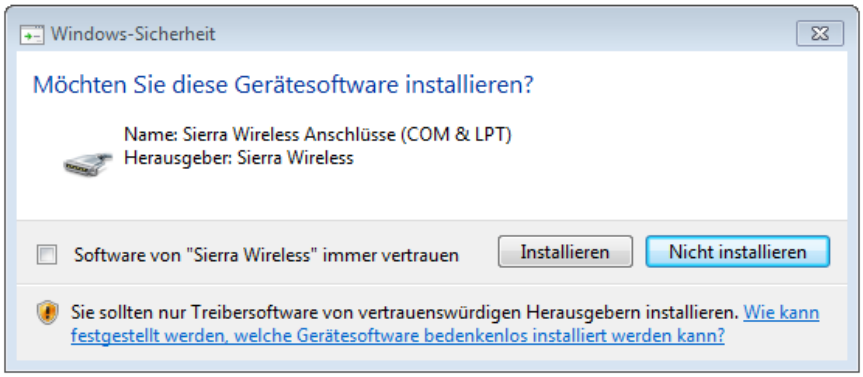


DeviceConfig Setup Wizard

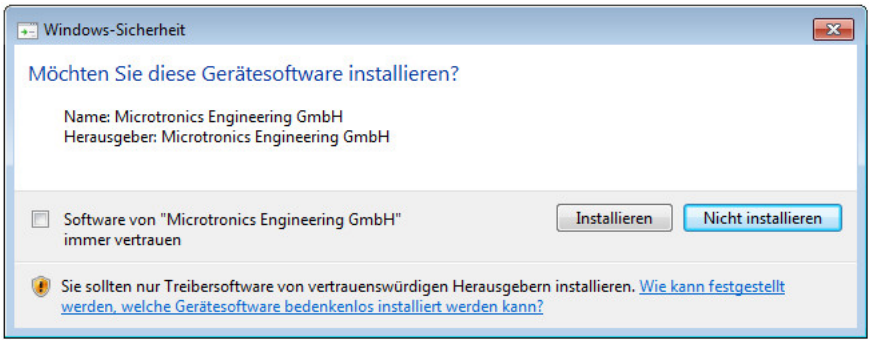
2. Folgen Sie den Anweisungen des Setup Wizzards bis Sie zu der folgenden Ansicht gelangen. Für den ordnungsgemäßen Betrieb müssen die folgenden Treiber zwingend installiert werden.

Installation der USB-Treiber für die Geräte

Installation des Treibers für den USB BLE-Adapter

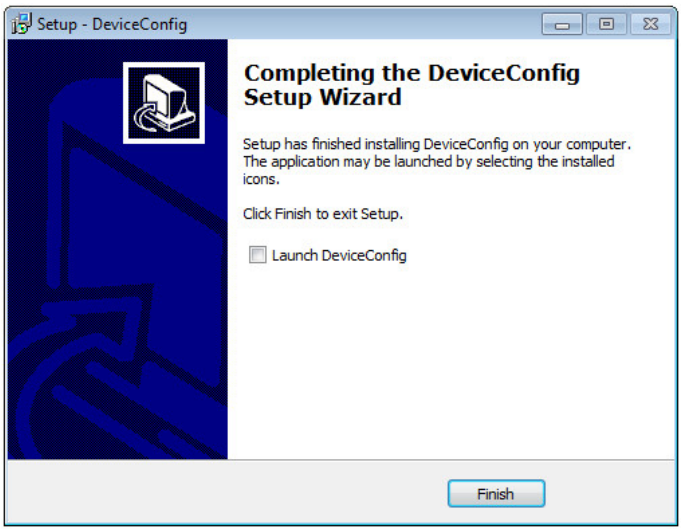


Installation der USB-Treiber für Geräte auf M1-Basis



Installation der USB-Treiber für Geräte auf M2/M3-Basis

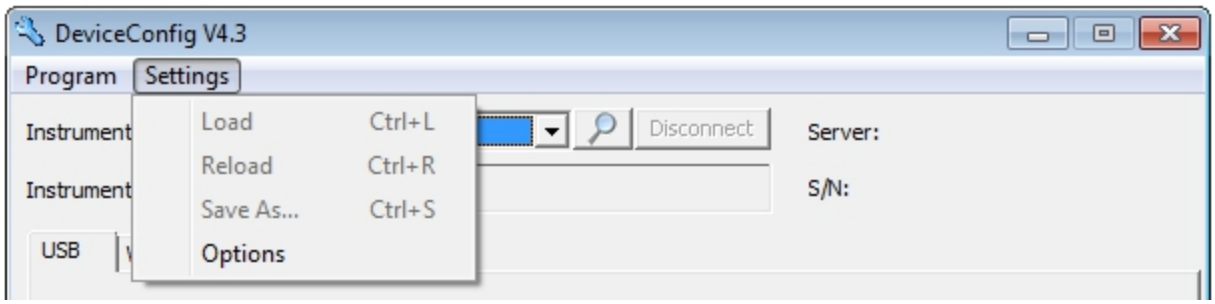
3. Wenn Sie schließlich zur folgenden Ansicht gelangen, schließen Sie den Installationsvorgang durch Klicken auf den Button *"Finish"* ab.



Setup abschließen

10.5 Menü des DeviceConfig

10.5.1 Settings

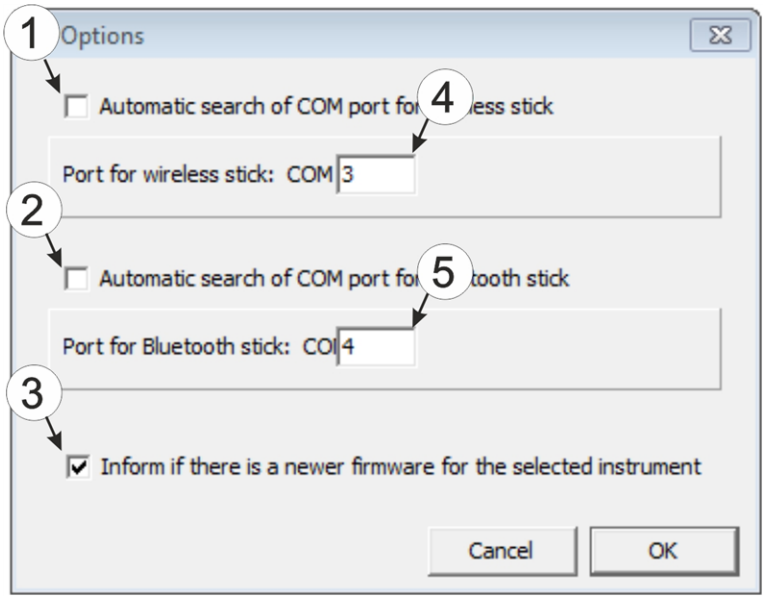


Menüpunkt "Settings"

10.5.1.1 Options

Über den Menüpunkt "Settings -> Options" lassen sich Einstellungen zu den COM-Ports an denen der USB-Funksender (206.657) bzw. der USB BLE-Adapter (300685) angeschlossen sind, festlegen sowie die automatische Suche nach verfügbaren Firmwareversionen aktivieren bzw. deaktivieren.

Der USB-Funksender (206.657) wird für myDatanet Geräte benötigt, die per Wireless M-Bus mit dem PC verbunden werden, der USB BLE-Adapter (300685) für jene, die per Bluetooth Low Energy mit dem PC verbunden werden. Informationen darüber, ob Ihr Gerät eine dieser Verbindungsmethoden unterstützt, finden Sie im Benutzerhandbuch des jeweiligen Geräts.

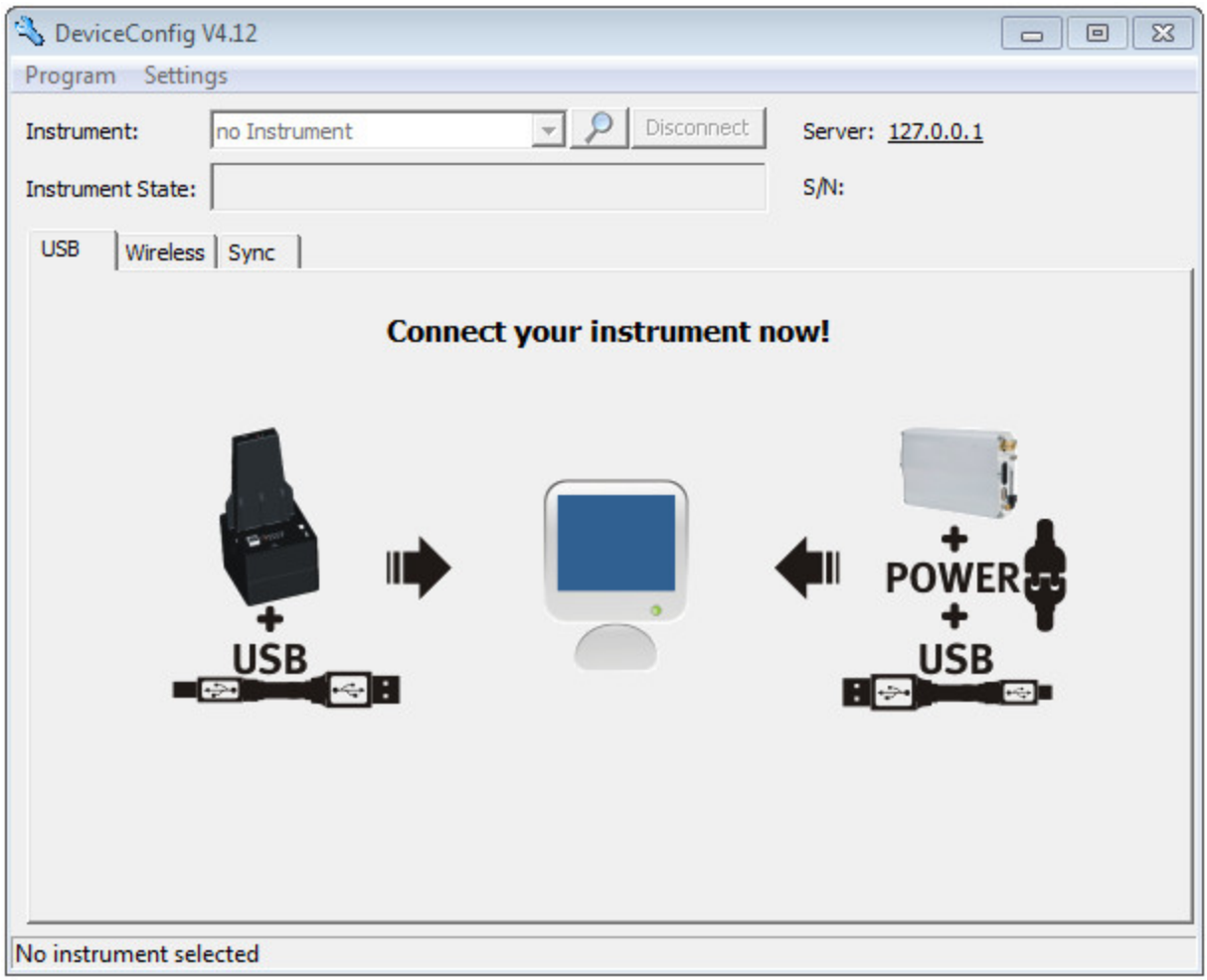


Menüpunkt "Settings -> Options"

1	aktiviert/deaktiviert die automatische Suche nach dem USB-Funksender (206.657) an allen verfügbaren COM-Ports	**4**	COM-Port, der mit dem USB-Funksender (206.657) verbunden ist (nur bei deaktivierter automatischer Suche sichtbar)
2	aktiviert/deaktiviert die automatische Suche nach dem USB BLE-Adapter (300685) an allen verfügbaren COM-Ports	**5**	COM-Port, der mit dem USB BLE-Adapter (300685) verbunden ist (nur bei deaktivierter automatischer Suche sichtbar)
3	aktiviert/deaktiviert die automatische Suche nach verfügbaren Firmwareversionen		

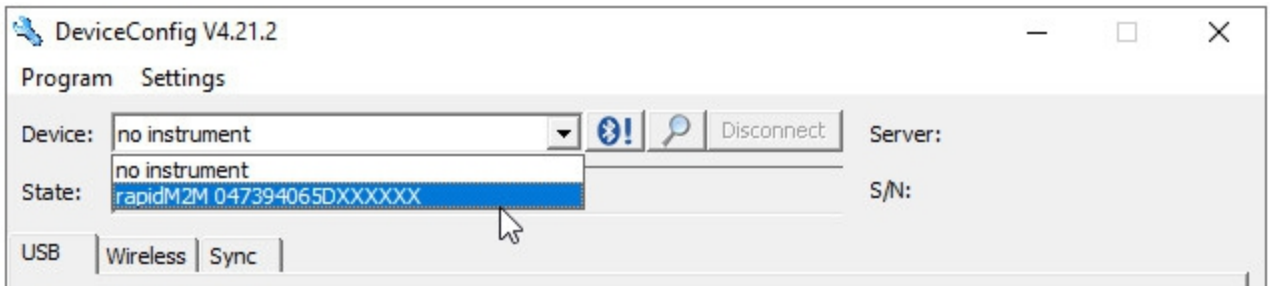
10.6 Verbindung zu einem Gerät mit USB Schnittstelle herstellen

1. Starten Sie das Konfigurationsprogramm DeviceConfig .



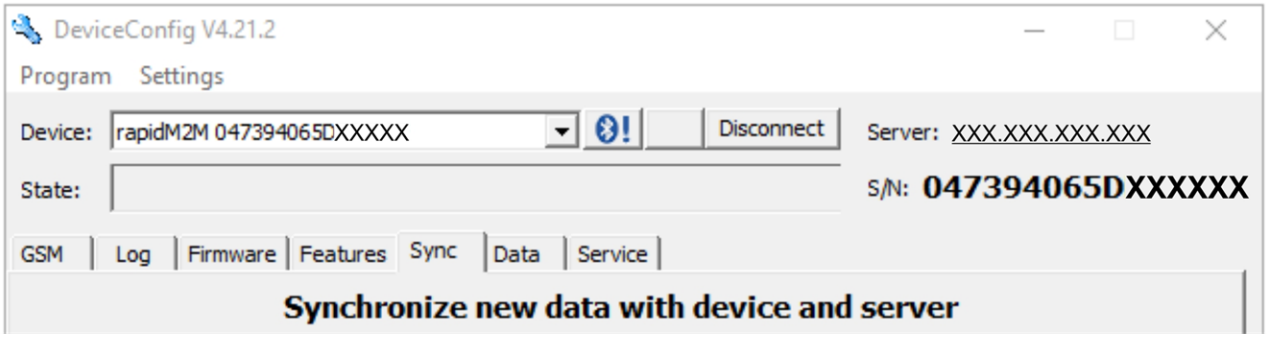
DeviceConfig

2. Verbinden Sie den myDatalogC33x mittels USB-Kabel mit dem PC.
3. Wählen Sie Ihr Gerät anhand der Seriennummer aus der Liste der gefundenen Geräte aus.



Liste der gefundenen Geräte

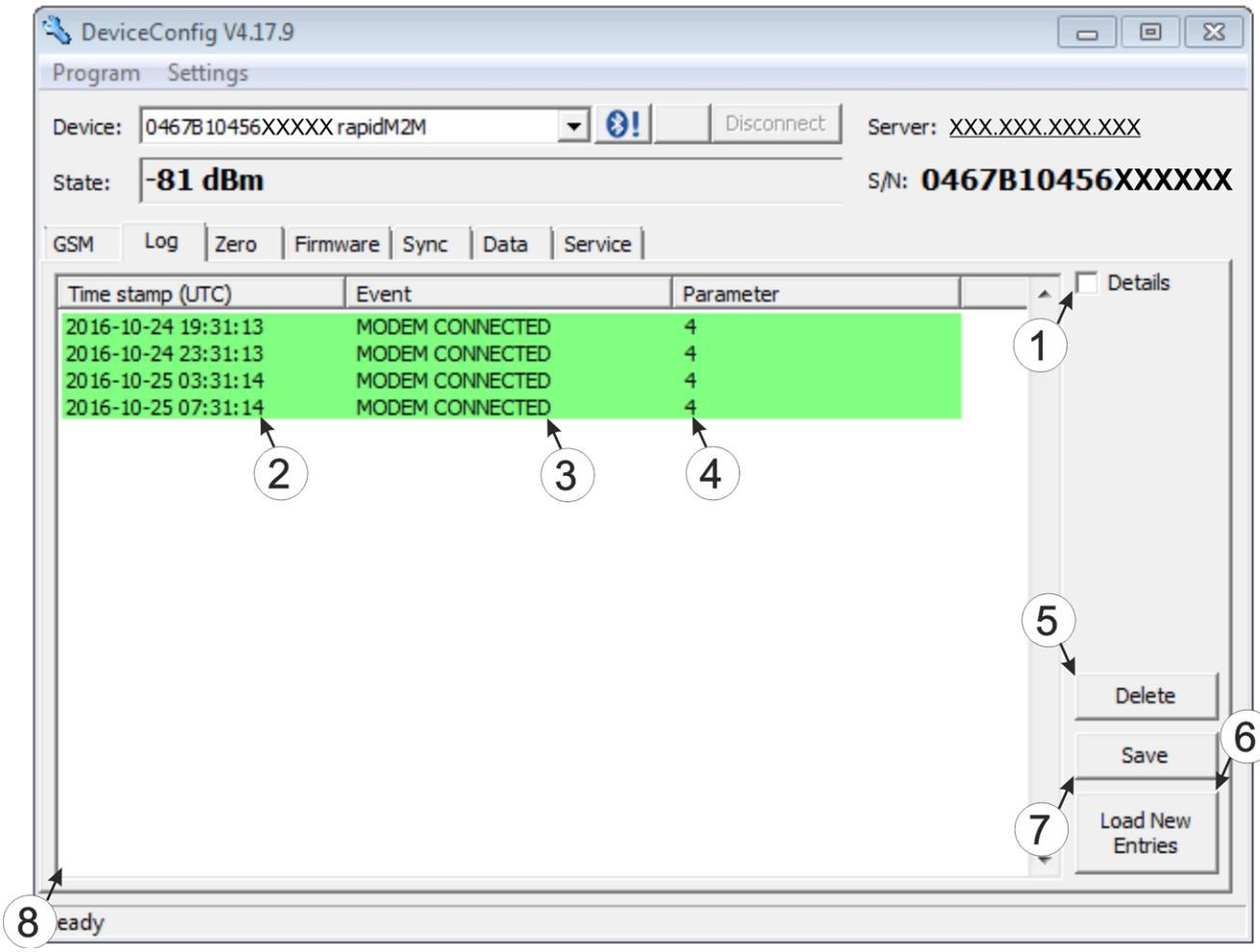
4. Warten Sie bis das DeviceConfig die Konfiguration des Geräts empfangen hat. Je nach Gerät werden daraufhin zusätzliche Karteireiter eingeblendet.



Karteireiter "Sync" bei aktiver Verbindung zu einem myDatalogC33x

10.7 Karteireiter "Log"

Dieser Karteireiter dient der Verwaltung der Log-Einträge. Er ermöglicht das Laden der Einträge vom myDatalogC33x , das Speichern als *.tsv-Datei und das Löschen der Einträge aus dem Speicher des myDatalogC33x .

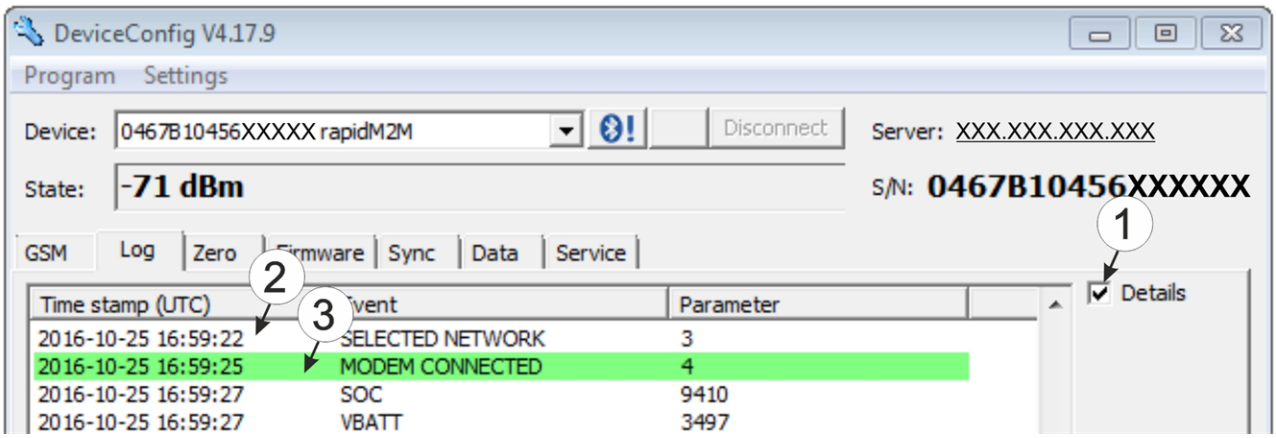


Karteireiter "Log"

1	Aktiviert die detaillierte Darstellung der Logeinträge	**5**	löscht die Log-Einträge aus dem Speicher des Geräts
2	Zeitstempel des Log-Eintrags	**6**	Lädt die Logeinträge vom Gerät
3	Log-Eintrag	**7**	Speichert die geladenen Logeinträge als tsv-File
4	Parameter des Log-Eintrags	**8**	Fenster für die Anzeige der geladenen Logeinträge

Die farbliche Markierung gibt Aufschluss darüber wie kritisch der Logeintrag zu bewerten ist. Die weiß gekennzeichneten informativen Logeinträge werden nur angezeigt, wenn die detaillierte Darstellung der Logeinträge aktiviert ist (siehe "Karteireiter "Log" mit aktivierter Detailansicht" auf Seite 90).

Farbe	Bewertung
weiß	Information über den aktuellen Betriebszustand
grün	
hellblau	
blau	
lila	
grau	
gelb	unkritischer Fehler
rot	kritischer Fehler



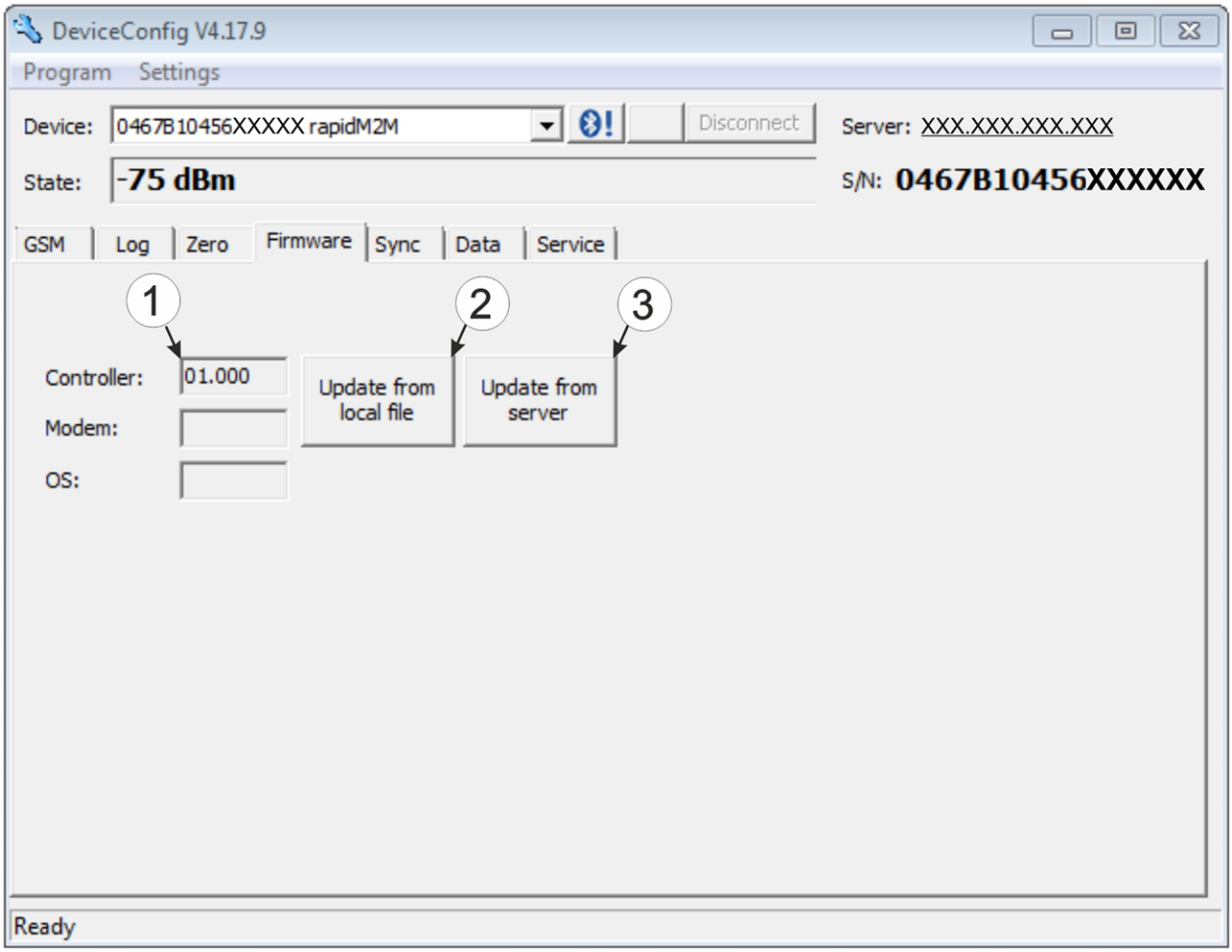
Karteireiter "Log" mit aktivierter Detailansicht

1 Aktiviert die detaillierte Darstellung der Logeinträge	**3** Log-Eintrag, der in jedem Fall angezeigt wird
2 informativer Log-Eintrag, der nur sichtbar ist, wenn die detaillierte Darstellung aktiviert ist	

10.8 Karteireiter "Firmware"

Dieser Karteireiter ermöglicht das direkte Einspielen der Firmware über die USB-Schnittstelle oder die Bluetooth Low Energy Schnittstelle. Es stehen 2 Varianten für das Updaten der Firmware zur Verfügung:

- Mittels zuvor heruntergeladenem Firmwarepaket
- Durch direktes Laden vom myDatanet-Server



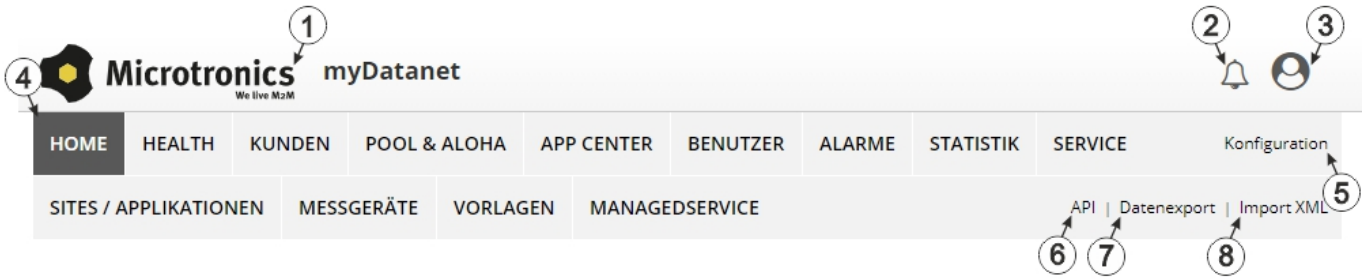
Karteireiter "Firmware"

1	aktuell installierte Softwareversion	**3**	Die Firmware wird direkt vom Server geladen und am Gerät installiert.
2	Button zum Einspielen eines zuvor heruntergeladenen Firmwarepaketes		

Kapitel 11 myDatanet-Server

Hinweis: *Alle Screenshots zeigen den myDatanet-Server in der Version 50v007 unter Verwendung des Standard-Farbschemas. Bei neueren Versionen können geringfügige Änderungen am Erscheinungsbild des Servers vorgenommen worden sein.*

11.1 Übersicht



Übersicht myDatanet-Server

1	frei wählbares Logo	**5**	öffnet die Maske zur Eingabe der globalen Einstellungen für den Server
2	öffnet das Fenster in dem die für den aktuell eingeloggten Benutzer bestimmten, vom System erstellten Benachrichtigungen zusammengefasst sind	**6**	öffnet den rapidM2M Playground
3	blendet das Menü zum Anpassen der Benutzereinstellungen und zum Ausloggen des aktuell aktiven Benutzers ein	**7**	wechselt in den Bereich "Datenexports" zur Konfiguration des Datenexports. Diese Schaltfläche ist nur sichtbar, wenn zumindest die Lizenz für eine Exportvariante vorhanden ist.
4	Schaltflächen zum Wechseln zwischen den einzelnen Serverbereichen	**8**	öffnet die Eingabemaske zum Upload einer XML-Datei. Diese Schaltfläche ist nur sichtbar, wenn die Lizenz für den XML-Import vorhanden ist.

11.1.1 Erklärung der Symbole


Fügt zur aktuellen Liste (Auswertungen, Sites, Benutzer, ...) einen neuen Eintrag hinzu


Löscht das nebenstehende Element (Auswertung, Site, Benutzer, ...) aus der Liste


Ruft die Eingabemaske zum Editieren des nebenstehenden Elements (Auswertung, Site, Benutzer, ...) auf

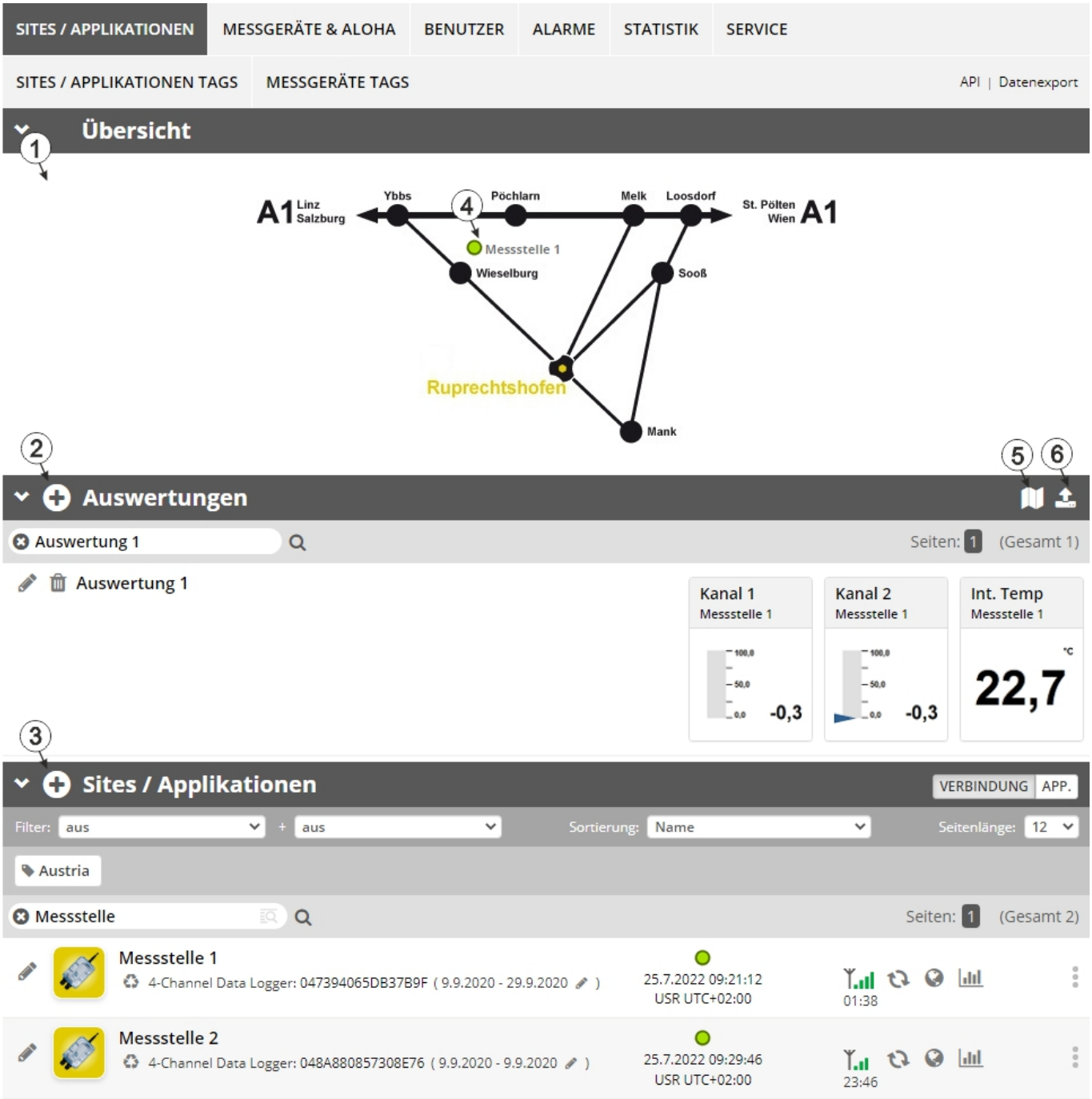
11.2 Bereich "Kunden"

Übersicht des Bereichs "Kunden"

1	Bereich, in dem eine Bilddatei als "Karte" und/oder die OpenStreetMaps Karte eingeblendet werden kann Auf der als "Karte" verwendeten Bilddatei lassen sich die Sites manuell platzieren. In der OpenStreetMaps Karte werden die Sites erst angezeigt, wenn der Site GPS-Koordinaten zugewiesen wurden.
2	fügt einen neuen Kunden hinzu

3	Liste der Tags, die mindestens einem der in der Kundenliste angezeigten Kunden zugewiesen sind. Wurde die Kundenliste mittels Suchfeld oder Auswahl eines Tags beschränkt, wird dies bei der Erstellung der Liste der Tags berücksichtigt. Sobald die Kundenliste durch Auswahl eines Tags eingeschränkt wurde, erscheint am Ende der Liste der Tags ein Kreuz. Durch Klicken auf dieses Kreuz wird die Auswahl aller Tags zurückgesetzt und die Einschränkung aufgehoben. Durch Klicken mit der linken Maustaste auf einen der Tags werden in der Kundenliste nur mehr jene Kunden angezeigt, denen der entsprechende Tag zugewiesen ist und der gewählte Tag ist farblich hinterlegt. Durch Klicken mit der rechten Maustaste auf einen der Tags werden alle Kunden, denen der entsprechende Tag zugewiesen ist, ausgeblendet, der gewählte Tag ist farblich hinterlegt und die Bezeichnung des Tags durchgestrichen. Erneutes Klicken mit derselben Maustaste hebt die Einschränkung wieder auf.
4	öffnet die Eingabemaske zur Konfiguration des Kunden
5	löscht den Kunden
6	Kommentar, der in der Konfiguration des Kunden eingegeben werden kann
7	Wurde eine Standardauswertung definiert, gelangen Sie durch Klicken auf den Namen des Kunden zur Standardauswertung. Andernfalls wird durch Klicken auf den Namen des Kunden der Bereich "Sites / Applikationen" auf Kundenebene geöffnet (siehe "Bereich "Sites / Applikationen" auf Kundenebene" auf Seite 96 bzw. "Auswertungen" auf Seite 97).
8	Suchfeld zum Filtern der Kundenliste
9	Adresse des Kunden, die über die Eingabemaske zur Konfiguration des Kunden eingegeben werden kann
10	Symbol, über das sich eine OpenStreetMaps Karte laden lässt, auf der die Sites dargestellt werden. (siehe "Kartendarstellung" auf Seite 97)
11	Symbol, über das sich eine Bilddatei als "Übersichts-Karte" auf den Server laden lässt Um die "Karte" wieder zu entfernen, öffnen Sie den Upload-Dialog erneut und klicken Sie auf "senden" ohne zuvor eine Bilddatei auszuwählen.

11.3 Bereich "Sites / Applikationen" auf Kundenebene



Übersicht des Bereichs "Sites / Applikationen" auf Kundenebene

1	Bereich, in dem eine Bilddatei als "Karte" und/oder die OpenStreetMaps Karte eingeblendet werden kann Auf der als "Karte" verwendeten Bilddatei lassen sich die Sites manuell platzieren. In der OpenStreetMaps Karte werden die Sites erst angezeigt, wenn der Site GPS-Koordinaten zugewiesen wurden.

2	Liste der Auswertungen (siehe "Auswertungen" auf Seite 97)
3	Liste der Sites / Applikationen (siehe "Site" auf Seite 74)
4	Symbol, das eine Site auf der "Karte" repräsentiert
5	Symbol, über das sich eine OpenStreetMaps Karte laden lässt, auf der die Sites dargestellt werden. (siehe "Kartendarstellung" auf Seite 97)
6	Symbol, über das sich eine Bilddatei als "Karte" auf den Server laden lässt Um die "Karte" wieder zu entfernen, öffnen Sie den Upload-Dialog erneut und klicken Sie auf "senden" ohne zuvor eine Bilddatei auszuwählen.

11.3.1 Auswertungen

Die Auswertungen bieten eine Vielzahl an Möglichkeiten zur grafischen Darstellung der Daten auf der Web-Oberfläche des myDatanet-Server bzw. dem Download der Daten vom myDatanet-Servers. Eine detailliertere Anleitung zum Erstellen und dem Umgang mit den Auswertungen finden Sie im Benutzerhandbuch für myDatanet-Server (206.886).

11.3.2 Kartendarstellung

Die Kartendarstellung dient dazu, einen Überblick über die geografische Position der Messstellen zu geben. Eine detailliertere Anleitung zur Bedienung und Konfiguration der Kartendarstellung finden Sie im Benutzerhandbuch für myDatanet-Server (206.886).

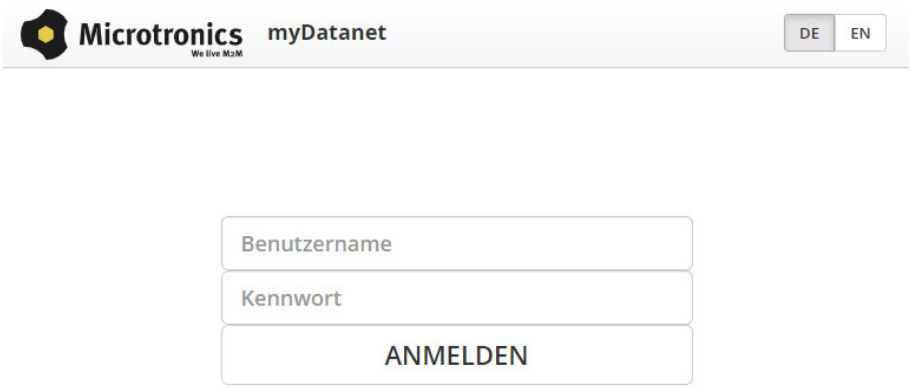
11.4 Empfohlene Vorgehensweise

11.4.1 Anlegen der Site

> ***Hinweis:*** *Abhängig vom jeweiligen Benutzerlevel sind einige der in den folgenden Kapiteln erwähnten Felder unter Umständen ausgeblendet. Wenden Sie sich in diesem Fall an den Administrator des myDatanet-Servers.*

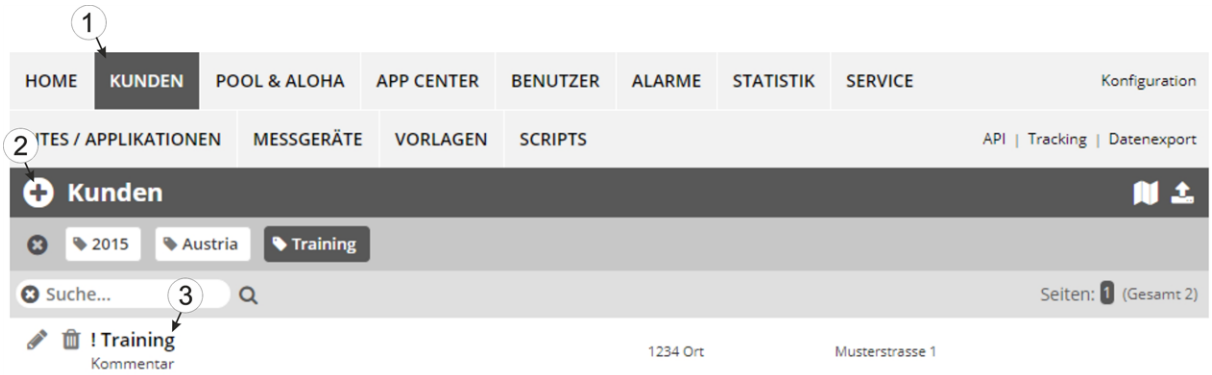
Eine detailliertere Anleitung zum Anlegen einer neuen Site finden Sie im Benutzerhandbuch für myDatanet-Server (206.886).

1. Loggen Sie sich über das Web-Interface am myDatanet–Server ein. Die Web-Adresse erhalten Sie von Ihrem zuständigen Vertriebspartner.



Login Formular des myDatanet–Servers

2. Klicken Sie auf den Menüpunkt "Kunde" des myDatanet–Servers, um die Liste der verfügbaren Kunden aufzurufen. Wählen Sie einen bestehenden Kunden aus oder legen Sie einen neuen Kunden an.

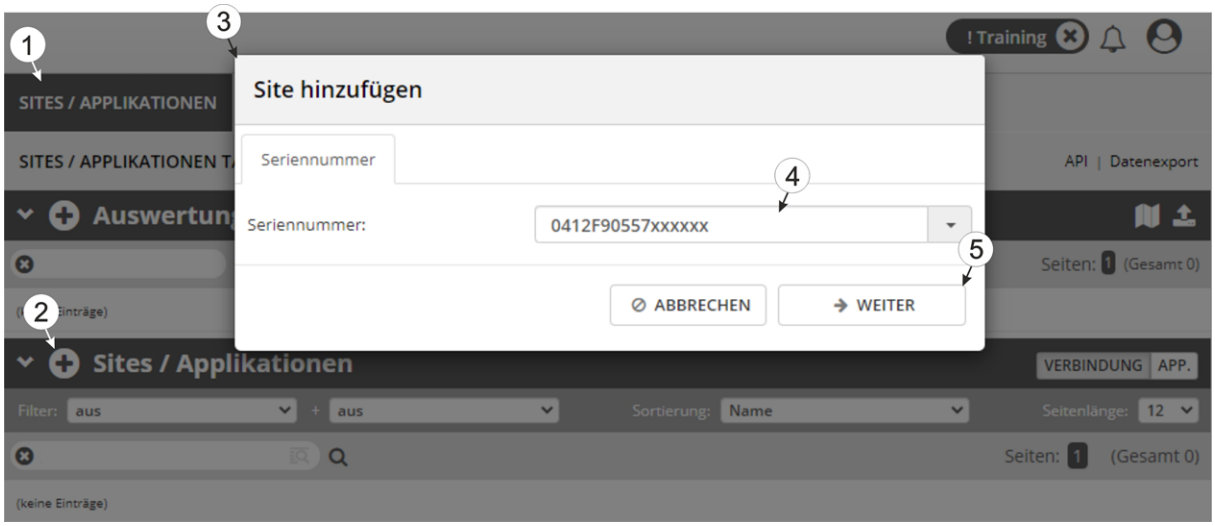


Auswählen des Kunden

1 Menüpunkt zum Aufrufen der Kundenliste	**3** Liste der verfügbaren Kunden
2 Anlegen eines neuen Kunden	

3. Klicken Sie auf den Menüpunkt "Sites / Applikationen" des myDatanet–Servers, um die Liste der bestehenden Sites / Applikationen aufzurufen. Öffnen Sie das Eingabefenster zum Anlegen einer neuen Site durch Klicken auf das Symbol "Neue Site / Applikation hinzufügen", geben Sie die Seriennummer Ihres Geräts in das entsprechende Feld ein und klicken Sie anschließend auf den "Weiter" Button.

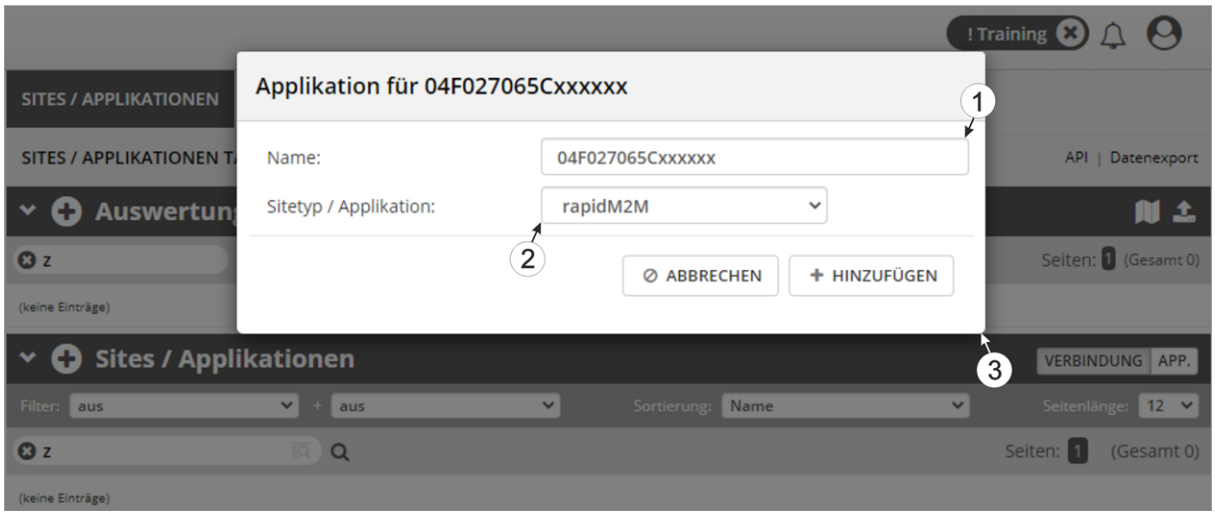
> ***Hinweis:*** *Die Seriennummer finden Sie auf dem Typenschild des Geräts (siehe "Gerätekennzeichnung" auf Seite 25)*



Anlegen der Site

1 Menüpunkt zum Aufrufen der Liste der bestehenden Sites / Applikationen	**4** Feld zur Eingabe der Seriennummer
2 Symbol "Neue Site / Applikation hinzufügen"	**5** Button "Weiter"
3 Eingabefenster für das Anlegen einer neuen Site	

4. Ändern Sie, falls erforderlich, den vorgeschlagenen Namen der Site, wählen Sie den gewünschten Sitetyp bzw. die gewünschte Applikation aus der Dropdown-Liste aus und klicken Sie anschließend auf den "Hinzufügen" Button.



Anlegen der Site abschließen

1 Name der Site (frei wählbar)	**3** Button "Hinzufügen"
2 Dropdown-Liste der verfügbaren Applikationen, Vorlagen und Site Typen	

Kapitel 12 rapidM2M Studio

Hinweis: *Die webbasierte Entwicklungsumgebung rapidM2M Studio wird kontinuierlich weiterentwickelt wodurch es auch zu geringfügigen Änderungen am Erscheinungsbild des Programms im Vergleich zu den in dieser Anleitung verwendeten Screenshots kommen kann.*

12.1 Allgemein

Der Zugang zur webbasierten Entwicklungsumgebung rapidM2M Studio ist im Microtronics Partner Programm, für das Sie sich unter folgender Adresse kostenlos anmelden können, enthalten:

https://partner.microtronics.com

Es handelt sich um eine webbasierte IDE, die den Kunden bei der Erstellung von IoT Applikationen für das myDatalogC33x unterstützen soll. Dies umfasst den kompletten Entwicklungsprozess - angefangen vom Editieren des Source-Codes, über das Testen im Zuge des Entstehungsprozesses bis hin zum Veröffentlichen der fertigen IoT Applikation im rapidM2M Store . Alle Elemente aus denen eine IoT Applikation besteht, sind dabei in einem einzigen Projekt zusammengefasst. Bei diesen handelt es sich um:

- **Device Logic:** lokale am myDatalogC33x installierte Intelligenz
- **Backend Logic:** am myDatanet-Server installierte Intelligenz
- **Data Descriptor:** beschreibt die Struktur der Daten (Messdaten, Konfigurationen, usw.), die zwischen myDatalogC33x , myDatanet-Server und externen Systemen (z.B. per REST-API angebundene Frontends) ausgetauscht werden.
- **Portal View:** Einfaches Frontend, das vom myDatanet-Server ausgeliefert wird (z.B. für schnelle Prototypenentwicklung und/oder Bereitstellen von administrativen Daten)

Neben dem Dashboard (siehe "Projekt Dashboard " auf Seite 103) zur Verwaltung der Projekte beseht das rapidM2M Studio aus zwei Hauptoberflächen:

- **CODEbed:** Bearbeiten und Kompilieren der Source Codes (siehe "CODEbed" auf Seite 104)
- **TESTbed:** Testen der IoT Applikation in Verbindung mit einem lokal verbundenen Gerät und dem zugehörigen Back-End, d.h. dem myDatanet-Server (siehe "TESTbed" auf Seite 105)

Die im rapidM2M Studio implementierte Sharing-Funktion ermöglicht es Entwicklern aus unterschiedlichen Disziplinen (Firmwareprogrammierer, Cloud-Developer, Web-Designer, usw.) gemeinsam eine IoT Applikation zu erstellen sowie Projekte und Bibliothek mit Kollegen und der Community zu teilen. Das integrierte Versionsmanagement sorgt zudem für eine geregelte Verteilung von Updates einer IoT Applikation über die komplette Kette vom rapidM2M Studio über den rapidM2M Store zu den Sites (die auf Basis der IoT Applikation erstellt wurden) bis hin zum myDatalogC33x .

12.2 Voraussetzungen

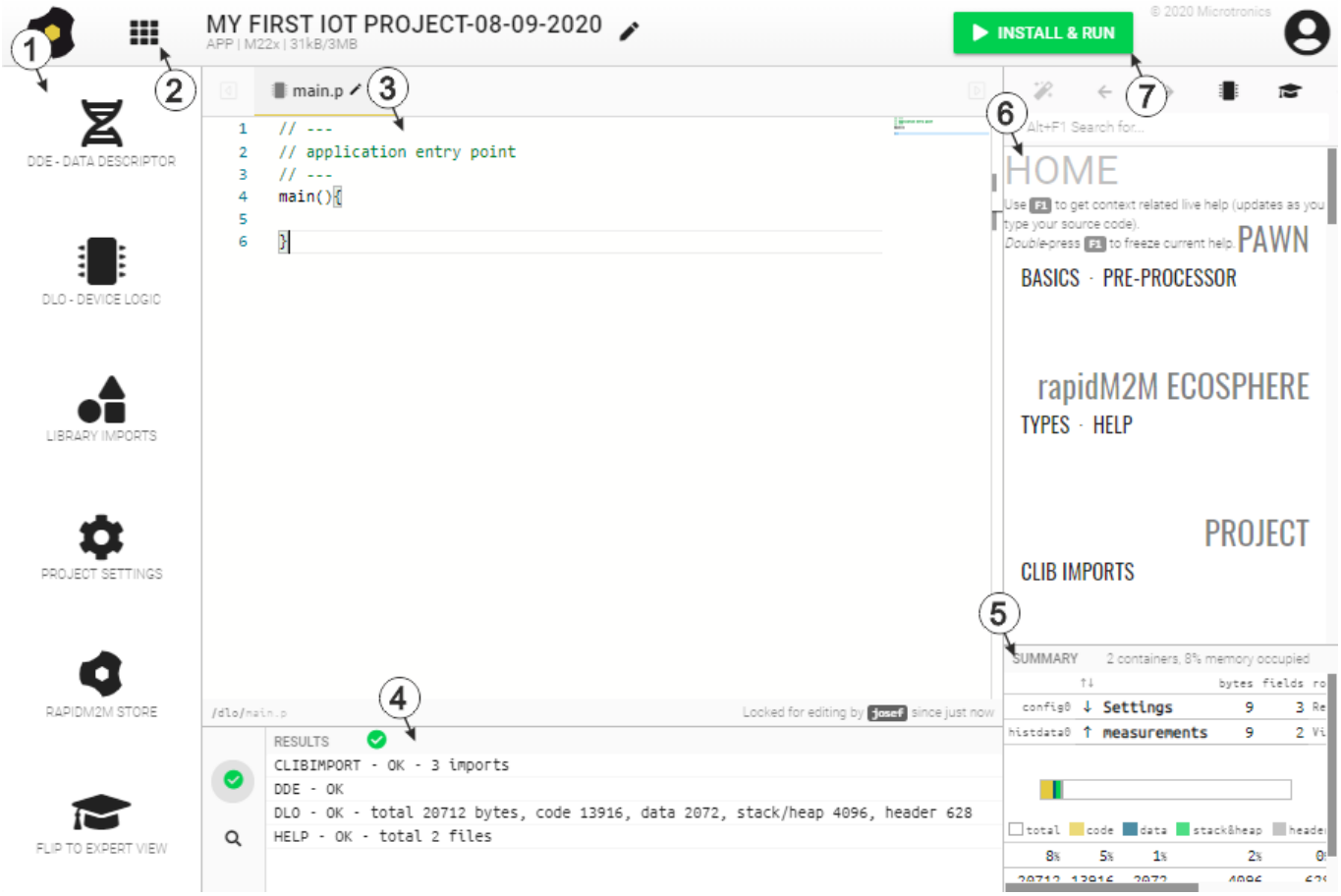
Schnittstellen	1 x USB
Betriebssystem	Windows 7 Windows 10 (empfohlen) MacOS 10.12 oder höher Linux (Fedora 32, Ubuntu 20.04, Archlinux 2020.06.01)
Internetverbindung	erforderlich
Benötigter Speicherplatz	keine Installation erforderlich
Browser	nur Google Chrome

12.3 Projekt Dashboard

Projekt Dashboard des rapidM2M Studio

1	Suchfeld zum Filtern der Liste der Projekte	
2	Button zum Umschalten der Sortierung der Projektliste nach Alphabet oder letzer Verwendung	
3	öffnet die Kurzanleitung für das rapidM2M Studio	
4	Button zum Einblenden des Menüs, das alle für den aktuell aktiven Benutzer relevanten Einstellungen erhält	
5	Button zum Erstellen eines neuen Projektes	
6	Buttons zum Filtern der Liste der Projekte nach:	
	kürzlich verwendet	
	alle meine Projekte	
	von mir geteilte Projekte	
	mit mir geteilte Projekte	
7	Kachel, die alle wichtigen Informationen zu einem IoT Projekt enthält	
8	Liste der "Collections"	
	alle Projekte, die nicht einer anderen "Collection" zugewiesen sind	
	favorisierte Projekte	
	von Microtronics bereitgestellte Beispiel-Bibliotheken	
	von Microtronics bereitgestellte Beispiele	
	vom Benutzer selbst erstellte "Collection"	
9	Button zum Erstellen einer neuen "Collection"	

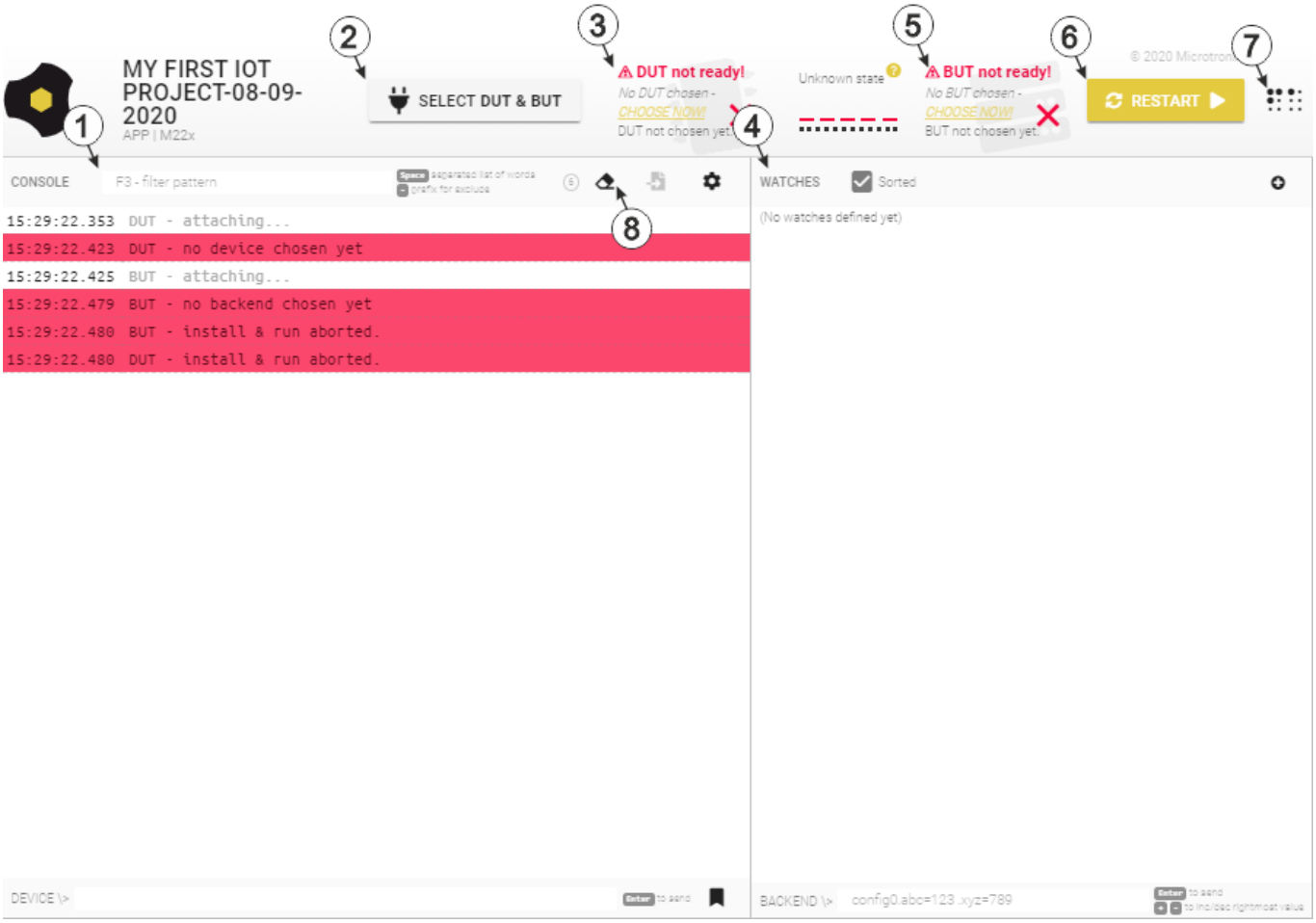
12.4 CODEbed



CODEbed des rapidM2M Studio

1	Navigationspanel
2	Zurück zum Projekt Dashboard
3	Editor-Panel
4	Compiler Results inkl. Warnungen und Fehler
5	Speichernutzung
6	Kontextsensitive Hilfe
7	Installiert die aktuellen Binaries des Projektes am Gerät und Backend (d.h. dem myDatanet-Server) und öffnet das TESTbed

12.5 TESTbed



TESTbed des rapidM2M Studio

1	Debug Konsole
2	Öffnet zuerst das Fenster zum Auswählen und Verbinden des "Device under test" und dann das Fenster für die Eingabe der Zugangsdaten für das "Backend under test"
3	Informationen zum "Device under test"
4	Watches Panel
5	Informationen zum "Backend under test" (d.h. dem myDatanet-Server)
6	Startet die auf dem Gerät installierte Device Logic neu. Dazu wird die Device Logic erneut ins Gerät geladen. Im CODEbed vorgenommene Änderungen werden dabei allerdings nicht berücksichtigt. Dies erfolgt erst wieder beim Klicken des Buttons "Install & Run" im CODEbed.
7	Button zum Ausblenden/Anzeigen zusätzlicher Panels
8	Button zum Löschen des Konsolen Outputs

Kapitel 13 Device Logic

13.1 Allgemein

Das folgende Kapitel beschreibt die Funktionalität der Device Logic. Bei der verwendeten Programmiersprache handelt es sich um eine C-ähnliche Scriptsprache built on PAWN, welche auf embedded Systemen läuft.

Zusätzliche detaillierte Informationen finden Sie auf der Website der Entwickler: http://www.compuphase.com/pawn/pawn.htm.

Es gibt mehrere Möglichkeiten, um ein Device Logic für das myDatalogC33x zu erstellen:

- direkte Eingabe in das Eingabefenster „Device Logic“ im Konfigurationsabschnitt „Steuerung“
- Hochladen eines zuvor erstellten Binary-Files (*.amx) auf den myDatanet–Server
- Verwendung des CODEbed (siehe "CODEbed" auf Seite 104) der webbasierten Entwicklungsumgebung rapidM2M Studio

13.1.1 Direkte Eingabe einer Device Logic

Die Eingabe der Device Logic erfolgt über den Konfigurationsabschnitt „Steuerung“ (siehe "Steuerung" auf Seite 75) der Eingabemaske zur Konfiguration der Messstelle. Als „Device Logic Type“ muss „Pawn“ ausgewählt werden, damit das myDatalogC33x die unter „Device Logic“ eingegebenen Befehle als Script interpretiert.

13.1.2 Hochladen eines Binary-Files

Wurde über die Listenauswahl "Device Logic Quelle" im Konfigurationsabschnitt „Steuerung“ (siehe "Steuerung" auf Seite 75) der Eingabemaske zur Konfiguration der Messstelle der Eintrag "Hochladen einer kompilierten Device Logic" ausgewählt, kann ein zuvor mittels z.B. der webbasierten Entwicklungsumgebung rapidM2M Studio (siehe "rapidM2M Studio " auf Seite 101) erstelltes Binary-File auf den myDatanet–Server hochgeladen werden. Dieses wird dann bei der nächsten Verbindung in das myDatalogC33x geladen. Als „Device Logic Type“ muss auch bei dieser Methode „Pawn“ ausgewählt werden, damit das myDatalogC33x die Befehle als Script interpretiert.

13.1.3 Verwenden des CODEbed der webbasierten Entwicklungsumgebung rapidM2M Studio

Beim CODEbed handelt es sich um eine der beiden Hauptoberflächen der webbasierten Entwicklungsumgebung rapidM2M Studio . Das CODEbed dient zum Erstellen und Kompilieren der Source Codes für alle Elemente (Device Logic, Backend Logic, Data Descriptor und Portal View) einer IoT Applikation. Zum Funktionsumfang des rapidM2M Studio gehört auch das Übertragen der kompilierten Device Logic per USB-Verbindung in das myDatalogC33x und das Kopieren des Data Descriptors zur Development Site die mit dem myDatalogC33x verknüpft ist.

13.2 Compiler-Optionen

Komprimierung des Pawn Programmcodes

```
// Über den Parameter wird angegeben welche der Sektionen komprimiert werden
// sollen
// 0: keine Komprimierung (default)
// 1: DATA
// 2: DATA und CODE
// 3: DATA, CODE und TABELS

#pragma amxcompress <0-3>
```

13.3 Device API

13.3.1 Konstanten

Returncodes für allgemeine Zwecke

```
OK                        = 0,
ERROR                     = -1,
ERROR_PARAM               = -2,  // Parameter error
ERROR_UNKNOWN_HDL         = -3,  // Unknown handler, handle or resource error
ERROR_ALREADY_SUBSCRIBED  = -4,  // Already subscribed service or resource error
ERROR_NOT_SUBSCRIBED      = -5,  // Not subscribed service error
ERROR_FATAL               = -6,  // Fatal error
ERROR_BAD_HDL             = -7,  // Bad handle or resource error
ERROR_BAD_STATE           = -8,  // Bad state error
ERROR_PIN_KO              = -9,  // Bad PIN state error
ERROR_NO_MORE_HANDLES     = -10, /* The service subscription maximum capacity is
                                    reached */
ERROR_DONE                = -11, /* The required iterative process is now
                                    terminated */
ERROR_OVERFLOW            = -12, /* The required operation has exceeded the
                                    function capabilities */
ERROR_NOT_SUPPORTED       = -13, /* An option, required by the function, is not
                                    enabled on the CPU, the function is not
                                    supported in this configuration */
ERROR_NO_MORE_TIMERS      = -14, /* The function requires a timer subscription,
                                    but no more timer resources are available */
ERROR_NO_MORE_SEMAPHORES  = -15, /* The function requires a semaphore allocation,
                                    but there are no more semaphore resources */
ERROR_SERVICE_LOCKED      = -16, /* The function was called from a low or high
                                    level interrupt handler (the function is
                                    forbidden in this case) */
ERROR_MEM                 = -100,// error allocating memory
ERROR_SIM_STATE           = -101,// SIM state error
ERROR_MODEM_DISABLED      = -102,// Modem disabled
ERROR_SENSOR_DISABLED     = -102,/* Sensor disabled
                                    (Alias for ERROR_MODEM_DISABLED) */
ERROR_FEATURE_LOCKED      = -103,// feature locked
ERROR_TXITF               = -104,/* tx interface (uplink) not available
                                    (e.g. not opened, currenly closing) */
```

13.3.2 Timer, Datum & Zeit

13.3.2.1 Arrays mit symbolischen Indizes

TrM2M_DateTime

detaillierte Aufschlüsselung von Datum und Zeit

```
// year         Jahr relativ zum Jahr 2000, d.h. 14 für das Jahr 2014
// month        Monat     (1..12)
// day          Tag       (1..31)
// hour         Stunden   (0..23)
// minute       Minuten   (0..59)
// second       Sekunden  (0..59)
// DoW          Wochentag (0 = Montag ... 6 = Sonntag)
// timestamp    Zeitstempel (Sekunden seit 31.12.1999)
// timestamp256 Bruchteil der nächsten begonnenen sec. (Auflösung 1/256 sec.)

#define TrM2M_DateTime[ .year, .month, .day, .hour, .minute, .second, .DoW,
                        .timestamp, .timestamp256                          ]
```

13.3.2.2 Konstanten

Zeitbasis-Flags

Steuerflags für die Funktion rM2M_SetDateTime()

```
RM2M_DATETIME_LOCALTIME = 0b00000001, // die Übergabe erfolgt in Local Time
```

13.3.2.3 Funktionen

native rM2M_GetTime(&hour=0, &minute=0, &second=0, timestamp=0);

Wurde kein Timestamp übergeben (timestamp=0), wird die aktuelle Systemzeit (in UTC) in Stunden / Minuten / Sekunden konvertiert. Andernfalls wird der übergebene Timestamp in Stunden / Minuten / Sekunden konvertiert.

Parameter	***Erklärung***
hour	*Variable zur Aufnahme der Stunden - OPTIONAL*
minute	*Variable zur Aufnahme der Minuten - OPTIONAL*
second	*Variable zur Aufnahme der Sekunden - OPTIONAL*
timestamp	*Zeitstempel, der konvertiert werden soll* *= 0: Es wird die aktuelle Systemzeit (in UTC) konvertiert.* *> 0: Es wird der übergebene Zeitstempel konvertiert.* *(Der Zeitstempel muss in Sekunden seit 31.12.1999 angegeben werden.)*

	Erklärung
Rückgabewert	• *timestamp = 0: Sekunden seit 31.12.1999 (aktuelle Systemzeit in UTC)* • *timestamp > 0: Der übergebene Zeitstempel wird zurückgegeben.*

native rM2M_GetDate(&year=0, &month=0, &day=0, timestamp=0);

Wurde kein Timestamp übergeben (timestamp=0), wird für die aktuelle Systemzeit (in UTC) das Datum (Jahr, Monat, Tag) ermittelt. Andernfalls wird für den übergebenen Timestamp das Datum (Jahr, Monat, Tag) ermittelt.

Parameter	***Erklärung***
year	*Variable zur Aufnahme des Jahres - OPTIONAL* ***Hinweis:*** *Die Angabe des Jahres erfolgt relativ zum Jahr 2000, d.h. für das Jahr 2014 wird der Wert 14 retourniert.*
month	*Variable zur Aufnahme des Monats - OPTIONAL*
day	*Variable zur Aufnahme des Tags - OPTIONAL*
timestamp	*Zeitstempel für den das Datum ermittelt werden soll* *= 0: Es wird das Datum für die aktuelle Systemzeit (in UTC) ermittelt.* *> 0: Es wird das Datum für den übergebene Zeitstempel ermittelt.* *(Der Zeitstempel muss in Sekunden seit 31.12.1999 angegeben werden.)*

	Erklärung
Rückgabewert	• *timestamp = 0: Sekunden seit 31.12.1999 (aktuelle Systemzeit in UTC)* • *timestamp > 0: Der übergebene Zeitstempel wird zurückgegeben.*

native rM2M_GetDateTime(datetime[TrM2M_DateTime]);

liest die aktuelle Zeit (in UTC) und das Datum vom System

Parameter	***Erklärung***
datetime	*Struktur zur Aufnahme einer detaillierten Aufschlüsselung von Datum und Zeit (siehe "TrM2M_DateTime" im Kapitel "Arrays mit symbolischen Indizes" auf Seite 109)*

	Erklärung
Rückgabewert	• *OK, wenn erfolgreich* • *ERROR, wenn ein ungültiger Parameter übergeben wurde*

native rM2M_SetDateTime(datetime[TrM2M_DateTime], flags=0);

setzt Datum und Zeit des Systems auf die in der übergebenen Struktur enthaltenen Werte

Parameter	***Erklärung***
datetime	*Struktur, die eine detaillierte Aufschlüsselung von Datum und Zeit enthält (siehe "TrM2M_DateTime" im Kapitel "Arrays mit symbolischen Indizes" auf Seite 109).* *.timestamp = 0:* *Die in .year, .month, .day, .hour, .minute und .second enthalten Werte werden für das Setzen von Datum/Zeit herangezogen.* *.timestamp != 0:* *Der in .timestamp enthaltene Zeitstempel wird für das Setzen von Datum/Zeit herangezogen.*
flags	*Konfigurationsflags für das Setzen der Systemzeit - OPTIONAL* *Bit0 (RM2M_DATETIME_LOCALTIME):* *ist zu setzen, wenn die übergebene Struktur die Zeit in Local Time enthält*

	Erklärung
Rückgabewert	• *> 0, Differenz in sec. zw. bisheriger und zu setzender Zeit* • *0, wenn die Differenz zw. bisheriger und zu setzender Zeit kleiner als 5sec. ist* • *ERROR, wenn ungültige Parameter übergeben wurden* • *ERROR-1, wenn der zu setzende Zeitpunkt relativ zur bisherigen Systemzeit mehr als einen Tag in der Zukunft liegt*

native rM2M_GetTimezoneOffset();

liefert die Differenz (in Sekunden) zwischen Systemzeit (UTC) und der für die Messstelle am myDatanet-Server konfigurierten lokalen Zeit. Dadurch kann im Skript die lokale Zeit bestimmt werden, indem diese Differenz zur Systemzeit (UTC) addiert wird. Der Offsetwert wird vom myDatanet-Server entsprechend der eingestellten Zeitzone (inkl. Sommer-/Winterzeit) gebildet und bei jeder Verbindung mit dem Gerät synchronisiert.

Bsp.: Für die Messstelle wird die mitteleuropäische Zeit (MEZ = UTC+1) verwendet -> Offset = 3600sec.

	Erklärung
Rückgabewert	*Offsetwert in Sekunden*

native rM2M_DoW(timestamp);

berechnet den Wochentag aus einem gegebenen Timestamp

Parameter	*Erklärung*
timestamp	*Zeitstempel des zu berechnenden Tages*

	Erklärung
Rückgabewert	*Wochentag, 0=Montag ... 6=Sonntag*

native rM2M_TimerAdd(funcidx);

erzeugt einen neuen 1s Timer

Parameter	*Erklärung*
funcidx	*Index der öffentlichen Funktion, die nach Ablauf des Timers aufgerufen werden soll* *Typ der Funktion: public func();*

	Erklärung

Rückgabewert	• *OK, wenn erfolgreich*
	• *ERROR, wenn einer der folgenden Fehler auftritt:*
	  • *Es wurde kein gültiger Index übergeben*
	  • *Es können keine weiteren Timer mehr angelegt werden (maximale Anzahl erreicht)*
	  • *Es kommt zu einem internen Fehler*
	• *< OK, wenn ein anderer Fehler auftritt (siehe "Returncodes für allgemeine Zwecke" im Kapitel "Konstanten" auf Seite 108)*

native rM2M_TimerRemove(funcidx);

entfernt einen 1s Timer

Parameter	*Erklärung*
funcidx	*Index der öffentlichen Funktion des zu entfernenden Timers* *Typ der Funktion: public func();*

	Erklärung
Rückgabewert	• *OK, wenn erfolgreich* • *ERROR, wenn kein gültiger Index übergeben wurde oder bei einem internen Fehler* • *< OK, wenn ein anderer Fehler auftritt (siehe "Returncodes für allgemeine Zwecke" im Kapitel "Konstanten" auf Seite 108)*

native rM2M_TimerAddExt(funcidx, bool:cyclic, time);

erzeugt einen neuen ms Timer

Wichtiger Hinweis: *Die maximale Anzahl gleichzeitig aktiver ms Timer ist 8.*

Parameter	*Erklärung*
funcidx	*Index der öffentlichen Funktion, die nach Ablauf des Timers aufgerufen werden soll* *Typ der Funktion: public func();*
cyclic	*Einstellung für das Verhalten nach Ablauf des Timerintervalls:* *true: Der Timer soll nach dem Ablauf des Intervalls neu gestartet werden.* *false: Der Timer wird nach Ablauf des Intervalls gestoppt.*
time	*Timerintervall in Millisekunden (max. 60.000ms)* ***Hinweis:*** *Durch Setzen des Intervalls auf 0ms kann ein Timer erzeugt werden dessen Callback-Funktion unmittelbar nachdem der aktuelle Codeblock (z.B. main-Funktion) ausgeführt wurde aufgerufen wird. Allerdings dürfen nur Single-Shot-Timer (d.h. der Timer wird nach Ablauf des Intervalls gestoppt) mit einem Intervall von 0ms initialisiert werden.*

	Erklärung

Rückgabewert	• *OK, wenn erfolgreich*
	• *ERROR, wenn einer der folgenden Fehler auftritt*
	  • *Es wurde kein gültiger Index übergeben.*
	  • *Es wurde ein Intervall von 0ms angegeben und der Timer soll nach Ablauf des Timeouts automatisch neu gestartet werden (d.h. zyklischer 0ms-Timer).*
	  • *Internen Fehler*
	  • *Es können keine weiteren Timer mehr angelegt werden (maximale Anzahl erreicht).*
	• *< OK, wenn ein anderer Fehler auftritt (siehe "Returncodes für allgemeine Zwecke" im Kapitel "Konstanten" auf Seite 108)*

native rM2M_TimerRemoveExt(funcidx);

entfernt einen ms Timer

Parameter	*Erklärung*
funcidx	*Index der öffentlichen Funktion des zu entfernenden Timers* *Typ der Funktion: public func();*

	Erklärung
Rückgabewert	• *OK, wenn erfolgreich* • *ERROR, wenn kein gültiger Index übergeben wurde oder bei einem internen Fehler* • *< OK, wenn ein anderer Fehler auftritt (siehe "Returncodes für allgemeine Zwecke" im Kapitel "Konstanten" auf Seite 108)*

13.3.3 Uplink

13.3.3.1 Arrays mit symbolischen Indizes

TrM2M_GSMInfo

Informationen zum GSM-Modem, SIM-Chip sowie dem bei der letzten Verbindung verwendeten GSM-Netz

```
// cgmi        Manufacturer Identification des Modems
// cgmm        Modem Modellinformation
// cgmr        Modem Revisionsinformation
// imei        International Mobile Equipment Identity des Modems
// imsi        International Mobile Subscriber Identity des SIM-Chips, der für
//             die letzte Verbindung verwendet wurde
//             Leerstring, wenn noch keine Verbindung stattgefunden hat
// iccid       Integrated Circuit Card Identifier des SIM-Chips, der für die
//             letzte Verbindung verwendet wurde
//             Leerstring, wenn noch keine Verbindung stattgefunden hat
// mcc         MCC (Mobile Country Code) des Netzes, das für die letzte/aktuelle
//             Verbindung verwendet wurde
//             0, wenn noch keine Verbindung stattgefunden hat
// mnc         MNC (Mobile Network Code) des Netzes, das für die letzte/aktuelle
//             Verbindung verwendet wurde
//             0, wenn noch keine Verbindung stattgefunden hat
// simstate    Aktueller SIM-Status (siehe "SIM-Status" im Kapitel
//             "Konstanten" auf Seite 115)
// act         Radio Access Technology, die für die letzte/aktuelle Verbindung
//             verwendet wurde (siehe "Mobile Radio AcT" im Kapitel
//             "Konstanten" auf Seite 115)
// lac         LAC (Location Area Code)des Netzes, das für die letzte/aktuelle
//             Verbindung verwendet wurde
// cid         Kennung der Zelle (Cell Identifier)des Netzes, das für die
//             letzte/aktuelle Verbindung verwendet wurde
//             2G AcT:16-bit Zellen ID
//             3G AcT:28-bit UTRAN Zellen ID(16-bit Zellen ID + 12-bit RNC-ID)
//             4G AcT:28-bit E-UTRAN Zellen ID(8-bit Sector ID + 20-bit
//             eNodeB-ID)

#define TrM2M_GSMInfo[  .cgmi{20}, .cgmm{20}, .cgmr{20}, .imei{16}, .imsi{16},
                        .iccid{21}, .mcc, .mnc, .simstate, .act, .lac, .cid   ]
```

TrM2M_TxItfStats

Statistische Informationen zum Uplink-Kommunikationsinterface

```
// rtt         Zeit [ms] die es dauert, bis das Gerät die Antwort auf einen an
//             den Server gesendeten Keep Alive Ping vom Server erhält (round
//             trip time)[1)]

#define TrM2M_TxItfStats [.rtt]
```

[1)] kann nur ermittelt werden, wenn am Server der "Bidirektionale Alive Ping" aktiviert ist. Mittels des "Bidirektionalen Alive Ping" können sowohl Gerät als auch Server einfach erkennen, ob die Verbindung noch besteht. Der "Bidirektionale Alive Ping" kann global für den kompletten Server, für einen speziellen Kunden oder für eine einzelne Site aktiviert werden (siehe "Benutzerhandbuch für myDatanet-Server " 206.886).

13.3.3.2 Konstanten

SIM-Status

```
//Verbindung kann per Device Logic ausgelöst werden
RM2M_SIM_STATE_NONE       = 0,   //Initialzustand
RM2M_SIM_STATE_PRODUCTION = 1,   //Neu produziertes Gerät liegt auf Lager
RM2M_SIM_STATE_HOT        = 2,   //Gültiger Vertrag

//Auslösen der Verbindung per Device Logic nicht möglich
RM2M_SIM_STATE_COLD       = 3,   //Vertragsende oder Fair-Use Verletzung
RM2M_SIM_STATE_DISCARDED  = 4,   //Gerät wurde außer Dienst gestellt
```

Mobile Radio AcT (Access Technology)

```
// Mobile Radio AcT (Access Technology) nach 3GPP TS27.007
RM2M_TX_ACT_GSM                 = 0,   // GSM
RM2M_TX_ACT_GSM_COMPACT,        = 1,   // GSM Compact
RM2M_TX_ACT_UTRAN,              = 2,   // UTRAN
RM2M_TX_ACT_GSM_W_EGPRS,        = 3,   // GSM with EGPRS
RM2M_TX_ACT_UTRAN_W_HSDPA,      = 4,   // UTRAN with HSDPA
RM2M_TX_ACT_UTRAN_W_HSUPA,      = 5,   // UTRAN with HSUPA
RM2M_TX_ACT_UTRAN_W_HSDPA_HSUPA = 6,   // UTRAN with HSDPA and HSUPA
RM2M_TX_ACT_E_UTRAN,            = 7,   // E-UTRAN

//rapidM2M spezifisch
RM2M_TX_ACT_WIFI                = 100, // WiFi
RM2M_TX_ACT_ETH                 = 101, // Ethernet

RM2M_TX_ACT_UNKNOWN             = 255  // Unbekannt
```

Verbindungsflags

Steuerflags für die Funktion rM2M_TxStart()

```
RM2M_TX_POSUPDATE          = 0b00000001, /* Update der GSM-Positionsdaten beim
                                            Verbindungsaufbau                */
RM2M_TX_REFRESH_CONFIG     = 0b00000100, /* zusätzlich eine Verbindung zum
                                            Maintenance Server herstellen    */
RM2M_TX_SUPPRESS_POSUPDATE = 0b00001000, /* Update der GSM-Positionsdaten beim
                                            Verbindungsaufbau unterdrücken 1) */
RM2M_TX_POSUPDATE_ONLY     = 0b00010000, /* Beim Verbindungsaufbau erfolgt nur
                                            ein Update der GSM-Postionsdaten.
                                            Messdaten, Konfigurationen usw.
                                            werden nicht synchronisiert.     */
```

1) Damit lässt sich das alle 24h von der Firmware automatisch ausgeführte Update der GSM-Positionsdaten unterdrücken.

Kommunikationsmodi

Kommunikationsmodi für die Funktion rM2M_TxSetMode()

```
RM2M_TXMODE_TRIG   = 0,          // Intervall
RM2M_TXMODE_WAKEUP = 1,          // Intervall & Wakeup
RM2M_TXMODE_ONLINE = 2,          // Online
```

Kommunikationsmodus-Flags

Konfigurationsflags für die Funktion rM2M_TxSetMode()

```
RM2M_TXMODE_SUPPRESS_SYNC = 0b00000001, /* keine aut. Sync. mit dem Server
                                           bei Änderung der Verbindungsart */
```

Verbindungsstatus

Rückgabewerte der Funktion rM2M_TxGetStatus()

```
RM2M_TX_FAILED           = 0b0000000001, // Verbindungsaufbau fehlgeschlagen
RM2M_TX_ACTIVE           = 0b0000000010, // GPRS-Verbindung besteht
RM2M_TX_STARTED          = 0b0000000100, // Verbindungsaufbau gestartet
RM2M_TX_RETRY            = 0b0000001000, // Wartezeit bis zum Retry
RM2M_TX_WAKEUPABLE       = 0b0000010000, // Modem ins GSM-Netz eingebucht
RM2M_TX_DISABLED         = 0b0001000000, // Modem wurde deaktiviert
RM2M_TX_WAKEUP           = 0b0100000000, /* Verbindungsaufbau durch Wakeup SMS
                                            getriggert */
RM2M_TX_POSUPDATE_ACTIVE = 0b1000000000, // Positionsbestimmung läuft
```

Verbindungsfehlercodes

Fehlercodes, die von der Funktion rM2M_TxGetStatus() über den optionalen Parameter "errorcode" zurückgeliefert werden falls der letzte Verbindungsversuches fehlgeschlagen ist.

```
RM2M_TXERR_NONE = 0,              // no error

// Allgemeine Fehler
RM2M_TXERR_CONNECTION_TIMEOUT,    // connection timed out
RM2M_TXERR_NEWDATA_TIMEOUT,       // timeout during server sync in online mode
RM2M_TXERR_IRREGULAR_OFF,         // irregulary closed connection
RM2M_TXERR_SERVER_NOT_AVAILABLE,  // server not available
RM2M_TXERR_SERVER_COMMUNICATION,  // error during communication with server

// Allgemeine Modem Fehler
RM2M_TXERR_MODEM = 10,            // unspecified modem error
RM2M_TXERR_MODEM_TIMEOUT,         // timeout modem communication
RM2M_TXERR_MODEM_HW_NOT_FOUND,    // modem not found
RM2M_TXERR_MODEM_HW_UNKNOWN,      // unknown modem
RM2M_TXERR_MODEM_INIT,            // error during init
RM2M_TXERR_MODEM_UNS_RESTART,     /* unsolicited restart (e.g. due to weak power
                                     supply) */
RM2M_TXERR_MODEM_RESETLOOP,       // modem reset-loop detected
RM2M_TXERR_MODEM_UNDERVOLTAGE,    // modem undervoltage (power failure) detected
RM2M_TXERR_MODEM_OVERHEAT,        // modem overheat detected

// SIM bezogene Fehler
RM2M_TXERR_MODEM_SIM = 30,        // unspecified SIM related error
RM2M_TXERR_MODEM_SIM_NO_ATTEMPT,  // only one remaining pin input attempt
RM2M_TXERR_MODEM_SIM_PIN_WRONG,   // pin code is wrong
RM2M_TXERR_MODEM_SIM_NO_PIN,      // pin code required but not available
RM2M_TXERR_MODEM_EXTSIM_DENIED,   /* external SIM not allowed (APN and/or
                                     feature key) */
RM2M_TXERR_MODEM_EXTSIM_MISSING,  // external SIM not found
RM2M_TXERR_MODEM_SIM_OTHER,       /* any other problem with SIM card (e.g.
                                     internal SIM not found) */
```

```
// Netzwerkbezogene Fehler (GSM, GPRS, PDP, etc.)
RM2M_TXERR_MODEM_NETWORK = 50,    // unspecified network related error
RM2M_TXERR_MODEM_GSM_BAND_SEL,    // GSM not available (e.g. error with antenna)
RM2M_TXERR_MODEM_NETLOCK,         /* error registering within network (e.g. not
                                     allowed) */
RM2M_TXERR_MODEM_POSUPDATE,       // error with GSM position update
RM2M_TXERR_MODEM_PDP_CTX,         // error activating PDP context

// TCP bezogene Modem Fehler
RM2M_TXERR_MODEM_TCP = 70,        /* TCP error (e.g. timeout, server not
                                     available) */

// Allgemeine Wifi Fehler
RM2M_TXERR_WIFI = 200,            // unspecified WiFi error
RM2M_TXERR_WIFI_TIMEOUT,          // timeout WiFi communication
RM2M_TXERR_WIFI_HW_NOT_FOUND,     // WiFi device not found
RM2M_TXERR_WIFI_INIT,             // error during init
RM2M_TXERR_WIFI_IO,               // error IO communication

// Netzwerkbezogen WiFi Fehler
RM2M_TXERR_WIFI_NETWORK = 220,    // unspecified network related WiFi error
RM2M_TXERR_WIFI_NETWORK_TIMEOUT,  // timeout accessing network
RM2M_TXERR_WIFI_AP_SCAN_TIMEOUT,  /* timeout scanning for available access
                                     points */
RM2M_TXERR_WIFI_AP_SCAN,          /* error scanning access points (e.g.
                                     currently not possible) */
RM2M_TXERR_WIFI_DHCP_TIMEOUT,     /* timeout receiving IP address from DHCP
                                     server */
RM2M_TXERR_WIFI_AP_SETTINGS,      // access point settings not plausible
RM2M_TXERR_WIFI_AP_CONNECT,       // error connecting to access point
RM2M_TXERR_WIFI_AP_NOT_FOUND,     // access point not found during scan

// TCP bezogene WiFi Fehler
RM2M_TXERR_WIFI_TCP = 240,        // unspecified TCP related WiFi error
RM2M_TXERR_WIFI_TCP_OPEN_TO,      // timeout opening TCP connection
RM2M_TXERR_WIFI_TCP_SEND_TO,      // timeout sending data
RM2M_TXERR_WIFI_TCP_CONNECT,      // error connecting to server
RM2M_TXERR_WIFI_TCP_FAILED,       // other error concerning TCP connection

// Allgemeine Ethernet Fehler
RM2M_TXERR_ETH = 300,             // unspecified Ethernet error
RM2M_TXERR_ETH_TIMEOUT,           // timeout Ethernet communication
RM2M_TXERR_ETH_INIT,              // error during init
RM2M_TXERR_ETH_IO,                // error IO communication
RM2M_TXERR_ETH_INIT_MAC_PHY,      // error initialising MAC/PHY interface
RM2M_TXERR_ETH_ITF_UP,            /* TCP/IP stack: error bringing itf up
                                     (including dhcp) */

// Netzwerkbezogen Ethernet Fehler
RM2M_TXERR_ETH_NETWORK = 320,     // unspecified network related Ethernet error
RM2M_TXERR_ETH_NETWORK_TIMEOUT,   // timeout accessing network
RM2M_TXERR_ETH_DHCP_TIMEOUT,      /* timeout receiving IP address from DHCP
                                     server */
```

```
// TCP bezogene Ethernet Fehler
RM2M_TXERR_ETH_TCP = 340,          // unspecified TCP related Ethernet error
RM2M_TXERR_ETH_TCP_OPEN_TIMEOUT,  // timeout opening TCP connection
RM2M_TXERR_ETH_TCP_SEND_TIMEOUT,  // timeout sending data
RM2M_TXERR_ETH_TCP_CONNECT,       // error connecting to server
RM2M_TXERR_ETH_TCP_FAILED,        // other error concerning TCP connection
```

Verfügbare Uplink Interfaces

Wählbare Uplink Interfaces für die Funktion rM2M_TxSelectItf()

```
RM2M_TXITF_NONE    = 0,          /* kein Uplink, Kommunikation mit den Server
                                    nicht möglich */
RM2M_TXITF_MODEM   = 1,          // Mobilfunkmodem
RM2M_TXITF_WIFI    = 2,          // WiFi-Modul
RM2M_TXITF_LAN     = 3,          // LAN-Schnittstelle
```

Signalstärkemessungs-Flags

Steuerflags für die Funktionen rM2M_GSMGetRSSI() und rM2M_GetRSSI().

```
RM2M_RSSI_EXTENDED_VALUE = 0b00000001,  /* aktiviert den erweiterten
                                           Wertebereich(-32768 .. 32767)
                                           für die Rückgabewerte der
                                           Signalstärke */
```

Konfigurationsflags für die Funktion rM2M_CfgInit()

```
RM2M_CFG_VOLATILE  = 0b00000001, // flüchtige Speicherung (RAM)
```

13.3.3.3 Callback Funktionen

public func(const data[], len, timestamp, timestamp256);

vom Device Logic Entwickler bereitzustellende Funktion, die nach dem Lesen eines Datensatzes (mittels der Funktion "rM2M_ReadData()") aus dem internen Flash-Speicher aufgerufen wird.

***Wichtiger Hinweis:** Der Parameter "timestamp256" wurde erst bei späteren Firmware-Versionen hinzugefügt. Prüfen Sie daher mittels der Funktion "numargs()" die Anzahl der von der Firmware an die Callback Funktion übergebenen Argumente.*

Beispiel:

```
#callback readdata_callback(const data{}, len, timestamp, timestamp256)
{
  if(numargs() >= 4)
  {
    // parameter timestamp256 is available ...
  }
}
```

Parameter	*Erklärung*
data	*Array, das die Daten des gelesenen Datensatzes enthält*
len	*Länge des Datenbereichs des gelesenen Datensatzes in Bytes (max. 1024 Byte)*
timestamp	*Zeitstempel des Datensatzes (in UTC)*
timestamp256	*Bruchteil der nächsten begonnenen sec. (Auflösung 1/256 sec.)*

public func(cfg);

vom Device Logic Entwickler bereitzustellende Funktion, die aufgerufen wird, wenn sich einer der Konfigurationsspeicherblöcke geändert hat.

Parameter	*Erklärung*
cfg	*Nummer des Konfigurationsspeicherblocks, beginnend mit 0 für den ersten Speicherblock, der geändert wurde*

13.3.3.4 Funktionen

native rM2M_TxStart(flags=0);

löst eine Verbindung zum Server mit anschließender Synchronisation aller Speicherbereiche (Messdaten, Konfiguration, Positionsdaten, Gerätelog, Files,...) zwischen dem Gerät und dem Server aus. Übertragen werden nur jene Speicherbereiche, deren Inhalt geändert wurde. Wenn sich das Gerät im "online"-Modus befindet und eine aktive Verbindung zum Server besteht, löst diese Funktion nur eine Synchronisation aus. Die bestehende Verbindung wird zuvor nicht getrennt und anschließend wieder aufgebaut.

> ***Wichtiger Hinweis:*** *Im "online"-Modus werden neue Messdaten, die mittels der Funktion "rM2M_RecData()" im internen Flash abgelegt wurden, unverzüglich an den Server übermittelt. Ein Aufrufen der Funktion "rM2M_TxStart()" ist daher für den Transfer der Messdaten in diesem Fall nicht erforderlich. Ein Aufrufen der Funktion und die damit verbundene Synchronisation aller Speicherbereiche nach der Erzeugung eines jeden Messdatensatzes würde zu einem deutlich erhöhten Datenvolumen führen. Selbiges gilt auch für die Übertragung der Konfigurationen. Dennoch ist ein gelegentlicher Aufruf der Funktion "rM2M_TxStart()" (z.B. alle 2h) auch im "online"-Modus zu empfehlen, da nicht alle Speicherbereiche automatisch synchronisiert werden.*

Parameter	***Erklärung***	
flags	*Konfigurationsflags für den Verbindungsaufbau*	
	Bit0 (RM2M_TX_POSUPDATE):	*wenn gesetzt, erfolgt auch ein Update der GSM-Positionsdaten*
	Bit2 (RM2M_TX_REFRESH_ CONFIG):	*wenn gesetzt, erfolgt auch eine Verbindung zum Maintenance Server*
	Bit3 (RM2M_TX_SUPPRESS_ POSUPDATE):	*wenn gesetzt, wird das alle 24h von der Firmware automatisch ausgeführte Update der GSM-Positionsdaten unterdrückt*
	Bit4 (RM2M_TX_POSUPDATE_ ONLY):	*wenn gesetzt, erfolgt beim Verbindungsaufbau nur ein Update der GSM-Positionsdaten. Messdaten, Konfigurationen usw. werden nicht synchronisiert.*

	Erklärung
Rückgabewert	• *OK, wenn erfolgreich* • *ERROR_SIM_STATE, wenn aufgrund des aktuellen SIM-Status eine Verbindung nicht möglich ist (siehe "SIM-Status" im Kapitel "Konstanten" auf Seite 115)* • *ERROR_MODEM_DISABLED, wenn der Verbindungsaufbau aufgrund zu niedriger Versorgungsspannung nicht möglich ist* • *ERROR_TXITF, wenn aufgrund der TX-Interface-Konfiguration eine Verbindung nicht möglich ist (z.B. TX-Interface nicht geöffnet)* • *< OK, wenn ein anderer Fehler auftritt (siehe "Returncodes für allgemeine Zwecke" im Kapitel "Konstanten" auf Seite 108)*

native rM2M_TxSetMode(mode, flags=0);

setzt die zu verwendende Verbindungsart. Wird die Verbindungsart auf "online" oder "Intervall & Wakeup" geändert, erfolgt unverzüglich auch ein Verbindungsaufbau inkl. Synchronisation mit dem Server, sofern dies nicht durch Setzen des Flags "RM2M_TXMODE_SUPPRESS_SYNC" unterdrückt wird. Selbiges gilt auch für die Änderung der Verbindungsart von "Intervall" auf "Intervall & Wakeup".

Parameter	***Erklärung***
mode	*zu verwendende Verbindungsart:* *RM2M_TXMODE_TRIG: Die Verbindung erfolgt beim Aufruf der Funktion "rM2M_TxStart()"* *RM2M_TXMODE_WAKEUP: Die Verbindung erfolgt wie im Modus "intervall" beim Aufruf der Funktion "rM2M_TxStart()". Zusätzlich kann das Gerät über den Server dazu veranlasst werden, sofort eine Verbindung aufzubauen (siehe "Benutzerhandbuch für myDatanet-Server " 206.886). Dazu bucht sich das Gerät unverzüglich ins GSM-Netz ein sobald dieser Modus gesetzt wurde .*  *RM2M_TXMODE_ONLINE: Das Gerät trennt die Verbindung nicht und übermittelt kontinuierlich die Messdaten. Alle 7 Tage wird die Verbindung jedoch kurzzeitig zwecks Überprüfung der Serverzuordung unterbrochen. Der Verbindungsaufbau erfolgt unverzüglich sobald dieser Modus gesetzt wurde. Ein Aufruf der Funktion "rM2M_TxStart()" ist nicht erforderlich.*
flags	*Konfigurationsflags für den Kommunikationsmodus* *Bit0: automatische Sync. mit dem Server bei Änderung der Verbindungsart* *0 = Synchronisation durchführen* *RM2M_TXMODE_SUPPRESS_SYNC = Synchronisation unterdrücken*

	Erklärung
Rückgabewert	• *OK, wenn erfolgreich* • *ERROR_SIM_STATE, wenn aufgrund des aktuellen SIM-Status der Modus nicht möglich ist (siehe "SIM-Status" im Kapitel "Konstanten" auf Seite 115)* • *ERROR_MODEM_DISABLED, wenn der Verbindungsaufbau aufgrund zu niedriger Versorgungsspannung nicht möglich ist* • *ERROR_TXITF, wenn aufgrund der TX-Interface-Konfiguration eine Verbindung nicht möglich ist (z.B. TX-Interface nicht geöffnet)* • *< OK, wenn ein anderer Fehler auftritt (siehe "Returncodes für allgemeine Zwecke" im Kapitel "Konstanten" auf Seite 108)*

Hinweis: *Ergänzende Erklärung zu den Verbindungsarten*

Verbindungsart	*Energieverbrauch*	*Datenvolumen*	*Reaktionszeit*
online			
Intervall & Wakeup			
Intervall			

native rM2M_TxGetStatus(&errorcode=0);

liefert den aktuellen Verbindungsstatus

Parameter	*Erklärung*	
errorcode	*Variable zur Aufnahme des beim letzten Verbindungsversuch aufgetretenen Fehlercodes*	
	RM2M_TXERR_NONE:	*letzter Verbindungsaufbau erfolgreich*
	> RM2M_TXERR_NONE:	*letzter Verbindungsaufbau fehlgeschlagen. Eine detaillierte Aufschlüsselung der Fehlercodes finden Sie unter "Verbindungsfehlercodes" im Kapitel "Uplink" auf Seite 114.*

	Erklärung	
Rückgabewert	*Bit0 (RM2M_TX_FAILED):*	*gesetzt, wenn der letzte GPRS-Verbindungsaufbau fehlgeschlagen ist*
	Bit1 (RM2M_TX_ACTIVE):	*gesetzt, wenn eine GPRS-Verbindung besteht*
	Bit2 (RM2M_TX_STARTED):	*gesetzt, wenn der Verbindungsaufbau gestartet wurde*
	Bit3 (RM2M_TX_RETRY):	*gesetzt, während der Wartezeit bis zum erneuten automatischen Retry bei Verbindungsproblemen*
	Bit4 (RM2M_TX_WAKEUPABLE):	*gesetzt, wenn das Modem ins GSM-Netz eingebucht ist (Wakeup ist möglich)*
	Bit6 (RM2M_TX_DISABLED):	*gesetzt, wenn das Modem deaktiviert wurde*
	Bit8 (RM2M_TX_WAKEUP):	*gesetzt, wenn der Verbindungsaufbau durch den Empfang einer Wakeup SMS ausgelöst wurde*
	Bit9 (RM2M_TX_POSUPDATE_ ACTIVE):	*gesetzt, wenn die Positionsbestimmung läuft*

native rM2M_TxSelectItf(itf);

selektiert das für den Uplink zu verwendende Kommunikationsinterface

Parameter	*Erklärung*
itf	*Auswahl des Kommunikationsinterface* *RM2M_TXITF_NONE:* *kein Uplink, Kommunikation mit dem Server nicht möglich* *RM2M_TXITF_MODEM:* *Mobilfunkmodem* *RM2M_TXITF_WIFI:* *WiFi-Modul* *RM2M_TXITF_LAN:* *LAN-Schnittstelle*

	Erklärung
Rückgabewert	• *OK, wenn erfolgreich* • *ERROR, wenn das gewählte Kommunikationsinterface vom Gerät nicht unterstützt wird oder ein anderer Fehler auftritt*

native rM2M_TxItfGetStats(stats[TrM2M_TxItfStats], len=sizeof stats);

liefert die statistischen Informationen zum Uplink-Kommunikationsinterface

Parameter	*Erklärung*
stats	*Struktur zur Aufnahme der statistischen Informationen (siehe "TrM2M_ TxItfStats" im Kapitel "Arrays mit symbolischen Indizes" auf Seite 114)*
len	*Größe (in Cells) der Struktur zur Aufnahme der statistischen Informationen - OPTIONAL*

	Erklärung
Rückgabewert	• *OK, wenn erfolgreich*

native rM2M_SetTCPKeepAlive(time=0);

setzt das Intervall, in dem die Keep Alive Pings während des online-Modus gesendet werden

Parameter	***Erklärung***
time	*zeitlicher Abstand, in dem die Keep Alive Pings gesendet werden* *0: in der Firmware hinterlegte Standardeinstellung wird verwendet (15min. 3sec.)* *< 241: in 1sec. Schritten* *241 .. 255: in 5min. Schritten, wobei anschließend 3sec. addiert werden* *z.B. 243: 3*5min + 3 sec. = 15min. 3sec.*

	Erklärung
Rückgabewert	• *OK, wenn erfolgreich*

native rM2M_GSMGetRSSI(flags=0);

liefert die GSM/UMTS/LTE-Signalstärke

Wichtiger Hinweis: *Diese Funktion wird zwar weiterhin zugunsten der Abwärtskompatibilität unterstützt, sollte aber bei neuen Projekten nicht mehr verwendet werden. Alternativ sollte die Funktion "rM2M_GetRSSI()" verwendet werden.*

Parameter	***Erklärung***
flags	*Konfigurationsflags für die Signalstärkemessung* *Bit0 (RM2M_RSSI_EXTENDED_VALUE): wenn gesetzt, wird der erweiterte Wertebereich für die Rückgabe der Signalstärke verwendet*

	Erklärung
Rückgabewert	*Signalstärke in [dBm]* *RM2M_RSSI_EXTENDED_VALUE nicht gesetzt:* • *maximaler Wertebereich: -127...127* • *Out of Range bei: -128* *RM2M_RSSI_EXTENDED_VALUE gesetzt:* • *maximaler Wertebereich: -32767 ... 32767* • *Out of Range bei: -32768* *GSM-Werte sind im Bereich -113 ... -51 dBm.* *UMTS-Werte sind im Bereich -116 ... -54 dBm.* *LTE-Werte sind im Bereich -141 ... -44 dBm.*

native rM2M_GetRSSI(flags=0);

liefert die Signalstärke am für den Uplink verwendeten Kommunikationsinterface

	Erklärung
Rückgabewert	*Signalstärke in [dBm]* *RM2M_RSSI_EXTENDED_VALUE nicht gesetzt:* • *maximaler Wertebereich: -127...127* • *Out of Range bei: -128* *RM2M_RSSI_EXTENDED_VALUE gesetzt:* • *maximaler Wertebereich: -32767 ... 32767* • *Out of Range bei: -32768* *GSM-Werte sind im Bereich -113 ... -51 dBm.* *UMTS-Werte sind im Bereich -116 ... -54 dBm.* *LTE-Werte sind im Bereich -141 ... -44 dBm.* *Bei Verwendung des LAN-Interface ist der Rückgabewert 0 dBm.*

native rM2M_GSMGetInfo(info[TrM2M_GSMInfo], len=sizeof info);

liefert Informationen zum GSM-Modem, SIM-Chip sowie dem bei der letzten Verbindung verwendeten GSM-Netz

Parameter	***Erklärung***
info	*Struktur zur Aufnahme der Informationen (siehe "TrM2M_GSMInfo" im Kapitel "Arrays mit symbolischen Indizes" auf Seite 114)*
len	*Größe (in Cells) der Struktur zur Aufnahme der Informationen - OPTIONAL*

	Erklärung
Rückgabewert	• *Verwendete Größe (in Cells) der Struktur zur Aufnahme der Informationen* • *ERROR, wenn Adresse und/oder Länge der Info-Struktur ungültig sind (außerhalb des Skript-Datenspeichers)* • *< OK, wenn ein anderer Fehler auftritt (siehe "Returncodes für allgemeine Zwecke" im Kapitel "Konstanten" auf Seite 108)*

native rM2M_LiveData(const data{}, len);

überträgt einen Datensatz als Live-Daten an den Server. Ein Aufruf dieser Funktion ist nur gestattet, wenn sich das Gerät im "online"-Modus befindet und eine aktive Verbindung zum Server besteht. Benutzen Sie die Funktionen "rM2M_Pack", "rM2M_SetPacked" oder "rM2M_SetPackedB", um den Datenbereich zu erzeugen.

Parameter	***Erklärung***
data	*Array, das die zu übertragenden Live-Daten enthält* ***Wichtiger Hinweis:*** *Der Aufbau der Live-Daten muss exakt jenem der mittels der Funktion "rM2M_RecData()" im internen Flash abgelegten Messdaten entsprechen.*
len	*Anzahl der zu übertragenden Bytes (max. 1024 Byte)*

	Erklärung
Rückgabewert	• *OK, wenn erfolgreich* • *ERROR, wenn ein Fehler auftritt (z.B. Der Server unterstützt den Empfang von Live-Daten nicht.)*

native rM2M_RecData(timestamp, const data{}, len);

speichert einen Datensatz im internen Flash-Speicher. Benutzen Sie die Funktionen "rM2M_Pack", "rM2M_SetPacked" oder "rM2M_SetPackedB", um den Datenbereich zu erzeugen.

Parameter	***Erklärung***
timestamp	*Zeitstempel, der für die Aufzeichnung verwendet werden soll* *= 0: Die aktuelle Systemzeit wird als Zeitstempel verwendet.* *> 0: Der übergebene Zeitstempel wird verwendet.* *(Der Zeitstempel muss in Sekunden seit 31.12.1999 angegeben werden)*
data	*Array, das die zu speichernden Daten enthält*
len	*Anzahl der zu speichernden Bytes (max. 1024 Byte)*

	Erklärung

Rückgabewert	• *OK, wenn erfolgreich*
	• *-2, wenn die Datenspeicherung aktuell nicht möglich ist, weil der interne Speicher reorganisiert wird. Die Daten müssen im Skript zwischengespeichert und zu einem späteren Zeitpunkt erneut gesichert werden.*
	• *ERROR, wenn einer der folgenden Fehler auftritt*
	  ◦ *Speicherbereich (data{}, len) ist ungültig.*
	  ◦ *Mehr als 10 Aufrufe in einem Skriptdurchlauf*
	  ◦ *Anzahl der zu speichernden Bytes > 1024 Byte*
	  ◦ *FLASH-Schreibvorgang nicht erfolgreich*
	  ◦ *Übergabeparameter "timestamp" liegt mehr als 5 Minuten in der Zukunft*
	• *< OK, wenn ein anderer Fehler auftritt (siehe "Returncodes für allgemeine Zwecke" im Kapitel "Konstanten" auf Seite 108)*

native rM2M_ReadData(recidx, funcidx);

liest einen im internen Flash gespeicherten Datensatz aus und ruft daraufhin die Funktion, deren Index übergeben wurde, auf.

Parameter	*Erklärung*
recidx	*Index des zu lesenden Datensatzes (-1 = letzter/aktuellster Datensatz, -2 = vorletzter Datensatz, ...)*
funcidx	*Index der öffentlichen Funktion, die im Anschluss an das Lesen des Datensatzes aus dem internen Flash-Speicher aufgerufen werden soll.* *Typ der Funktion: public func(const data[], len, timestamp, timestamp256);*

	Erklärung
Rückgabewert	• *OK, wenn der Lesevorgang gestartet wurde* • *ERROR, wenn ein Fehler auftritt*

native rM2M_CfgInit(cfg, flags);

legt die Konfiguration für einen Konfigurationsspeicherblock fest. Ein Aufruf der Funktion ist nur notwendig, wenn eines der Konfigurationsflags gesetzt werden soll.

Parameter	*Erklärung*
cfg	*Nummer des Konfigurationsspeicherblocks, beginnend mit 0 für den ersten Speicherblock. Das Gerät verfügt über 10 voneinander unabhängige Speicherblöcke.*
flags	*Zu setzende/löschende Konfigurationsflags* *Bit0: Art der Speicherung* *0 (default) = nichtflüchtig im FLASH gespeichert* *RM2M_CFG_VOLATILE = flüchtig im RAM gespeichert*

	Erklärung
Rückgabewert	• *OK, wenn erfolgreich* • *< OK, wenn ein Fehler auftritt (siehe "Returncodes für allgemeine Zwecke" im Kapitel "Konstanten" auf Seite 108)*

Hinweis: *Ergänzende Erklärung zur Art der Speicherung:*

Wenn Bit 0 nicht gesetzt wurde (default), wird beim Aufruf der Funktion "rM2M_CfgWrite" der Konfigurationsspeicherblock nicht flüchtig im FLASH gespeichert.

Wenn Bit0 gesetzt wurde (Bit0 = RM2M_CFG_VOLATILE), wird beim Aufruf der Funktion "rM2M_CfgWrite" der Konfigurationsspeicherblock flüchtig im RAM gespeichert. Diese Option ist zu empfehlen, wenn sich die Daten im Konfigurationsspeicherblock häufig ändern, da dadurch die Anzahl der Flash-Schreibzyklen reduziert wird. Um den Konfigurationsspeicherblock nicht flüchtig im FLASH zu speichern, muss die Funktion "rM2M_CfgFlush" aufgerufen werden.

native rM2M_CfgWrite(cfg, pos, const data{}, size);

speichert den übergebenen Datenblock an der angegeben Position in einem Konfigurationsspeicherblock. Beachten Sie, dass der Konfigurationspeicherblock abhängig von der mit Hilfe der Funktion "rM2M_CfgInit" ausgewählten Art der Speicherung gespeichert wird, entweder flüchtig im RAM (Bit0 = RM2M_CFG_VOLATILE) oder nichtflüchtig im FLASH (Bit0 = 0, default). Der Funktion wird auch übergeben, welcher der 10 verfügbaren Speicherblocks im internen Flash-Speicher verwendet werden soll. Benutzen Sie die Funktionen "rM2M_Pack", "rM2M_SetPacked" oder "rM2M_SetPackedB", um den zu speichernden Datenblock zu erzeugen. Der Zeitstempel wird aktualisiert, wodurch der Konfigurationsspeicherblock bei der nächsten Verbindung automatisch mit dem myDatanet-Server synchronisiert wird.

Parameter	***Erklärung***
cfg	*Nummer des Konfigurationsspeicherblocks, beginnend mit 0 für den ersten Speicherblock. Das Gerät verfügt über 10 voneinander unabhängige Speicherblöcke.*
pos	*Byteoffset innerhalb des Konfigurationsspeicherblocks zur Bestimmung der Position, an die die Daten geschrieben werden sollen*
data	*Array, das die Daten, die in den Konfigurationsspeicherblock geschrieben werden sollen, enthält*
size	*Anzahl der Bytes, die in den Konfigurationsspeicherblock geschrieben werden sollen*

	Erklärung
Rückgabewert	• *> 0: Aktuelle Größe des Konfigurationsspeicherblocks wenn erfolgreich* • *ERROR_MEM, wenn momentan nicht genügend temporärer Speicher (RAM) verfügbar ist. (Kann auftreten, wenn für mehrere Konfigurationsspeicherblöcke als Art der Speicherung "flüchtig im RAM" gewählt wurde.)* • *< OK, wenn ein anderer Fehler auftritt (siehe "Returncodes für allgemeine Zwecke" im Kapitel "Konstanten" auf Seite 108)*

native rM2M_CfgFlush(cfg);

speichert den Konfigurationsspeicherblock dessen Nummer übergeben wurde nicht flüchtig im FLASH. Ein Aufruf der Funktion ist nur dann notwendig, wenn mittels der Funktion "rM2M_CfgInit" als Art der Speicherung für den betreffenden Konfigurationsspeicherblock "flüchtig im RAM (Bit0 = RM2M_CFG_VOLATILE)" ausgewählt wurde.

Parameter	***Erklärung***
cfg	*Nummer des Konfigurationsspeicherblocks, beginnend mit 0 für den ersten Speicherblock. Das Gerät verfügt über 10 voneinander unabhängige Speicherblöcke.*

	Erklärung
Rückgabewert	• *OK, wenn erfolgreich* • *< OK, wenn ein Fehler auftritt (siehe "Returncodes für allgemeine Zwecke" im Kapitel "Konstanten" auf Seite 108)*

native rM2M_CfgRead(cfg, pos, data{}, size);

liest einen Datenblock von der angegeben Position aus einem Konfigurationsspeicherblock. Der Funktion wird auch übergeben, von welchem der 10 verfügbaren Speicherblocks im internen Flash-Speicher gelesen werden soll. Benutzen Sie die Funktionen "rM2M_Pack", "rM2M_GetPacked" oder "rM2M_GetPackedB", um gelesene Daten zu entpacken.

Parameter	***Erklärung***
cfg	*Nummer des Konfigurationsspeicherblocks, beginnend mit 0 für den ersten Speicherblock. Das Gerät verfügt über 10 voneinander unabhängige Speicherblöcke.*
pos	*Byteoffset innerhalb des Konfigurationsspeicherblocks zur Bestimmung der Position, von der die Daten gelesen werden soll*
data	*Array zur Aufnahme der zu lesenden Daten*
size	*Anzahl der Bytes, die aus dem Konfigurationsspeicherblock zu lesen sind*

	Erklärung
Rückgabewert	• *> 0: Anzahl der Bytes, die tatsächlich gelesen wurden. Diese kann kleiner oder gleich der angefragten Anzahl an Bytes sein.* • *ERROR_MEM, wenn momentan nicht genügend temporärer Speicher (RAM) verfügbar ist. (Kann auftreten, wenn für mehrere Konfigurationsspeicherblöcke als Art der Speicherung "flüchtig im RAM" gewählt wurde.)* • *< OK, wenn ein anderer Fehler auftritt (siehe "Returncodes für allgemeine Zwecke" im Kapitel "Konstanten" auf Seite 108)*

native rM2M_CfgDelete(cfg);

löscht alle Daten des übergebenen Konfigurationsspeicherblocks

Parameter	***Erklärung***
cfg	*Nummer des Konfigurationsspeicherblocks, beginnend mit 0 für den ersten Speicherblock. Das Gerät verfügt über 10 voneinander unabhängige Speicherblöcke.*

	Erklärung
Rückgabewert	• *OK, wenn erfolgreich* • *ERROR_MEM, wenn momentan nicht genügend temporärer Speicher (RAM) verfügbar ist. (Kann auftreten, wenn für mehrere Konfigurationsspeicherblöcke als Art der Speicherung "flüchtig im RAM" gewählt wurde.)* • *< OK, wenn ein anderer Fehler auftritt (siehe "Returncodes für allgemeine Zwecke" im Kapitel "Konstanten" auf Seite 108)*

native rM2M_CfgOnChg(funcidx);

legt die Funktion fest, die aufgerufen werden soll, wenn sich einer der Konfigurationsspeicherblöcke geändert hat

Parameter	*Erklärung*
funcidx	*Index der öffentlichen Funktion, die aufgerufen werden soll, wenn sich die Konfiguration geändert hat* *Typ der Funktion: public func(cfg);*

	Erklärung
Rückgabewert	• *OK, wenn erfolgreich* • *ERROR, wenn kein gültiger Index einer öffentlichen Funktion übergeben wurde* • *< OK, wenn ein anderer Fehler auftritt (siehe "Returncodes für allgemeine Zwecke" im Kapitel "Konstanten" auf Seite 108)*

13.3.4 Encoding

13.3.4.1 Konstanten

Konfigurierungsflags für die Funktion rM2M_Pack()

```
RM2M_PACK_GET      = 0b00000001, // Wert soll gelesen werden (Get Packed)
RM2M_PACK_BE       = 0b00000010, // "Big Endian"-Format verwenden
RM2M_PACK_U8       = 0b00010000, // 8-Bit Unsigned
RM2M_PACK_S8       = 0b10010000, // 8-Bit Signed
RM2M_PACK_U16      = 0b00100000, // 16-Bit Unsigned
RM2M_PACK_S16      = 0b10100000, // 16-Bit Signed
RM2M_PACK_U32      = 0b01000000, // 32-Bit Unsigned
RM2M_PACK_S32      = 0b11000000, // 32-Bit Signed
RM2M_PACK_F32      = 0b01000000, // 32-Bit Float
```

13.3.4.2 Funktionen

native rM2M_SetPacked(data{}, pos, &{Float,Fixed,_}:value, size=4, bool:bigendian=false);

schreibt den übergebenen Wert an die angegebene Position in ein Array

> ***Wichtiger Hinweis:*** *Diese Funktion wird zwar weiterhin zugunsten der Abwärtskompatibilität unterstützt, sollte aber bei neuen Projekten nicht mehr verwendet werden, da es bei Signed-Datentypen zu Problemen kommen kann. Alternativ sollte die Funktion "rM2M_Pack()" verwendet werden.*

Parameter	***Erklärung***
data	*Array, das als Datenbereich für einen Datensatz oder eine Konfiguration verwendet werden soll*
pos	*Byteoffset innerhalb des Arrays zur Bestimmung der Position, an die der Wert geschrieben werden soll*
value	*Wert, der in das Array geschrieben werden soll*
size	*Anzahl der Bytes, die für den zu schreibenden Wert verwendet werden sollen*
bigendian	*Einstellung, für die zu verwendende Byte-Reihenfolge beim Schreiben des Werts:* *true: "Big Endian" wird verwendet* *false: "Little Endian" wird verwendet*

	Erklärung
Rückgabewert	• *OK, wenn erfolgreich* • *< OK, wenn ein Fehler auftritt (siehe "Returncodes für allgemeine Zwecke" im Kapitel "Konstanten" auf Seite 108)*

Hinweis: *Ergänzende Erklärung zur Byte-Reihenfolge:*

Im folgenden Beispiel wird die Ganzzahl 439.041.101 als 32-Bit-Integer-Wert ab Speicheradresse 10000 gespeichert.

	Big Endian			***Little Endan***		
Adressen	***Hex***	***Dez***	***Binär***	***Hex***	***Dez***	***Binär***
10000	*1A*	*26*	*00011010*	*4D*	*77*	*01001101*
10001	*2B*	*43*	*00101011*	*3C*	*60*	*00111100*
10002	*3C*	*60*	*00111100*	*2B*	*43*	*00101011*
10003	*4D*	*77*	*01001101*	*1A*	*26*	*00011010*

native rM2M_SetPackedB(data{}, pos, const block{}, size);

schreibt den übergebenen Datenblock an die angegebene Position in ein Array

Parameter	*Erklärung*
data	*Array, das als Datenbereich für einen Datensatz oder eine Konfiguration verwendet werden soll*
pos	*Byteoffset innerhalb des Arrays zur Bestimmung der Position, an die der Datenblock geschrieben werden soll*
block	*Datenblock, der in das Array geschrieben werden soll*
size	*Anzahl der Bytes, die vom Datenblock in das Array geschrieben werden sollen*

	Erklärung
Rückgabewert	• *OK, wenn erfolgreich* • *< OK, wenn ein Fehler auftritt (siehe "Returncodes für allgemeine Zwecke" im Kapitel "Konstanten" auf Seite 108)*

native rM2M_GetPacked(const data{}, pos, &{Float,Fixed,_}:value, size=4, bool:bigendian=false);

liefert den Wert, der sich an der angegebenen Position im einem Array befindet

Wichtiger Hinweis: *Diese Funktion wird zwar weiterhin zugunsten der Abwärtskompatibilität unterstützt, sollte aber bei neuen Projekten nicht mehr verwendet werden, da es bei Signed-Datentypen zu Problemen kommen kann. Alternativ sollte die Funktion "rM2M_Pack()" verwendet werden.*

Parameter	***Erklärung***
data	*Array, das als Datenbereich für einen Datensatz oder eine Konfiguration verwendet werden soll*
pos	*Byteoffset innerhalb des Arrays zur Bestimmung der Position, von der die Daten gelesen werden sollen*
value	*Variable zur Aufnahme der zu lesenden Daten*
size	*Anzahl der Bytes, die zu lesen sind*
bigendian	*Gibt an, wie die gepackten Daten zu interpretieren sind:* *true: Die Daten sind im "Big Endian"-Format im Array gespeichert.* *false: Die Daten sind im "Little Endian"-Format im Array gespeichert.*

	Erklärung
Rückgabewert	• *OK, wenn erfolgreich* • *< OK, wenn ein Fehler auftritt (siehe "Returncodes für allgemeine Zwecke" im Kapitel "Konstanten" auf Seite 108)*

Hinweis: *Ergänzende Erklärung zur Byte-Reihenfolge:*

Im folgenden Beispiel wird die Ganzzahl 439.041.101 als 32-Bit-Integer-Wert ab Speicheradresse 10000 gespeichert.

	Big Endian			***Little Endan***		
Adressen	***Hex***	***Dez***	***Binär***	***Hex***	***Dez***	***Binär***
10000	*1A*	*26*	*00011010*	*4D*	*77*	*01001101*
10001	*2B*	*43*	*00101011*	*3C*	*60*	*00111100*
10002	*3C*	*60*	*00111100*	*2B*	*43*	*00101011*
10003	*4D*	*77*	*01001101*	*1A*	*26*	*00011010*

native rM2M_GetPackedB(const data{}, pos, block{}, size);

liest einen Datenblock, der sich an der angegebenen Position in einem Array befindet

Parameter	***Erklärung***
data	*Array, das als Datenbereich für einen Datensatz oder eine Konfiguration verwendet werden soll*
pos	*Byteoffset innerhalb des Arrays zur Bestimmung der Position, von der die Daten gelesen werden sollen*
block	*Array zur Aufnahme der zu lesenden Daten*
size	*Anzahl der Bytes, die zu lesen sind*

	Erklärung
Rückgabewert	• *OK, wenn erfolgreich* • *< OK, wenn ein Fehler auftritt (siehe "Returncodes für allgemeine Zwecke" im Kapitel "Konstanten" auf Seite 108)*

native rM2M_Pack(const data{}, pos, &{Float,Fixed,_}:value, type);

Funktion für den Zugriff auf gepackte Daten. Wurde das Bit0 (RM2M_PACK_GET) des Parameters "type" gesetzt, liefert die Funktion den Wert, der sich an der angegebenen Position im Array befindet. Andernfalls schreibt die Funktion den übergebenen Wert an die angegebene Position ins Array.

Parameter	***Erklärung***
data	*Array mit den gepackten Inhalten* *Set Packed: Array, in das der Wert geschrieben werden soll* *Get Packed: Array, aus dem der Wert gelesen werden soll*
pos	*Byteoffset innerhalb des Arrays* *Set Packed: Position, an die der Wert geschrieben werden soll* *Get Packed: Position, von der der Wert gelesen werden soll*
value	*Set Packed: Wert, der in das Array geschrieben werden soll* *Get Packed: Variable zum Speichern der zu lesenden Daten*
type	*Konfigurierungsflags für die Funktion* *Bit0: Auswahl Set Packed / Get Packed* *0 = Wert soll geschrieben werden* *1 = Wert soll gelesen werden* *Bit1: Byte-Reihenfolge* *0 = "Little Endian"-Format* *1 = "Big Endian"-Format* *Bit2...3* *reserviert für Erweiterungen* *Bit4...7: Datentyp* *1 = 8-Bit Unsigned* *2 = 16-Bit Unsigned* *4 = 32-Bit Unsigned / 32-Bit Float* *9 = 8-Bit Signed* *10 = 16-Bit Signed* *12 = 32-Bit Signed* ***Hinweis:** Sie können für diesen Parameter auch die vordefinierten Konstanten verwenden (siehe "Konfigurierungsflags für die Funktion rM2M_Pack()" im Kapitel "Konstanten" auf Seite 130). Die Konstanten lassen sich auch durch "oder"-Verknüpfung kombinieren.*

	Erklärung
Rückgabewert	• *OK, wenn erfolgreich* • *< OK, wenn ein Fehler auftritt (siehe "Returncodes für allgemeine Zwecke" im Kapitel "Konstanten" auf Seite 108)*

13.3.5 Registry

13.3.5.1 Konstanten

Indizes der Registrierungsspeicherblöcke

auf die mittels der Funktionen "rM2M_RegGetString()", "rM2M_RegGetValue()", "rM2M_RegSetString()", "rM2M_RegSetValue()", "rM2M_RegDelValue()" und "rM2M_RegDelKey()" zugegriffen werden kann. Detaillierte Informationen zu den Registrierungsspeicherblöcken finden Sie im Kapitel "Registrierungsspeicherblöcke" auf Seite 36.

```
//Systemspezifische Daten
RM2M_REG_SYS_OTP   = 0, /* Einmalig im Zuge des Produktionsprozesses
                           geschrieben (read only für die Device Logic). */
RM2M_REG_SYS_FLASH = 1, /* im Betrieb veränderbar (read only für die
                           Device Logic) */

//Applikationsspezifische Daten
RM2M_REG_APP_OTP   = 2, /* Empfehlung: Einmalig im Zuge der Produktion
                           geschrieben (Lese- und Schreibberechtigung
                           durch die Device Logic) */
RM2M_REG_APP_FLASH = 3, /* im Betrieb veränderbar (Lese- und
                           Schreibberechtigung durch die Device Logic) */

//Applikationsspezifische, flüchtige Daten
RM2M_REG_APP_STATE = 4, /* im Betrieb veränderbar (Lese- und
                           Schreibberechtigung durch die Device Logic).
                           Erfordert "rM2M_RegInit()" */

//Anzahl der Registrierungsspeicherblöcke
RM2M_REG_NUM_REGS  = 5,
```

Fehlercodes für die Registrierungsspeicherblock-Zugriffsoperationen

```
RM2M_REG_ERROR_TOKENMEM  = -101,  // Not enough tokens were provided
RM2M_REG_ERROR_INVAL     = -102,  // Invalid character inside JSON string
RM2M_REG_ERROR_PART      = -103,  /* The string is not a full JSON packet, more
                                     bytes expected */

RM2M_REG_ERROR_NOMEM     = -200,  // memory allocation failed
RM2M_REG_ERROR_NUMTOKENS = -201,  /* not enough token available for this
                                     object/array size */
RM2M_REG_ERROR_PAIR      = -202,  // found invalid pair (string : value)
RM2M_REG_ERROR_NOTOKENS  = -203,  // not enough tokens free for appending
RM2M_REG_ERROR_NOTFOUND  = -204,  // specified pair not found
RM2M_REG_ERROR_TYPE      = -205,  // token type mismatch
RM2M_REG_ERROR_PARAM     = -206,  // invalid parameters
RM2M_REG_ERROR_SIZE      = -207,  // size exceeds maximum allowed
RM2M_REG_ERROR_INVALID   = -208,  // JSON structure invalid
RM2M_REG_ERROR_ISNULL    = -209,  // value is null
```

Konfigurierungsflags für die Funktion rM2M_RegInit()

```
RM2M_REG_VOLATILE  = 0b00000001, // flüchtige Speicherung (RAM)
```

13.3.5.2 Callback Funktionen

public func(reg);

vom Device Logic Entwickler bereitzustellende Funktion, die aufgerufen wird, wenn sich die Registrierung geändert hat

Parameter	*Erklärung*
reg	*Index des Registrierungsspeicherblocks (siehe "Indizes der Registrierungsspeicherblöcke" im Kapitel "Konstanten" auf Seite 136), der geändert wurde*

13.3.5.3 Funktionen

native rM2M_RegInit(reg, flags, data{}, len=sizeof data);

initialisiert einen der optionalen Registrierungsspeicherblöcke, die im RAM abgelegt werden. Ein Aufruf der Funktion ist nur für die bei der Erklärung des Parameters "reg" angeführten Registrierungsspeicherblöcke notwendig. Detaillierte Informationen zu den Registrierungsspeicherblöcken finden Sie im Kapitel "Registrierungsspeicherblöcke" auf Seite 36.

Parameter	*Erklärung*
reg	*Index des Registrierungsspeicherblocks* *Folgende Registrierungsspeicherblöcke erfordern eine Initialisierung:* • *RM2M_REG_APP_STATE: Applikationsspezifische, flüchtige Daten (z.B. aktueller Gerätestatus)*
flags	*Zu setzende/löschende Konfigurationsflags* *Bit0: Art der Speicherung* *0 = ungültig, wird derzeit nicht unterstützt* *RM2M_REG_VOLATILE = flüchtig im RAM gespeichert*
data	*Array zur Aufnahme des Registrierungsspeicherblocks*
len	*Größe (in Cells) des übergebenen Arrays zur Aufnahme des Registrierungsspeicherblocks (max. 1kB) - OPTIONAL*

	Erklärung
Rückgabewert	• *OK, wenn erfolgreich* • *ERROR, wenn ein unspezifizierter Fehler auftritt* • *< OK, wenn ein anderer Fehler auftritt (siehe "Fehlercodes für die Registrierungsspeicherblock-Zugriffsoperationen" im Kapitel "Konstanten" auf Seite 136)*

native rM2M_RegGetString(reg, const name[], string[], len=sizeof string);

liest eine Zeichenkette aus einem Registrierungsspeicherblock. Detaillierte Informationen zu den Registrierungsspeicherblöcken finden Sie im Kapitel "Registrierungsspeicherblöcke" auf Seite 36.

Parameter	***Erklärung***
reg	*Index des Registrierungsspeicherblocks (siehe "Indizes der Registrierungsspeicherblöcke" im Kapitel "Konstanten" auf Seite 136)* ***Hinweis:*** *RM2M_REG_APP_STATE erfordert vorher "rM2M_RegInit ()".*
name	*Name des Eintrags*
string	*Array zur Aufnahme des zu lesenden Strings* *(Erweitertes JSON Stringformat, siehe "rM2M_RegSetString ()" für weitere Details)*
len	*Größe (in Cells) des übergebenen Arrays zur Aufnahme des zu lesenden Strings - OPTIONAL*

	Erklärung
Rückgabewert	• *OK, wenn erfolgreich* • *ERROR, wenn ein unspezifizierter Fehler auftritt* • *RM2M_REG_ERROR_NOTFOUND, wenn der angegebene Eintrag nicht existiert* • *RM2M_REG_ERROR_ISNULL, wenn der Wert des angegebenen Eintrags auf "null" gesetzt wurde* • *< OK, wenn ein anderer Fehler auftritt (siehe "Fehlercodes für die Registrierungsspeicherblock-Zugriffsoperationen" im Kapitel "Konstanten" auf Seite 136)*

native rM2M_RegGetValue(reg, const name[], &{Float,Fixed,_}:value, tag=tagof value);

liest einen Wert aus einem Registrierungsspeicherblock. Detaillierte Informationen zu den Registrierungsspeicherblöcken finden Sie im Kapitel "Registrierungsspeicherblöcke" auf Seite 36.

Parameter	***Erklärung***
reg	*Index des Registrierungsspeicherblocks (siehe "Indizes der Registrierungsspeicherblöcke" im Kapitel "Konstanten" auf Seite 136)* ***Hinweis:*** *RM2M_REG_APP_STATE erfordert vorher "rM2M_RegInit ()".*
name	*Name des Eintrags*
value	*Variable zur Aufnahme des zu lesenden Werts*
tag	*Anhand des "tag" der Variablen wird zwischen Integer oder Gleitkomma Konvertierung differenziert. - OPTIONAL*

	Erklärung
Rückgabewert	• *OK, wenn erfolgreich* • *ERROR, wenn ein unspezifizierter Fehler auftritt* • *RM2M_REG_ERROR_NOTFOUND, wenn der angegebene Eintrag nicht existiert* • *RM2M_REG_ERROR_ISNULL, wenn der Wert des angegebenen Eintrags auf "null" gesetzt wurde* • *< OK, wenn ein anderer Fehler auftritt (siehe "Fehlercodes für die Registrierungsspeicherblock-Zugriffsoperationen" im Kapitel "Konstanten" auf Seite 136)*

native rM2M_RegSetString(reg, const name[], const string[]);

schreibt eine Zeichenkette in einen Registrierungsspeicherblock. Detaillierte Informationen zu den Registrierungsspeicherblöcken finden Sie im Kapitel "Registrierungsspeicherblöcke" auf Seite 36.

> ***Wichtiger Hinweis:*** *Diese Funktion erlaubt auch Zeichen wie "\t" , "\n" ,..., welche nach JSON Standards verboten sind. Deshalb wird das Javascript JSON.parse() mit Fehlermeldungen fehlschlagen. Verwenden Sie deshalb JSON5 um solche erweiterte Strings zu entschlüsseln.*

Parameter	***Erklärung***
reg	*Index des Registrierungsspeicherblocks (siehe "Indizes der Registrierungsspeicherblöcke" im Kapitel "Konstanten" auf Seite 136)* ***Hinweis:*** *RM2M_REG_APP_STATE erfordert vorher "rM2M_RegInit ()".*
name	*Name des Eintrags* *Wenn bereits ein Eintrag mit diesem Namen existiert, wird die bestehende Zeichenkette durch die übergebene Zeichenkette ersetzt. Andernfalls wird ein neuer Eintrag angelegt.*
string	*Array, das den zu schreibenden String enthält*

	Erklärung
Rückgabewert	• *OK, wenn erfolgreich* • *ERROR, wenn ein unspezifizierter Fehler auftritt* • *< OK, wenn ein anderer Fehler auftritt (siehe "Fehlercodes für die Registrierungsspeicherblock-Zugriffsoperationen" im Kapitel "Konstanten" auf Seite 136)*

native rM2M_RegSetValue(reg, const name[], {Float,Fixed,_}:value, tag=tagof value);

schreibt einen Wert in einen Registrierungsspeicherblock. Detaillierte Informationen zu den Registrierungsspeicherblöcken finden Sie im Kapitel "Registrierungsspeicherblöcke" auf Seite 36.

Parameter	*Erklärung*
reg	*Index des Registrierungsspeicherblocks (siehe "Indizes der Registrierungsspeicherblöcke" im Kapitel "Konstanten" auf Seite 136)* ***Hinweis:** RM2M_REG_APP_STATE erfordert vorher "rM2M_RegInit ()".*
name	*Name des Eintrags* *Wenn bereits ein Eintrag mit diesem Namen existiert, wird der bestehende Wert durch den übergebenen Wert ersetzt. Andernfalls wird ein neuer Eintrag angelegt.*
value	*zu schreibender Wert*
tag	*Anhand des "tag" des Wertes wird zwischen Integer- oder Gleitkomma-Konvertierung differenziert. - OPTIONAL*

	Erklärung
Rückgabewert	• *OK, wenn erfolgreich* • *ERROR, wenn ein unspezifizierter Fehler auftritt* • *< OK, wenn ein anderer Fehler auftritt (siehe "Fehlercodes für die Registrierungsspeicherblock-Zugriffsoperationen" im Kapitel "Konstanten" auf Seite 136)*

native rM2M_RegDelValue(reg, const name[]);

sucht einen Eintrag anhand seines Namens und setzt den Wert dieses Eintrags (egal ob String oder Value) auf "null". Detaillierte Informationen zu den Registrierungsspeicherblöcken finden Sie im Kapitel "Registrierungsspeicherblöcke" auf Seite 36.

Parameter	*Erklärung*
reg	*Index des Registrierungsspeicherblocks (siehe "Indizes der Registrierungsspeicherblöcke" im Kapitel "Konstanten" auf Seite 136)*
name	*Name des Eintrags dessen Wert auf "null" gesetzt werden soll*

	Erklärung
Rückgabewert	• *OK, wenn erfolgreich* • *ERROR, wenn ein unspezifizierter Fehler auftritt* • *< OK, wenn ein anderer Fehler auftritt (siehe "Fehlercodes für die Registrierungsspeicherblock-Zugriffsoperationen" im Kapitel "Konstanten" auf Seite 136)*

native rM2M_RegDelKey(reg, const name[]);

sucht einen Eintrag anhand seines Namens und löscht den Eintrag aus dem Registrierungsspeicherblock. Detaillierte Informationen zu den Registrierungsspeicherblöcken finden Sie im Kapitel "Registrierungsspeicherblöcke" auf Seite 36.

Parameter	***Erklärung***
reg	*Index des Registrierungsspeicherblocks (siehe "Indizes der Registrierungsspeicherblöcke" im Kapitel "Konstanten" auf Seite 136)*
name	*Name des Eintrags der aus dem Registrierungsspeicherblock gelöscht werden soll*

	Erklärung
Rückgabewert	• *OK, wenn erfolgreich* • *ERROR, wenn ein unspezifizierter Fehler auftritt* • *< OK, wenn ein anderer Fehler auftritt (siehe "Fehlercodes für die Registrierungsspeicherblock-Zugriffsoperationen" im Kapitel "Konstanten" auf Seite 136)*

native rM2M_RegOnChg(funcidx);

legt die Funktion fest, die aufgerufen werden soll, wenn sich einer der Registrierungsspeicherblöcke geändert hat. (d.h. wurde durch den Server aktualisiert). Der Callback wird nicht durch lokale (geräteseitige) Änderungen eines Registrierungsspeichers herbeigeführt.Detaillierte Informationen zu den Registrierungsspeicherblöcken finden Sie im Kapitel "Registrierungsspeicherblöcke" auf Seite 36.

Parameter	***Erklärung***
funcidx	*Index der öffentlichen Funktion, die aufgerufen werden soll, wenn sich die Registrierung geändert hat* *Typ der Funktion: public func(reg);*

	Erklärung
Rückgabewert	• *OK, wenn erfolgreich* • *< OK, wenn ein Fehler auftritt (siehe "Returncodes für allgemeine Zwecke" im Kapitel "Konstanten" auf Seite 108)*

13.3.6 Position

13.3.6.1 Arrays mit symbolischen Indizes

TrM2M_GSMPos

Informationen über eine GSM/UMTS/LTE-Zelle im Empfangsbereich

```
// mcc        MCC (Mobile Country Code) der GSM-Zelle
// mnc        MNC (Mobile Network Code) der GSM-Zelle
// lac        LAC (Location Area Code) der GSM-Zelle
// cellid     Cell ID der GSM-Zelle
// rssi       empfangene GSM-Feldstärke [dBm] für die GSM-Zelle
// ta         TA (Timing Advance) der GSM-Zelle (dzt. immer auf 0)

#define TrM2M_GSMPos[.mcc, .mnc, .lac, .cellid, .rssi, .ta]
```

TrM2M_PosUpdateGSM

Informationen über eine GSM-Zelle im Empfangsbereich

```
// type       gibt den Typ des Eintrags an (RM2M_POSUPDATE_TYPE_GSM)
// stamp      Zeitpunkt zu dem die Daten ermittelt wurden
// mcc        MCC (Mobile Country Code) der GSM-Zelle
// mnc        MNC (Mobile Network Code) der GSM-Zelle
// lac        LAC (Location Area Code) der GSM-Zelle
// cid        Cell ID der GSM-Zelle
// rssi       empfangene GSM-Feldstärke [dBm] für die GSM-Zelle
// ta         TA (Timing Advance) der GSM-Zelle (dzt. immer auf 0)

#define TrM2M_PosUpdateGSM [.type, .stamp, .mcc, .mnc, .lac, .cid, .rssi, .ta]
```

TrM2M_PosUpdateUMTS

Informationen über eine UMTS-Zelle im Empfangsbereich

```
// type       gibt den Typ des Eintrags an (RM2M_POSUPDATE_TYPE_UMTS)
// stamp      Zeitpunkt zu dem die Daten ermittelt wurden
// mcc        MCC (Mobile Country Code) der GSM-Zelle
// mnc        MNC (Mobile Network Code) der GSM-Zelle
// lac        LAC (Location Area Code) der GSM-Zelle
// cid        Cell ID der GSM-Zelle
// rscp       Received Signal Code Power [dBm]
// pscr       Primary Scrambling Code

#define TrM2M_PosUpdateUMTS [.type, .stamp, .mcc, .mnc, .lac, .cid, .rscp,
                             .pscr]
```

TrM2M_PosUpdateLTE

Informationen über eine LTE-Zelle im Empfangsbereich

```
// type       gibt den Typ des Eintrags an (RM2M_POSUPDATE_TYPE_LTE)
// stamp      Zeitpunkt zu dem die Daten ermittelt wurden
// mcc        MCC (Mobile Country Code) der GSM-Zelle
// mnc        MNC (Mobile Network Code) der GSM-Zelle
// lac        LAC (Location Area Code) der GSM-Zelle
// cid        Cell ID der GSM-Zelle
// rsrp       Reference Signal Received Power [dBm]

#define TrM2M_PosUpdateLTE [.type, .stamp, .mcc, .mnc, .lac, .cid, .rsrp]
```

TrM2M_PosUpdateWiFi

Informationen über ein WiFi-Netzwerk im Empfangsbereich

```
// type       gibt den Typ des Eintrags an (RM2M_POSUPDATE_TYPE_WIFI)
// stamp      Zeitpunkt zu dem die Daten ermittelt wurden
// ch         WiFi RF Kanal (1-14 im 2.4 GHz Frequenzbereich), Datentyp: u8
// rssi       WiFi RSSI Level [dBm], Datentyp: s8
// bssid      Basic Service Set Identification (z.B. MAC-Adresse des
//            Access Point)
// ssid       Service Set Identifier des WiFi-Netzwerks

#define TrM2M_PosUpdateWiFi [.type, .stamp, .ch, .rssi, .bssid{6}, .ssid{32+1}]
```

TNMEA_GGA

Informationen (Position, Höhe über Meeresoberfläche und Genauigkeit), die aus einem GGA Datensatz extrahiert wurden

```
// Lat        geografische Breite in Grad (Auflösung: 0,000001°)
//            -90 000 000 =Südpol 90° S, 0 =Äquator, +90 000 000 =Nordpol 90° N
//
// Long       geografische Länge in Grad (Auflösung: 0,000001°)
//            -180 000 000 =180° West, 0 =Nullmeridian, +180 000 000 =180° Ost
//
// Alt        Höhe über dem Meeresspiegel in Meter
// Qual       NMEA Qualitätsindikator(siehe "Konstanten" auf Seite 145)
// SatUsed    Anzahl der zur Positionsbestimmung verwendeten Satelliten
// HDOP       relative Genauigkeit der horizontalen Position [0,01]

#define TNMEA_GGA[.Lat, .Long, .Alt, .Qual, .SatUsed, .HDOP]
```

13.3.6.2 Konstanten

Liste der unterstützten Typen von Zell/Netzwerk-Informationseinträgen

mögliche Typen der Zell/Netzwerk-Informationseinträge, die mittels der Funktion "rM2M_EnumPosUpdate()" vom System gelesen werden können

```
RM2M_POSUPDATE_TYPE_ERR  = 0, //ungültiger Eintrag
RM2M_POSUPDATE_TYPE_GSM  = 1, //Informationen über eine GSM-Zelle
RM2M_POSUPDATE_TYPE_UMTS = 2, //Informationen über eine UMTS-Zelle
RM2M_POSUPDATE_TYPE_LTE  = 3, //Informationen über eine LTE-Zelle
RM2M_POSUPDATE_TYPE_WIFI = 4, //Informationen über ein WiFi-Netzwerk
```

NMEA Fehlercodes

Fehlercodes der Funktion rM2M_SetPosNMEA()

```
RM2M_NMEA_ERR_DATATYPE  = -2, // Datentyp (z.B. $GGSA) wird nicht unterstützt.
RM2M_NMEA_ERR_SENTENCE  = -3, // Sentence ungültig (z.B. Checksum Fehler)
RM2M_NMEA_ERR_LATITUDE  = -4, // geographische Breite ungültig
RM2M_NMEA_ERR_LONGITUDE = -5, // geographische Länge ungültig
RM2M_NMEA_ERR_ALTITUDE  = -6, // Höhe über dem Meeresspiegel ungültig
RM2M_NMEA_ERR_SAT_USED  = -7, // Anzahl der verwendeten Satelliten ist ungültig.
RM2M_NMEA_ERR_QUAL      = -8, // GPS-Qualitätsangabe wird nicht unterstützt.
```

NMEA Qualitätsindikator

```
RM2M_NMEA_FIX_NOK      = 0,   // invalid/no fix
RM2M_NMEA_FIX_GPS      = 1,   // non-differential GPS fix
RM2M_NMEA_FIX_DGPS     = 2,   // differential GPS fix
RM2M_NMEA_FIX_PPS      = 3,   // Precise Positioning Service (PPS)
RM2M_NMEA_FIX_RTK      = 4,   // Real Time Kinematic (RTK)
RM2M_NMEA_FIX_FLOATRTK = 5,   // Float Real Time Kinematic
RM2M_NMEA_FIX_EST      = 6,   // estimated fix(dead reckoning, Koppelnavigation)
RM2M_NMEA_FIX_MAN      = 7,   // manual input mode
RM2M_NMEA_FIX_SIM      = 8,   // simulation mode
```

Liste der unterstützten GNSS Geräte-ID's

dient dazu die Quelle des NMEA-Datensatzes zu identifizieren (gemäß der bei NMEA 0183 Standard verwendeten "Talker ID")

```
RM2M_NMEA_DEVICE_GP = 0x244750,  // $GP (GPS)
RM2M_NMEA_DEVICE_GL = 0x24474C,  // $GL (GLONASS)
RM2M_NMEA_DEVICE_GA = 0x244741,  // $GA (GALILEO)
RM2M_NMEA_DEVICE_GN = 0x24474E,  // $GN (GENERIC GNSS)
```

Liste der unterstützten NMEA-Datensätze

```
RM2M_NMEA_RECORD_GGA = 0x474741, // GGA (Global Positioning System Fix Data)
```

13.3.6.3 Funktionen

native rM2M_SetPos(Lat, Long, Elev, Qual, SatUsed);

speichert die GPS-Positionsinformationen im Gerät. Es erfolgt keine historische Aufzeichnung. D.h. die aktuellen Positionsinformationen überschreiben immer die letzte bekannte Position. Die Informationen werden zum myDatanet-Server übertragen und können z.B. per API (siehe "API" auf Seite 225) ausgelesen werden.

Parameter	***Erklärung***
Lat	*geografische Breite in Grad (Auflösung: 0,000001°)* *-90 000 000 = Südpol 90° Süd* *0 = Äquator* *+90 000 000 = Nordpol 90° Nord*
Long	*geografische Länge in Grad (Auflösung: 0,000001°)* *-180 000 000 = 180° West* *0 = Nullmeridian (Greenwich)* *+180 000 000 = 180° Ost*
Elev	*Höhe über dem Meeresspiegel in Meter (gültiger Bereich: -999...+9999)*
Qual	*Qualitätsindikator (GPS quality indicator)* *RM2M_NMEA_FIX_NOK: invalid/no fix* *RM2M_NMEA_FIX_GPS: non-differential GPS fix* *RM2M_NMEA_FIX_DGPS: differential GPS fix* *RM2M_NMEA_FIX_PPS: Precise Positioning Service (PPS)* *RM2M_NMEA_FIX_RTK: Real Time Kinematic (RTK)* *RM2M_NMEA_FIX_FLOATRTK: Float Real Time Kinematic* *RM2M_NMEA_FIX_EST: estimated fix (dead reckoning, Koppelnavigation)* *RM2M_NMEA_FIX_MAN: manual input mode* *RM2M_NMEA_FIX_SIM: simulation mode*
SatUsed	*Anzahl der zur Positionsbestimmung verwendeten Satelliten (gültiger Bereich: 0 ... 99)*

	Erklärung
Rückgabewert	• *OK, wenn erfolgreich* • *ERROR* ***Hinweis:** Die Parameter werden auf die angegebenen Bereichsgrenzen geprüft. Bei Verletzung der Grenzen ist der Rückgabewert der Funktion "ERROR".*

native rM2M_DecodeNMEA(const sentence{}, data[], len=sizeof data);

decodiert einen übergebenen NMEA Datensatz

Parameter	***Erklärung***
sentence	*NMEA-Datensatz vom einem GPS-Empfänger, beginnend mit dem Zeichen '$'.* ***Wichtiger Hinweis:** Die Terminierung des Strings ('\0') muss unmittelbar hinter der Checksumme erfolgen.*
data	*Puffer (Cell-Array) zur Aufnahme der decodierten Daten* *[0] : enthält die GNSS Geräte-ID (siehe "Liste der unterstützten GNSS Geräte-ID's" im Kapitel "Konstanten" auf Seite 145)* *[1] : enthält den Typ des decodierten NMEA-Datensatzes (siehe "Liste der unterstützten NMEA-Datensätze" im Kapitel "Konstanten" auf Seite 145)* *[2] ... [n] : abhängig vom Typ des decodierten NMEA-Datensatzes* *Für einen Datensatz vom Typ "RM2M_NMEA_RECORD_GGA" beispielsweise entspricht der weitere Aufbau der Struktur von "TNMEA_GGA".* *[2]: .Lat* *[3]: .Long* *[4]: .Alt* *[5]: .Qual* *[6]: .SatUsed* *[7]: .HDOP*
len	*Größe (in Cells) des Puffers zur Aufnahme der decodierten Daten - OPTIONAL*

	Erklärung
Rückgabewert	• *positiver Wert, wenn erfolgreich (Anzahl der befüllten Arrayelemente, d.h. Cells)* • *negativer Wert, wenn ein Fehler aufgetreten ist (siehe "NMEA Fehlercodes" im Kapitel "Konstanten" auf Seite 145)*

native rM2M_SetPosNMEA(const Sentence{});

entnimmt die GPS-Positionsinformationen aus dem übergebenen NMEA-Datensatz und speichert sie im Gerät. Es erfolgt keine historische Aufzeichnung. D.h. die aktuellen Positionsinformationen überschreiben immer die letzte bekannte Position. Die Informationen werden zum myDatanet-Server übertragen und können z.B. per API (siehe "API" auf Seite 225) ausgelesen werden.

Parameter	***Erklärung***
Sentence	*NMEA-Datensatz vom einem GPS-Empfänger, beginnend mit dem Zeichen '$'. Derzeit werden folgende Datensätze unterstützt:* • *$GPGGA - Standortbestimmung (fix information)* ***Wichtiger Hinweis:*** *Die Terminierung des Strings ('\0') muss unmittelbar hinter der Checksumme erfolgen.*

	Erklärung
Rückgabewert	• *OK, wenn erfolgreich* • *< OK, wenn ein Fehler auftritt (siehe "NMEA Fehlercodes" im Kapitel "Konstanten" auf Seite 145)*

native rM2M_GetPos(&Lat, &Long, &Elev, &Qual=0, &SatUsed=0);

liest die im Gerät gespeicherten GPS-Positionsinformationen aus

Parameter	*Erklärung*
Lat	*Variable zur Aufnahme der geographischen Breite in Grad (Auflösung: 0,000001°)* *-90 000 000 = Südpol 90° Süd* *0 = Äquator* *+90 000 000 = Nordpol 90° Nord*
Long	*Variable zur Aufnahme der geographischen Länge in Grad (Auflösung: 0,000001°)* *-180 000 000 = 180° West* *0 = Nullmeridian (Greenwich)* *+180 000 000 = 180° Ost*
Elev	*Variable zur Aufnahme der Höhe über dem Meeresspiegel in Meter (gültiger Bereich: -999...+9999)*
Qual	*Variable zur Aufnahme des Qualitätsindikators (GPS quality indicator) - OPTIONAL* *RM2M_NMEA_FIX_NOK: invalid/no fix* *RM2M_NMEA_FIX_GPS: non-differential GPS fix* *RM2M_NMEA_FIX_DGPS: differential GPS fix* *RM2M_NMEA_FIX_PPS: Precise Positioning Service (PPS)* *RM2M_NMEA_FIX_RTK: Real Time Kinematic (RTK)* *RM2M_NMEA_FIX_FLOATRTK: Float Real Time Kinematic* *RM2M_NMEA_FIX_EST: estimated fix (dead reckoning, Koppelnavigation)* *RM2M_NMEA_FIX_MAN: manual input mode* *RM2M_NMEA_FIX_SIM: simulation mode*
SatUsed	*Variable zur Aufnahme der Anzahl der zur Positionsbestimmung verwendeten Satelliten - OPTIONAL*

	Erklärung
Rückgabewert	• *OK, wenn gültige GPS-Positionsinformationen im Gerät gespeichert sind* • *< OK, wenn ein Fehler auftritt (siehe "Returncodes für allgemeine Zwecke" im Kapitel "Konstanten" auf Seite 108)*

native rM2M_PosUpdate(...)

ermöglicht es, Informationen über die GSM/UMTS/LTE-Zellen und WiFi-Netzwerke ans Gerät zu übergeben. Die Positionsdaten werden somit auf Basis dieser vom Benutzer bereitgestellten Informationen erzeugt. Bei dieser Funktion wird eine variable Liste an Parametern verwendet. Die zu übergebenden Parameter sind vom Verwendungszweck abhängig. Folgender Ablauf wird empfohlen:

1. *Vorbereiten eines neuen Positionsdatensatzes*

   ```
   rM2M_PosUpdate(RM2M_POSUPDATE_TYPE_WIFI);          // e.g. WiFi
   ```

   Die intern gespeicherte Liste der Zell/Netzwerk-Informationen wird verworfen und eine neue wird vorbereitet. Der Zeitstempel dieser neuen Liste wird auf die aktuelle Zeit gesetzt.

2. *Übergeben der Zell/Netzwerk-Informationseinträge ans System*

   ```
   rM2M_PosUpdate(RM2M_POSUPDATE_TYPE_WIFI, sData[TrM2M_PosUpdateWiFi],
                  sizeof sData);                      // e.g. WiFi
   ```

   Der übergebene Zell/Netzwerk-Informationseintrag wird in die Liste eingetragen, die im Schritt 1 neu erstellte wurde. Die in der Struktur "TrM2M_PosUpdatexxx" enthaltenen Einträge ".type" und ".stamp" werden ignoriert. Es können maximal 20 Zell/Netzwerk-Informationseinträge in die Liste eingetragen werden.

   > ***Hinweis:*** *Für "type" muss der zur im Datenpuffer "sData" übergebenen "TrM2M_PosUpdatexxx" Struktur passende Typ der Zell/Netzwerk-Informationen angegeben werden.*

3. *Abschließen des aktuellen Positionsdatensatzes durch das Übergeben eines "Dummy" Zell/Netzwerk-Informationseintrags mit der länge 0*

   ```
   rM2M_PosUpdate(RM2M_POSUPDATE_TYPE_WIFI, [0], 0);  // e.g. WiFi
   ```

   Die Liste der Zell/Netzwerk-Informationseinträge wird sortiert (absteigend nach Empfangspegel) und der Zeitstempel wird aktualisiert. Der Zeitstempel wird für die Synchronisation der Positionsdaten mit dem Server verwendet. Wenn das Gerät gerade im online-Modus ist, werden die Positionsdaten unmittelbar an den Server übermittelt. Wenn die Zell/Netzwerk-Informationseinträge mittels der Funktion "rM2M_EnumPosUpdate()" wieder ausgelesen werden, wird dieser im Schritt 3 erstellte Zeitstempel in die "TrM2M_PosUpdatexxx" Struktur eingetragen.

Parameter	***Erklärung***
type	*Typ der Zell/Netzwerk-Informationen die ans System übergeben werden (siehe "RM2M_POSUPDATE_TYPE_xxx" im Kapitel "Konstanten" auf Seite 145)*
sData	*Zell/Netzwerk-Informationseintrag, der ans System übergeben werden soll. Die Datenstruktur hängt vom ausgewählten Typ (Parameter "type") ab.* *z.B. "TrM2M_PosUpdateWiFi" (siehe TrM2M_PosUpdateWiFi im Kapitel "Arrays mit symbolischen Indizes" auf Seite 143) für Typ "RM2M_POSUPDATE_TYPE_WIFI".*
len	• *Größe (in Cells) des Zell/Netzwerk-Informationseintrags, der in den Datenpuffer "sData" kopiert wurde* • *Null, um anzuzeigen, dass keine weiteren Zell/Netzwerk-Informationseinträge an das System übergeben werden, die für den aktuellen Positionsdatensatz berücksichtigt werden sollen. -->*

	Erklärung
Rückgabewert	• *OK, wenn erfolgreich* • *ERROR_NOT_SUPPORTED* • *ERROR-1, wenn vorübergehend nicht möglich (z. B. internes Positionsupdate aktiv),* • *< OK, wenn ein anderer Fehler auftritt (siehe "Returncodes für allgemeine Zwecke" im Kapitel "Konstanten" auf Seite 108)*

native rM2M_EnumPosUpdate(...);

listet die im Gerät gespeicherten Informationen über die GSM/UMTS/LTE-Zellen und WiFi-Netzwerke im Empfangsbereich auf. Bei dieser Funktion wird eine variable Liste an Parametern verwendet. Die zu übergebenden Parameter sind vom Verwendungszweck abhängig. Folgender Ablauf wird empfohlen:

1. *Auslesen der Anzahl von verfügbaren Zell/Netzwerk-Informationseinträgen*

```
new nEnum;

rM2M_EnumPosUpdate(nEnum);
```

2. *Ermittlung des jeweiligen Typs der Zell/Netzwerk-Informationseinträge*

```
new type;
new idxEnum = 0;

for(idxEnum=0 ; idxEnum < nEnum ; idxEnum++)
   rM2M_EnumPosUpdate(idxEnum, type);
```

3. *Auslesen der Zell/Netzwerk-Informationseinträge basierend auf den zuvor ermittelten Typen (im folgenden Beispiel nur jene, die Informationen über eine GSM-Zelle enthalten).*

```
new sGSMPos[TrM2M_PosUpdateGSM];

if(type == RM2M_POSUPDATE_TYPE_GSM)
   rM2M_EnumPosUpdate(idxEnum, sGSMPos, sizeof sGSMPos);
```

Parameter	***Erklärung***
nEnum	*Variable zur Aufnahme der Anzahl der verfügbaren Zell/Netzwerk-Informationseinträgen*
idxEnum	*Index des Zell/Netzwerk-Informationseintrags dessen Typ ermittelt werden soll oder der vom System gelesen werden soll.* *Abhängig von der gewünschten Aktion sind zusätzlich entweder der Parameter "type" oder die beiden Parameter "buf" und "len" erforderlich.*
type	*Variable zur Aufnahme des Typs eines Zell/Netzwerk-Informationseintrags (siehe "RM2M_POSUPDATE_TYPE_xxx" im Kapitel "Konstanten" auf Seite 145)*
buf	*Puffer zur Aufnahme eines Zell/Netzwerk-Informationseintrags* *Die Struktur des Puffers ist abhängig vom zu lesenden Zell/Netzwerk-Informationseintrag (siehe "TrM2M_PosUpdatexxx" im Kapitel "Arrays mit symbolischen Indizes" auf Seite 143)*
len	*Größe (in Cells) der Struktur zur Aufnahme eines Zell/Netzwerk-Informationseintrags*

	Erklärung
Rückgabewert	• *OK, wenn erfolgreich* • *ERROR, wenn ein Fehler auftritt*

native rM2M_GetGSMPos(posidx, pos[TrM2M_GSMPos]=0);

liefert die Anzahl der GSM/UMTS/LTE-Zellen, für die gültige Informationen im Gerät gespeichert sind (posidx < 0) bzw. liest die im Gerät gespeicherten Informationen über eine GSM/UMTS/LTE-Zelle im Empfangsbereich aus (posidx >= 0)

> ***Hinweis:*** *Nutzen Sie die Funktion "rM2M_EnumPosUpdate()" um Informationen zu WiFi-Netzwerken im Empfangsbereich oder spezifischere Informationen zu UMTS- bzw. LTE-Zellen zu erhalten.*

Parameter	***Erklärung***
posidx	*Auswahl der von der Funktion gelieferten Information* *posidx < 0:* *Anzahl der GSM/UMTS/LTE-Zellen, für die gültige Informationen im Gerät gespeichert sind, auslesen* *posidx >=0:* *Nummer des GSM/UMTS/LTE-Zellen-Informationsblocks, der ausgelesen werden soll*
pos	*posidx < 0:* *nicht erforderlich* *posidx >=0:* *Struktur zur Aufnahme der Informationen über eine GSM/UMTS/LTE-Zelle im Empfangsbereich (siehe "TrM2M_GSMPos" im Kapitel "Arrays mit symbolischen Indizes" auf Seite 143)*

	Erklärung
Rückgabewert	*posidx < 0:* *Anzahl der GSM/UMTS/LTE-Zellen, für die gültige Informationen im Gerät gespeichert sind (max. 10)* *posidx >=0:* • *OK, wenn der gewünschte Zellen-Informationsblock gültige Daten einer GSM-Zelle enthält* • *OK+1, wenn der gewünschte Zellen-Informationsblock gültige Daten einer UMTS-Zelle enthält* • *OK+2, wenn der gewünschte Zellen-Informationsblock gültige Daten einer LTE-Zelle enthält* • *ERROR*

13.3.7 Mathematik

Hilfreiche Konstanten

Definintition	Wert	Beschreibung
M_E	2.7182818284590452354	e
M_LOG2E	1.4426950408889634074	$\log_2 e$
M_LOG10E	0.43429448190325182765	$\log_{10} e$
M_LN2	0.69314718055994530942	ln 2
M_LN10	2.30258509299404568402	ln 10
M_PI	3.14159265358979323846	π
M_PI_2	1.57079632679489661923	π/2
M_PI_4	0.78539816339744830962	π/4
M_1_PI	0.31830988618379067154	1/π
M_2_PI	0.63661977236758134308	2/π
M_2_SQRTPI	1.12837916709551257390	2/sqrt(π)
M_SQRT2	1.41421356237309504880	sqrt(2)
M_SQRT1_2	0.70710678118654752440	1/sqrt(2)

native fround(Float:x);

führt kaufmännisches Runden des übergebenen Floats durch

Parameter	*Erklärung*
x	*Float, der gerundet werden soll*

	Erklärung
Rückgabewert	*kaufmännisch gerundeter ganzzahliger Wert*

native min(value1, value2);

liefert den kleineren der beiden übergebenen Werte

Parameter	*Erklärung*
value1	*zwei Werte, von denen der kleinere ermittelt werden soll*
value2	

	Erklärung
Rückgabewert	*der kleinere der beiden übergebenen Werte*

native max(value1, value2);
liefert den größeren der beiden übergebenen Werte

Parameter	*Erklärung*
value1	*zwei Werte, von denen der größere ermittelt werden soll*
value1	

	Erklärung
Rückgabewert	*der größere der beiden übergebenen Werte*

native clamp(value, min=cellmin, max=cellmax);
prüft, ob der übergebene Wert zwischen "min" und "max" liegt

Parameter	*Erklärung*
value	*Wert, der geprüft werden soll*
min	*untere Bereichsgrenze*
max	*obere Bereichsgrenze*

	Erklärung
Rückgabewert	• *"value", wenn der Wert zwischen "min" und "max" liegt* • *"min", wenn der Wert kleiner "min" ist* • *"max", wenn der Wert größer "max" ist*

native swapchars(c);
vertauscht die Reihenfolge der Bytes

Parameter	*Erklärung*
c	*Wert für den die Bytes vertauscht werden sollen*

	Erklärung
Rückgabewert	*Wert, bei dem die Bytes im Parameter "c" vertauscht sind (das niedrigste Byte wird das höchste Byte)*

Die Arbeitsweise der folgenden Funktionen entspricht jener der Standard ANSI-C Implementierung:

native Float:sin(Float:x);
Sinus von x

native Float:cos(Float:x);
Kosinus von x

native Float:tan(Float:x);
Tangens von x

native Float:asin(Float:x);
arcsin(x) im Bereich [-π/2, π/2], x Element von [-1, 1]

native Float:acos(Float:x);
arccos(x) im Bereich [0, π], x Element von [-1, 1]

native Float:atan(Float:x);
arctan(x) im Bereich [-π/2, π/2]

native Float:atan2(Float:y, Float:x);
arctan(y/x) im Bereich [-π, π]

native Float:sinh(Float:x);
Sinus Hyperbolicus von x

native Float:cosh(Float:x);
Cosinus Hyperbolicus von x

native Float:tanh(Float:x);
Tangens Hyperbolicus von x

native Float:exp(Float:x);
Exponentialfunktion e^x

native Float:log(Float:x);
natürlicher Logarithmus ln(x), x > 0

native Float:log10(Float:x);
Logarithmus zur Basis 10 $\log_{10}(x)$, x > 0

native Float:pow(Float:x, Float:y);
x^y. Ein Argumentenfehler liegt vor bei x = 0 und y <= 0, oder bei x < 0 und y ist nicht ganzzahlig.

native Float:sqrt(Float:x);
Quadratwurzel x, x >= 0

native Float:ceil(Float:x);
kleinster ganzzahliger Wert, der nicht kleiner als x ist

native Float:floor(Float:x);
größter ganzzahliger Wert, der nicht größer als x ist

native Float:fabs(Float:x);
absoluter Wert | x |

native Float:ldexp(Float:x, n);
$x*2^n$

native Float:frexp(Float:x, &n);
zerlegt x in eine normalisierte Mantisse im Bereich [1/2, 1], die als Resultat geliefert wird, und eine Potenz von 2, die in n abgelegt wird. Ist x null, sind beide Teile des Resultats null.

native Float:modf(Float:x, &Float:ip);
zerlegt x in einen ganzzahligen Teil und einen Rest, die beide das gleiche Vorzeichen wie x besitzen. Der ganzzahlige Teil wird bei ip abgelegt, der Rest ist das Resultat.

native Float:fmod(Float:x, Float:y);
Gleitpunktrest von x/y, mit dem gleichen Vorzeichen wie x. Wenn y null ist, hängt das Resultat von der Implementierung ab.

native isnan(Float:x);
liefert einen Wert ungleich Null, wenn x "not a number" ist

13.3.8 64-Bit Signed Integer

native s64_set(val_s64{8}, val);

überträgt den Inhalt einer Standardvariable der Scriptsprache (32-Bit signed Integer) in ein Array, welches der Aufnahme eines 64-Bit signed Integer Werts dient.

Parameter	*Erklärung*
val_s64	*Array zur Aufnahme eines 64-Bit signed Integer Werts*
val	*Variable, deren Inhalt in einen 64-Bit signed Integer Wert übertragen werden soll*

	Erklärung
Rückgabewert	• *OK, wenn erfolgreich* • *ERROR, wenn ein ungültiger Parameter übergeben wurde*

native s64_get(val_s64{8}, &val);

überträgt einen 64-Bit signed Integer Wert in eine Standardvariable der Scriptsprache

> ***Hinweis:*** *Die Standardvariablen der Scriptsprache können nur 32-Bit signed Integer Werte aufnehmen. Wenn der im Array gespeicherte Wert den 32-Bit signed Integer Wertebereich überschreitet, ist der in "val" übertragene Wert ungültig, da die oberen 4 Bytes abgeschnitten werden. Es empfiehlt sich, vor dem Auslesen mittels der Funktion "s64_cmp()" zu überprüfen, ob der 32-Bit signed Integer Wertebereich überschritten wurde.*

Parameter	*Erklärung*
val_s64	*Array, das den 64-Bit signed Integer Wert enthält*
val	*Variable zur Aufnahme des zu lesenden Werts*

	Erklärung
Rückgabewert	• *OK, wenn erfolgreich* • *ERROR, wenn ein ungültiger Parameter übergeben wurde*

native s64_add(term1{8}, term2{8}, sum{8});

addiert zwei 64-Bit signed Integer Werte (sum = term1 + term2)

Parameter	*Erklärung*
term1	*Arrays, die die zwei zu addierenden Werte enthalten*
term2	
sum	*Array zur Aufnahme des Ergebnisses der Addition*

	Erklärung
Rückgabewert	• *OK, wenn erfolgreich* • *ERROR, wenn ein ungültiger Parameter übergeben wurde*

native s64_sub(minuend{8}, subtrahend{8}, difference{8});

subtrahiert einen 64-Bit signed Integer Wert von einem anderen (difference = minuend - subtrahend)

Parameter	*Erklärung*
minuend	*Array, das den Wert enthält, von dem subtrahiert werden soll*
subtrahend	*Array, das den Wert enthält, der subtrahiert werden soll*
difference	*Array zur Aufnahme des Ergebnisses der Subtraktion*

	Erklärung
Rückgabewert	• *OK, wenn erfolgreich* • *ERROR, wenn ein ungültiger Parameter übergeben wurde*

native s64_mul(multiplier{8}, multiplicand{8}, product{8});

*multipliziert zwei 64-Bit signed Integer Werte miteinander (product = multiplier * multiplicand)*

Parameter	*Erklärung*
multiplier	*Arrays, die die zwei miteinander zu mutiplizierenden Werte enthalten*
multiplicand	
product	*Array zur Aufnahme des Ergebnisses der Multiplikation*

	Erklärung
Rückgabewert	• *OK, wenn erfolgreich* • *ERROR, wenn ein ungültiger Parameter übergeben wurde*

native s64_div(dividend{8}, divisor{8}, quotient{8});

dividiert einen 64-Bit signed Integer Wert durch einen anderen (quotient = dividend / divisor)

Parameter	*Erklärung*
dividend	*Array, das den Wert enthält, der geteilt werden soll*
divisor	*Array, das den Wert enthält, durch den dividiert werden soll*
quotient	*Array zur Aufnahme des Ergebnisses der Division*

	Erklärung
Rückgabewert	• *OK, wenn erfolgreich* • *ERROR, wenn ein ungültiger Parameter übergeben wurde*

native s64_mod(dividend{8}, divisor{8}, reminder{8});

ermittelt den Rest der Division von 64-Bit signed Integer Werten (reminder = dividend % divisor)

Parameter	*Erklärung*
dividend	*Array, das den Wert enthält, der geteilt werden soll*
divisor	*Array, das den Wert enthält, durch den dividiert werden soll*
reminder	*Array zur Aufnahme des bei der ganzzahligen Division auftretenden Rests*

	Erklärung
Rückgabewert	• *OK, wenn erfolgreich* • *ERROR, wenn ein ungültiger Parameter übergeben wurde*

native s64_lshift(val{8}, bits);

verschiebt den im Array enthaltenen 64-Bit signed Integer Wert um "bits" Bits nach links (val <<= bits). Die frei werden Bits werden mit 0 aufgefüllt. Es erfolgt eine arithmetische Verschiebung, d.h. das Vorzeichen wird beibehalten.

Parameter	*Erklärung*
val	*Array, das den Wert enthält*
bits	*Anzahl der Bits, um die der Wert nach links verschoben werden soll*

	Erklärung
Rückgabewert	• *OK, wenn erfolgreich* • *ERROR, wenn ein ungültiger Parameter übergeben wurde*

native s64_rshift(val{8}, bits);

verschiebt den im Array enthaltenen 64-Bit signed Integer Wert um "bits" Bits nach rechts (val >>= bits'). Die frei werden Bits werden mit 0 aufgefüllt. Es erfolgt eine arithmetischen Verschiebung, d.h. das Vorzeichen wird beibehalten

Parameter	*Erklärung*
val	*Array, das den Wert enthält*
bits	*Anzahl der Bits, um die der Wert nach rechts verschoben werden soll*

	Erklärung
Rückgabewert	• *OK, wenn erfolgreich* • *ERROR, wenn ein ungültiger Parameter übergeben wurde*

native s64_cmp(val1{8}, val2{8});

vergleicht die 64-Bit signed Integer Werte val1 und val2 miteinander

Parameter	*Erklärung*
val1	*Arrays, die die beiden zu vergleichenden Werte enthalten*
val2	

	Erklärung
Rückgabewert	• *1: val1 > val2* • *0: die beiden Werte sind gleich* • *-1: val1 < val2*

13.3.9 Char & String

Die Arbeitsweise der folgenden Funktionen entspricht im Wesentlichen jener der Standard ANSI-C Implementierung:

native strlen(const string[]);

liefert die Länge von string (ohne '\0')

Parameter	*Erklärung*
string	*Zeichenkette, deren Länge bestimmt werden soll*

	Erklärung
Rückgabewert	*Anzahl der Zeichen ohne der abschließenden '\0'*

native sprintf(dest[], maxlength=sizeof dest, const format[], {Float,Fixed,_}:...);

speichert den übergebenen Format-String in dem Array dest. Die Arbeitsweise der Funktionen entspricht der Funktion "snprintf" der Standard ANSI-C Implementierung.

Hinweis:

- *Wenn der erzeugte String länger als die Größe von <dest> ist, wird das letzte Zeichen als abschließende 0 gesetzt.*
- *Die Größe von <dest> wird immer auf ein Vielfaches von 4 aufgerundet.*

Parameter	***Erklärung***
dest	*Array zur Aufnahme des formatierten Ergebnisses*
maxlength	*maximale Zeichenanzahl, die das dest Array aufnehmen kann*
format	*die zu verwendende Format-Zeichenkette (C-style Formatierungscodes)* *%b : Zahl im Binärradix* *%c : Zeichen* *%d : Zahl im Dezimalradix* *%f : Fließkommazahl* *%s : String* *%x : Zahl im Hexadezimalradix* *... : s32 | f32 | astr - Zusätzliche Argumente.* *Abhängig von der Formatzeichenkette kann die Funktion eine Sequenz von zusätzlichen Argumenten erwarten, die jeweils einen Wert zum Ersetzen eines Formatspezifikators in der Formatzeichenkette enthalten.*

	Erklärung
Rückgabewert	• *-1 im Fehlerfall* • *Anzahl der Zeichen, die geschrieben worden wäre, wenn das Array dest lang genug gewesen wäre (ohne '\0').* *Das Array dest erhält in jedem Fall ein abschließendes Nullzeichen. In keinem Fall wird über die Länge des Arrays dest hinausgeschrieben.*

native strcpy(dest[], const source[], maxlength=sizeof dest);

kopiert die Zeichenkette source in das Array dest (inklusive '\0').

Parameter	***Erklärung***
dest	*Array zur Aufnahme der zu kopierenden Zeichenkette*
source	*zu kopierende Zeichenkette*
maxlength	*Größe (in Cells) des Arrays zur Aufnahme der zu kopierenden Zeichenkette - OPTIONAL*

	Erklärung
Rückgabewert	*Anzahl der kopierten Zeichen*

native strcat(dest[], const source[], maxlength=sizeof dest);

fügt die Zeichenkette source an die Zeichenkette dest an (inklusive '\0')

Wichtiger Hinweis: *An beide Zeichenketten muss die abschließende Null angefügt werden.*

Parameter	***Erklärung***
dest	*Array zur Aufnahme des Ergebnisses. Dieses Array enthält bereits eine Zeichenkette an die die Zeichenkette source angefügt werden soll.*
source	*Zeichenkette, die an die im Array dest enthaltene Zeichenkette angefügt werden soll*
maxlength	*Größe (in Cells) des Arrays zur Aufnahme des Ergebnisses - OPTIONAL*

	Erklärung
Rückgabewert	*Anzahl der angefügten Zeichen*

native strcmp(const string1[], const string2[], length=cellmax);

vergleicht die Zeichenketten string1 und string2

Wichtiger Hinweis: *An beide Zeichenketten muss die abschließende Null angefügt werden.*

Parameter	***Erklärung***
string1	*die beiden Zeichenketten, die verglichen werden sollen*
string2	
length	*Die maximale Anzahl von Zeichen, die beim Vergleich berücksichtigt werden sollen - OPTIONAL*

	Erklärung
Rückgabewert	• *1: string1 > string 2* • *0: die beiden Zeichenketten sind gleich (zumindest die berücksichtigte Länge)* • *-1: string1 < string 2*

native strchr(const string[], char);

sucht ein Zeichen (erstes Vorkommen) in einer Zeichenkette

Parameter	***Erklärung***
string	*Zeichenkette, die durchsucht werden soll*
char	*Zeichen, das gesucht werden soll*

	Erklärung
Rückgabewert	• *-1, wenn das gesuchte Zeichen nicht in der Zeichenkette enthalten ist* • *Array-Index des gesuchten Zeichens (erstes in der Zeichenkette vorkommendes Zeichen)*

native strrchr(const string[], char);

sucht ein Zeichen (letztes vorkommen) in einer Zeichenkette

Parameter	*Erklärung*
string	*Zeichenkette, die durchsucht werden soll*
char	*Zeichen, das gesucht werden soll*

	Erklärung
Rückgabewert	• *-1, wenn das gesuchte Zeichen nicht in der Zeichenkette enthalten ist* • *Array-Index des gesuchten Zeichens (letztes in der Zeichenkette vorkommendes Zeichen)*

native strspn(const string1[], const string2[]);

*sucht die Position des ersten Zeichens in string1, das **nicht** in der Zeichenkette erlaubter Zeichen (string2) enthalten ist*

Parameter	*Erklärung*
string1	*Zeichenkette, die durchsucht werden soll*
string2	*Zeichenkette erlaubter Zeichen*

	Erklärung
Rückgabewert	• *Länge von string1, wenn keine unerlaubten Zeichen gefunden wurden* • *Position des ersten Zeichens in der zu durchsuchenden Zeichenkette, das nicht in der Zeichenkette der erlaubten Zeichen enthalten ist*

native strcspn(const string1[], const string2[]);

sucht die Position des ersten Zeichens in string1, das auch in der Zeichenkette erlaubter Zeichen (string2) enthalten ist

***Hinweis:** Siehe ähnliche Funktion strpbrk (), die ein leicht abweichendes Ergebnis hat.*

Parameter	*Erklärung*
string1	*Zeichenkette, die durchsucht werden soll*
string2	*Zeichenkette erlaubter Zeichen*

	Erklärung
Rückgabewert	• *Länge von string1, wenn kein erlaubtes Zeichen gefunden wurde* • *Position des ersten Zeichens in der zu durchsuchenden Zeichenkette, das auch in der Zeichenkette der erlaubten Zeichen enthalten ist*

native strpbrk(const string1[], const string2[]);

sucht den Array-Index des ersten Zeichens, das auch in der Zeichenkette erlaubter Zeichen enthalten ist

Hinweis: *Siehe ähnliche Funktion strcspn (), die ein leicht abweichendes Ergebnis hat.*

Parameter	***Erklärung***
string1	*Zeichenkette, die durchsucht werden soll*
string2	*Zeichenkette erlaubter Zeichen*

	Erklärung
Rückgabewert	• *-1: wenn kein erlaubtes Zeichen gefunden wurde* • *>=0: Array-Index des ersten Zeichens in der zu durchsuchenden Zeichenkette, das auch in der Zeichenkette der erlaubten Zeichen enthalten ist*

native strstr(const string1[], const string2[]);

sucht die Zeichenkette string2 in der Zeichenkette string1

Parameter	***Erklärung***
string1	*Zeichenkette, die durchsucht werden soll*
string2	*zu suchende Zeichenkette*

	Erklärung
Rückgabewert	• *-1: wenn die zu suchende Zeichenkette string2 nicht in string1 enthalten ist* • *>=0: Array-Index an der die zu suchende Zeichenkette string2 im string1 beginnt*

native strtol(const string[], base);

wandelt eine Zeichenkette in einen Wert um

Hinweis:

- *Funktion weicht leicht von seiner C Variante ab.*
- *Bis zum ersten Zeichen, das nicht mit der angegebenen Basis übereinstimmt, werden beim Parsen so viele Zeichen wie möglich verarbeitet.*

Parameter	***Erklärung***
string	*umzuwandelnde Zeichenkette* ***Wichtiger Hinweis:*** *Strings > 128 Bytes werden nicht unterstützt!*
base	*gibt die Basis an, die für die Umwandlung verwendet werden soll* *2-36: Die angegebene Basis wird verwendet* *0: Als Basis wird 8, 10 oder 16 verwendet, abhängig von der umzuwandelnden Zeichenkette* *Basis 8: bei einer führenden 0* *Basis 16: bei 0x oder 0X* *Basis 10: Standardeinstellung*

	Erklärung
Rückgabewert	*Wert, der der Zeichenkette entspricht*

native Float: atof(const string[]);

wandelt eine Zeichenkette in einen Float um

Hinweis:

- *Dezimaltrennzeichen ist immer ".", es werden keine Tausendertrennzeichen unterstützt.*
- *Bis zum ersten Zeichen, das nicht in einem Float vorkommt, werden beim Parsen so viele Zeichen wie möglich verarbeitet.*

Parameter	***Erklärung***
string	*umzuwandelnde Zeichenkette* ***Wichtiger Hinweis:*** *Strings > 128 Bytes werden nicht unterstützt!*

	Erklärung
Rückgabewert	*Float, dessen Zahlenwert der Zeichenkette entspricht*

native memcpy_native(dst{}, const dstofs, const src{}, const srcofs, const bytes, const dst_cells=sizeof dst, const src_cells=sizeof src);

kopiert Bytes von einem Puffer in einen anderen

Parameter	***Erklärung***
dst	*Ziel-Puffer, in den die Daten kopiert werden sollen*
dstofs	*Position (Byteoffset) im Ziel-Puffer, an die die Daten kopiert werden sollen*
src	*Quell-Puffer, aus dem die Daten kopiert werden sollen*
srcofs	*Position (Byteoffset) innerhalb des Quell-Puffers, ab der die Daten kopiert werden sollen*
bytes	*Anzahl der Bytes, die kopiert werden sollen*
dst_cells	*Größe (in Cells) des Ziel-Puffers - OPTIONAL*
src_cells	*Größe (in Cells) des Quell-Puffers- OPTIONAL*

	Erklärung

Rückgabewert	• *OK, wenn erfolgreich*
	• *ERROR, wenn einer der folgenden Fehler auftritt*
	  • *Wenn einer der beiden Byteoffsets oder die Anzahl der zu kopierenden Bytes < 0 ist*
	  • *Wenn die Byteoffsets auf ein Byte außerhalb des jeweiligen Puffers verweisen*
	  • *Wenn über den Quell-Puffer hinaus gelesen werden würde (d.h. "Quell-Byteoffset" + "Anz. der Bytes" auf ein Byte außerhalb des Quell-Puffers verweisen würde)*
	  • *Wenn über den Ziel-Puffer hinaus geschrieben werden würde (d.h. "Ziel-Byteoffset" + "Anz. der Bytes" auf ein Byte außerhalb des Ziel-Puffers verweisen würde)*
	  • *Wenn ungültige Puffer übergeben wurden*

native memset_native(dst{}, const dstofs, const srcval, const bytes, dstcells=sizeof dst);

schreibt den gewünschten Wert in die einzelnen Bytes des übergebenen Puffers

Parameter	*Erklärung*
dst	*Puffer, in dem die Bytes auf den gewünschten Wert gesetzt werden sollen*
dstofs	*Position (Byteoffset) innerhalb des übergebenen Puffers, ab der die Bytes auf den gewünschten Wert gesetzt werden sollen*
srcval	*Wert, auf den die einzelnen Bytes gesetzt werden sollen*
bytes	*Anzahl der Bytes, die auf den gewünschten Wert gesetzt werden sollen*
dstcells	*Größe (in Cells) des übergebenen Puffers - OPTIONAL*

	Erklärung

Rückgabewert	• *OK, wenn erfolgreich*
	• *ERROR, wenn einer der folgenden Fehler auftritt*
	  • *Der Byteoffset < 0 ist*
	  • *Der Byteoffset auf ein Byte außerhalb des Puffers verweist*
	  • *Ein ungültiger Puffer übergeben wurden*

native memcmp_native(const src1{}, const src1ofs, const src2{}, const src2ofs, bytes, src1cells=sizeof src1, src2cells=sizeof src2);

vergleicht zwei Puffer Byte für Byte

Parameter	*Erklärung*
src1	*Puffer #1*
src1ofs	*Position (Byteoffset) innerhalb des Puffers #1, ab der die Bytes verglichen werden sollen*
src2	*Puffer #2*
src2ofs	*Position (Byteoffset) innerhalb des Puffers #2, ab der die Bytes verglichen werden sollen*
bytes	*Anzahl der Bytes, die verglichen werden sollen*
src1cells	*Größe (in Cells) des Puffer #1 - OPTIONAL*
src2cells	*Größe (in Cells) des Puffer #2 - OPTIONAL*

	Erklärung
Rückgabewert	• *> 0: Puffer #1 > Puffer #2* • *0: die Inhalte der beiden Puffer sind gleich (zumindest die berücksichtigten Bytes)* • *<0: Puffer #1 < Puffer #2*

native tolower(c);

wandelt ein Zeichen in einen Kleinbuchstaben um

Parameter	*Erklärung*
c	*Zeichen, das in einen Kleinbuchstaben umgewandelt werden soll*

	Erklärung
Rückgabewert	*Die Kleinbuchstaben-Variante des übergebenen Zeichens, falls vorhanden, oder der unveränderte Zeichencode von "c", wenn der Buchstabe "c" kein Kleinbuchstaben-Äquivalent hat.*

native toupper(c);

wandelt ein Zeichen in einen Großbuchstaben um

Parameter	*Erklärung*
c	*Zeichen, das in einen Großbuchstaben umgewandelt werden soll*

	Erklärung
Rückgabewert	*Die Großbuchstaben-Variante des übergebenen Zeichens, falls vorhanden, oder der unveränderte Zeichencode von "c", wenn der Buchstabe "c" kein Großbuchstaben-Äquivalent hat.*

13.3.10 CRC & Hash

13.3.10.1 Arrays mit symbolischen Indizes

TMD5_Ctx

Kontextstruktur für die MD5-Berechnung

```
// init    Nach dem Setzen auf "0" kann die Kontextstruktur für eine neue
//         Berechnung eines Hashs verwendet werden. Soll eine Berechnung
//         durch mehrfachen Aufruf der Funktion "MD5" erfolgen, darf kein
//         Schreibzugriff auf dieses Element erfolgen.
// tmp     für interne Verwendung, kein Schreibzugriff gestattet

#define TMD5_Ctx[.init, .tmp[22]]
```

13.3.10.2 Funktionen

native CRC16(data{}, len, initial=0xFFFF);

liefert die berechnete Modbus CRC16 der übergebenen Daten

Parameter	*Erklärung*
data	*Array, das die Daten enthält für die die CRC16 berechnet werden soll*
len	*Anzahl der Bytes, die bei der Berechnung berücksichtigt werden sollen*
initial	*Initialwert für die Berechnung der CRC16 - OPTIONAL*

	Erklärung
Rückgabewert	*berechnete CRC16*

native CRC32(data{}, len, initial=0);

liefert die berechnete Ethernet CRC32 der übergebenen Daten

Parameter	*Erklärung*
data	*Array, das die Daten enthält für die die CRC32 berechnet werden soll*
len	*Anzahl der Bytes, die bei der Berechnung berücksichtigt werden sollen*
initial	*Initialwert für die Berechnung der CRC32 - OPTIONAL*

	Erklärung
Rückgabewert	*berechnete CRC32*

native MD5(data{}, len, hash{16}, ctx[TMD5_Ctx] = [0]);

berechnet den MD5 Hash für die übergebenen Daten. Wenn der Hash für einen Datenblock durch mehrfache Aufrufe dieser Funktion berechnet werden soll (z.B. beim blockweisen Empfang von Daten), muss bei jedem Aufruf dieselbe Kontextstruktur übergeben werden. Zwischen den Aufrufen darf die Kontextstrutur nicht verändert werden. Wenn der Hash durch einen einmaligen Aufruf der Funktion berechnet werden kann (d.h. kompletter Datenblock liegt bereits vor) ist die Übergabe einer eigenen Kontextstruktur nicht erforderlich.

Parameter	*Erklärung*
data	*Array, das die Daten enthält für die der MD5 Hash berechnet werden soll*
len	*Anzahl der Bytes, die bei der Berechnung berücksichtigt werden sollen*
hash	*Array zur Aufnahme des berechneten 128-Bit-Hashwerts*
ctx	*Kontextstruktur für die MD5-Berechnung - OPTIONAL (nur notwendig, wenn die Berechnung MD5() mehrfach aufruft)*

	Erklärung
Rückgabewert	*---*

13.3.11 Verschiedene Funktionen

13.3.11.1 Arrays mit symbolischen Indizes

TablePoint

zweispaltige Stützpunkttabelle, Datentyp Integer

```
// key     Spalte, die durchsucht wird
// value   Spalte mit den zurückzuliefernden Ergebniswerten

#define TablePoint[.key, .value]
```

TablePointF

zweispaltige Stützpunkttabelle, Datentyp Float

```
// key     Spalte, die durchsucht wird
// value   Spalte mit den zurückzuliefernden Ergebniswerten

#define TablePointF[Float:.key, Float:.value]
```

TrM2M_Id

Informationen zur Identifikation des Moduls/Geräts

```
// string     rapidM2M Modulidentifikation (z.B. "rapidM2M M23
          HW3.0")
// module     rapidM2M Modultyp (z.B. "M23")
// hwmajor    Hardware: Hauptversionsnummer
// hwminor    Hardware: Nebenversionsnummer
// sn         Geräte-Seriennummer (binär) im BIG Endian Format
//            Bsp.: "010146AF251CED1C" --> "01" in sn{0}, "1C" in sn{7}
// fwmajor    Firmware: Hauptversionsnummer
// fwminor    Firmware: Nebenversionsnummer
// ctx        Messstellenbezeichnung (Kontext)
//            Leerstring, wenn kein Kontext vorhanden ist

#define TrM2M_Id[  .string{50}, .module{10}, .hwmajor, .hwminor,
                   .sn{8}, .fwmajor, .fwminor, .ctx{50}             ]
```

TRTM_Data

Informationen zu einer Laufzeitmessung

```
// runtime       ermittelte Laufzeit in [ms]
// instructions  Anzahl der ausgeführten Instructions
// tmp           für interne Verwendung, kein Schreibzugriff gestattet

#define TRTM_Data[.runtime, .instructions, .tmp[3]]
```

13.3.11.2 Konstanten

Fehlercodes der Funktionen "CalcTable" und "CalcTableF"

```
const
{
  TAB_ERR_FLOOR = -1, // gesuchter Wert kleiner als der erste Tabelleneintrag
  TAB_ERR_CEIL = -2,  // gesuchter Wert größer als der letzte Tabelleneintrag
};
```

13.3.11.3 Funktionen

native getapilevel();

gibt das implementierte API-Level der Skript-Engine aus

	Erklärung
Rückgabewert	*implementiertes API-Level der Skript-Engine* • *1: Initiale Funktionalität* • *2: Funktion "exists ()" hinzugefügt, um die Laufzeitverfügbarkeit zu prüfen.* • *3: Funktion "loadmodule ()" hinzugefügt, um ein Device Logic Modul zu laden*

native exists(const name[]);

prüft, ob die benötigte rapidM2M API Funktion von der Firmware des Geräts unterstützt wird

> ***Wichtiger Hinweis:*** *Verwenden Sie zuvor getapilevel (), um zu überprüfen ob exists() für Ihre Firmwareversion verfügbar ist.*

Parameter	***Erklärung***
name	*Name der rapidM2M API Funktion, die benötigt wird*

	Erklärung
Rückgabewert	• *true, wenn die Funktion verfügbar ist* • *false, wenn die Firmware des Geräts die Funktion nicht unterstützt*

native loadmodule(mod{});

lädt ein Script Modul zur Laufzeit. Dadurch kann die Script-Engine um eigene native Funktionen erweitert werden. Die Implementierung von Operationen als native Funktion ermöglicht eine deutliche Steigerung der Abarbeitungsgeschwindigkeit im Vergleich zur Umsetzung im Script. Ein Script Modul kann mehrere native Funktionen enthalten. Nach dem Aufruf dieser Funktion können die im Script Modul enthaltenen nativen Funktionen auf dieselbe Weise wie die standardmäßig vorhandenen Funktionen der Script-Engine verwendet werden.

Parameter	***Erklärung***
mod	*Byte-Array, welches das zu ladende Script Modul enthält.*

	Erklärung
Rückgabewert	• *OK, wenn erfolgreich* • *ERROR, wenn ein Fehler auftritt*

native rtm_start(measurement[TRTM_Data]);

startet eine Laufzeitmessung

Wichtiger Hinweis: *Die Ausführung gleichzeitiger Messungen ist nicht erlaubt.*

Parameter	***Erklärung***
measurement	*Struktur zur Aufnahme der Informationen zu einer Laufzeitmessung* ***Wichtiger Hinweis:*** *Diese Struktur muss vom Aufruf von "rtm_start()" bis zum Aufruf von "rtm_stop()" persistent sein.*

	Erklärung
Rückgabewert	• *OK, wenn erfolgreich* • *ERROR, wenn ein Fehler auftritt*

native rtm_stop(measurement[TRTM_Data]);

stoppt die Laufzeitmessung und berechnet die seit dem Aufrufen der Funktion "rtm_start()" vergangene Zeit in [ms] sowie die seitdem ausgeführten Instructions. Die ermittelten Werte werden in den Elementen ".runtime" und ".instructions" der übergebenen Struktur zur Aufnahme der Informationen zu einer Laufzeitmessung geschrieben.

Parameter	***Erklärung***
measurement	*Struktur zur Aufnahme der Informationen zu einer Laufzeitmessung* ***Wichtiger Hinweis:*** *Diese Struktur muss vom Aufruf von "rtm_start()" bis zum Aufruf von "rtm_stop()" persistent sein.*

	Erklärung
Rückgabewert	• *OK, wenn erfolgreich* • *ERROR, wenn ein Fehler auftritt*

native CalcTable(key, &value, const table[][TablePoint], size = sizeof table);

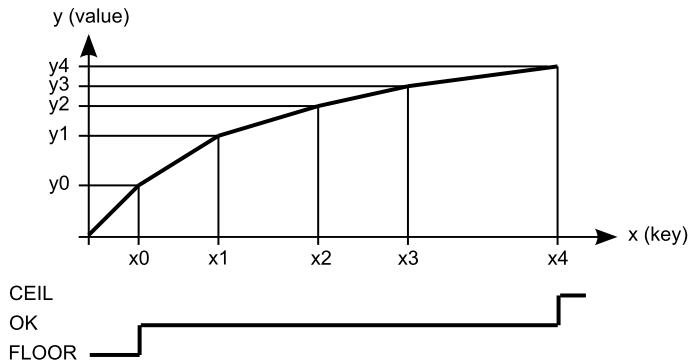
*sucht einen bestimmten Wert in der "key"-Spalte der übergebenen Stützpunktabelle und liefert den entsprechenden Wert der "value"-Spalte der Tabelle. Liegt der gesuchte Wert zwischen zwei Stützpunkten, wird der Rückgabewert zwischen den zwei angrenzenden "value"-Spaltenwerten linear interpoliert (Geradengleichung: y = k*x + d). Mit dieser Funktion können nicht lineare Kennlinien (z.B. Zusammenhang ADC-Wert -> Temperatur) nachgebildet werden.*

Parameter	***Erklärung***
key	*Wert, der für die Suche herangezogen wird*
value	*enthält das Ergebnis der Berechnung durch die Funktion*
table	*Die Tabelle, die durchsucht wird, muss vom Typ "TablePoint" sein.*
size	*Anzahl der Zeilen der Tabelle*

	Erklärung
Rückgabewert	• *OK, wenn der entsprechende Wert gefunden wurde* • *TAB_ERR_FLOOR, wenn der gesuchte Wert kleiner als der erste Tabelleneintrag ist. "value" beinhaltet den ersten Tabelleneintrag.* • *TAB_ERR_CEIL, wenn der gesuchte Wert größer als der letzte Tabelleneintrag ist. "value" beinhaltet den letzten Tabelleneintrag.* • *< OK, wenn ein anderer Fehler auftritt (siehe "Returncodes für allgemeine Zwecke" im Kapitel "Konstanten" auf Seite 108)*

Hinweis: *Ergänzende Erklärung zur Stützpunkttabelle "`table`"*

Die Tabellenzeilen können in einem x/y-Koordinatensystem dargestellt werden. Die Werte der "key"-Spalte werden dabei auf der x-Achse aufgetragen, die dazugehörenden Werte der "value"-Spalte auf der y-Achse.



Darstellung der Stützpunkttabelle als x/y-Koordinatensystem

native CalcTableF(Float:key, &Float:value, const table[][TablePointF], size = sizeof table);

die Funktionsweise entspricht der "CalcTable" Funktion. Der Unterschied besteht darin, dass "Float" der Datentyp für alle Elemente der "CalcTableF" Funktion ist.

native rM2M_GetId(id[TrM2M_Id], len=sizeof id);

liefert die Informationen zur Identifikation des Moduls/Geräts

Parameter	*Erklärung*
id	*Struktur zur Aufnahme der Informationen zur Identifikation des Moduls/Geräts (siehe "TrM2M_Id" im Kapitel "Arrays mit symbolischen Indizes" auf Seite 170)*
len	*Größe (in Cells) der Struktur zur Aufnahme der Informationen - OPTIONAL*

	Erklärung
Rückgabewert	• *Verwendete Größe (in Cells) der Struktur zur Aufnahme der Informationen* • *ERROR, wenn Adresse und/oder Länge der id-Struktur ungültig sind (außerhalb des Skript-Datenspeichers)* • *< OK, wenn ein anderer Fehler auftritt (siehe "Returncodes für allgemeine Zwecke" im Kapitel "Konstanten" auf Seite 108)* ***Hinweis:** Die Firmware des Moduls/Geräts erkennt, wenn eine Device Logic verwendet wird, bei dem die Funktion nur einen Übergabeparameter (Verwendung eines älteren Include-Files) besitzt und liefert aus Kompatibilitätsgründen "OK" anstelle der Größe der Struktur zur Aufnahme der Informationen zurück.*

native heapspace();

liefert den freien Speicherplatz auf dem Heap

	Erklärung
Rückgabewert	*Der freie Speicherplatz auf dem Heap. Der Stack und der Heap besetzen einen gemeinsamen Speicherbereich, so dass dieser Wert die Anzahl der Bytes angibt, die entweder für den Stack oder den Heap übrig sind.*

native funcidx(const name[]);

Liefert den Index einer öffentlichen Funktion; wird verwendet, um Callbacks für die Laufzeitumgebung zu registrieren.

Parameter	*Erklärung*
name	*Name der öffentlichen Funktion*

	Erklärung
Rückgabewert	• *-1, wenn keine Funktion mit dem übergebenen Namen existiert* • *Index der öffentlichen Funktion*

native numargs();

Liefert die Anzahl der an eine Funktion übergebenen Argumente; Dies ist innerhalb von Funktionen mit einer variablen Argumentenliste nützlich.

	Erklärung
Rückgabewert	*Die Anzahl der Argumente, die an eine Funktion übergeben wurden.*

native getarg(arg, index=0);

Diese Funktion liefert ein Argument aus einer variablen Argumentenliste. Wenn das Argument ein Array ist, gibt der "index" den Index des benötigten Array Elements an.

Parameter	***Erklärung***
arg	*Die Sequenznummer des Arguments. Verwenden Sie 0 für das erste Argument.*
index	*Index, falls sich "arg" auf ein Array bezieht*

	Erklärung
Rückgabewert	*Diese Funktion liefert ein Argument aus einer variablen Argumentenliste. Wenn das Argument ein Array ist, gibt "index" den Index des gewünschten Arrayelements an. Der Rückgabewert ist der Wert des Arguments.*

native setarg(arg, index=0, value);

setzt den Wert des Arguments

Parameter	***Erklärung***
arg	*Die Sequenznummer des Arguments. Verwenden Sie 0 für das erste Argument.*
index	*Index, falls sich "arg" auf ein Array bezieht*
value	*Wert auf den das Argument gesetzt werden soll*

	Erklärung
Rückgabewert	• *true, wenn der Wert gesetzt werden konnte* • *false, wenn das Argument oder der Index ungültig sind* *Diese Funktion setzt ein Argument in einer variablen Argumentenliste. Wenn das Argument ein Array ist, gibt "index" den Index des gewünschten Arrayelements an.*

native rand();

liefert eine Zufallszahl aus dem Wertebereich "32-Bit signed Integer". Der Wert "-1" (ERROR) ist allerdings für die Rückmeldung eines Fehlers reserviert.

	Erklärung
Rückgabewert	• *Zufallszahl aus dem Wertebereich "32-Bit signed Integer"* • *ERROR, wenn kein Zufallszahlengenerator verfügbar ist*

native delay_us(us);

blockierende Delay-Funktion. Die Ausführung der Device Logic wird gestoppt und die folgende Codezeile erst nach Ablauf der Verzögerungszeit ausgeführt.

Parameter	***Erklärung***
us	*Verzögerungszeit (1...10000 [µs]).*

	Erklärung
Rückgabewert	• *OK, wenn erfolgreich* • *ERROR, wenn ein Fehler auftritt*

13.3.12 Console

native print(const string[]);

druckt den angegebenen String auf die Standardausgabe

Parameter	***Erklärung***
string	*die auszugebende Zeichenfolge. Diese darf auch Escape-Sequenzen enthalten.*

	Erklärung
Rückgabewert	*OK*

native printf(const format[], {Float,Fixed,_}:...);

druckt den übergebenen Format-String auf die Standardausgabe. Die Arbeitsweise der Funktionen entspricht jener der Standard ANSI-C Implementierung.

Hinweis:

- *Zeichen können verloren gehen wenn der Ausgabepuffer der Konsole überläuft.*
- *Verwenden Sie sprintf () um in einem String Puffer anstatt einer Konsole zu schreiben.*

Parameter	***Erklärung***
format[]	*die zu verwendende Format-Zeichenkette (C-style Formatierungscodes)* *%b : Zahl im Binärradix* *%c : Zeichen* *%d : Zahl im Dezimalradix* *%f : Fließkommazahl* *%s : String* *%x : Zahl im Hexadezimalradix* *... : s32 | f32 | astr - Zusätzliche Argumente.* *Abhängig von der Formatzeichenkette kann die Funktion eine Sequenz von zusätzlichen Argumenten erwarten, die jeweils einen Wert zum Ersetzen eines Formatspezifikators in der Formatzeichenkette enthalten.*

	Erklärung
Rückgabewert	• *Anzahl der gedruckten Zeichen* • *ERROR, falls nicht erfolgreich*

native setbuf(buf{}, size);

stellt der Firmware einen Puffer aus dem für die Device Logic reservierten RAM-Bereich zur Verfügung der für die Ausgabe von Strings mittels der Funktion "printf()" verwendet wird. Beim Aufruf dieser Funktion wird vom in die Firmware integrierten Puffer mit einer Größe von 256 Byte auf den übergebenen Puffer umgeschaltet.

> ***Wichtiger Hinweis:*** *Der Puffer muss während der gesamten Nutzung durch die Firmware gültig sein (d.h. er muss als globale oder static Variable definiert werden).*

Parameter	***Erklärung***
buf	*statisches Byte-Array das als Puffer für die Ausgabe von Strings verwendet werden soll*
size	*Größe des Puffers in Byte* ***Hinweis:*** *Wird die Funktion erneut aufgerufen und dabei die Größe auf "0" gesetzt, wird wieder auf den integrierten Puffer (256 Bytes) zurückgeschaltet. Das übergebene statische Byte-Array kann anschließend wieder durch die Device Logic verwendet werden.*

	Erklärung
Rückgabewert	• *OK, wenn erfolgreich* • *ERROR, falls nicht erfolgreich*

13.3.13 SMS

> ***Wichtiger Hinweis:*** *Wenn sich das Gerät im "online"-Modus befindet, können keine SMS verarbeitet werden.*

13.3.13.1 Callback Funktionen

public func(const SmsTel[], const SmsText[]);

vom Device Logic Entwickler bereitzustellende Funktion, die aufgerufen wird, wenn eine SMS empfangen wurde

Parameter	***Erklärung***
SmsTel	*String, der die Telefonnummer des Absenders der SMS enthält*
SmsText	*String, der den Inhalt der SMS enthält*

13.3.13.2 Funktionen

native rM2M_SmsInit(funcidx, config);

initialisiert den SMS-Empfang

Parameter	*Erklärung*
funcidx	*Index der öffentlichen Funktion, die aufgerufen werden soll, wenn eine SMS empfangen wurde* *Typ der Funktion: public func(const SmsTel[], const SmsText[]);* ***Wichtiger Hinweis:** Wird eine SMS mit einer Länge von mehr als 160 Zeichen empfangen, wird diese sofort verworfen. Es erfolgt kein Aufruf der angegebenen öffentlichen Funktion.*
config	*reserviert für Erweiterungen*

	Erklärung
Rückgabewert	• *OK, wenn erfolgreich* • *< OK, wenn ein Fehler auftritt (siehe "Returncodes für allgemeine Zwecke" im Kapitel "Konstanten" auf Seite 108)*

native rM2M_SmsClose();

deaktiviert den SMS-Empfang

	Erklärung
Rückgabewert	*OK*

13.3.14 File Transfer

13.3.14.1 Arrays mit symbolischen Indizes

TFT_Info

Eigenschaften eines Dateieintrags

```
// name       Name der Datei
// stamp      Zeitstempel der Datei (Sekunden seit 31.12.1999)
// stamp256   Bruchteil der nächsten begonnenen sec. (Auflösung 1/256 sec.)
// size       Dateigröße in Byte
// crc        Ethernet CRC32 der Datei
// flags      Datei Flags (siehe "Datei Flags" im
//            Kapitel "Konstanten" auf Seite 180)
#define TFT_Info[  .name{256}, .stamp, .stamp256, .size, .crc, .flags ]
```

13.3.14.2 Konstanten

Datei Flags

```
FT_FLAG_READ   = 0x0001,   // Datei kann vom Server gelesen werden.
FT_FLAG_WRITE  = 0x0002,   // Datei kann vom Server geschrieben werden.
FT_FLAG_NODE   = 0x0004    /* Dateiknoten (erforderlich, um serverseitig eine
                              neue Datei anlegen zu können) */
FT_FLAG_SYSTEM = 0x0008    /* Systemdatei (kann von der Device Logic nicht
                              verwendet werden) */
```

File Transfer Kommando

```
FT_CMD_NONE   = 0,
FT_CMD_UNLOCK = 1,         /* File Transfersitzung beendet. Der Server gibt die
                              Sperre wieder frei. */
FT_CMD_LIST   = 2,         /* Der Server fordert die Eigenschaften einer
                              Datei an */
FT_CMD_READ   = 3,         // Der Server fordert einen Block einer Datei an.
FT_CMD_STORE  = 4,         // Der Server fordert das Schreiben einer Datei an.
FT_CMD_WRITE  = 5,         /* Der Server liefert einen Block zum Schreiben in
                              eine Datei. */
FT_CMD_DELETE = 6,         // Der Server fordert das Löschen einer Datei.
FT_CMD_ENUM   = 7,         /* Der Server fordert die Eigenschaften einer
                              Datei an, die zu einem Datei-Knoten gehört */
FT_CMD_RETR   = 8,         /* Der Server fordert eine Datei an, die zu einem
                              Datei-Knoten gehört */
```

13.3.14.3 Callback Funktionen

public func(id, cmd, const data{}, len, ofs);

vom Device Logic Entwickler bereitzustellende Funktion, die beim Empfang eines File Transfer Kommandos aufgerufen wird. Die Callback Funktion muss in der Lage sein, alle File Transfer Kommandos (siehe "File Transfer Kommandos" im Kapitel "Konstanten" auf Seite 180) zu behandeln.

<table>
<tr><th>Parameter</th><th>Erklärung</th></tr>
<tr><td>id</td><td>eindeutige Identifikation, mit der die Datei referenziert wird (wurde bei der Registrierung festgelegt)</td></tr>
<tr><td>cmd</td><td>File Transfer Kommando, das vom System erhalten wurde und das die Callback Funktion verarbeiten muss</td></tr>
<tr><td>data</td><td>Dieser Parameter ist nur beim Empfang der folgenden File Transfer Kommandos relevant:
FT_CMD_STORE: Array, das die Eigenschaften der Datei, die neu angelegt werden soll, enthält. Aufbau:<table><tr><th>Offset</th><th>Bytes</th><th>Erklärung</th></tr><tr><td>0</td><td>4</td><td>Zeitstempel der Datei</td></tr><tr><td>8</td><td>4</td><td>Dateigröße in Byte</td></tr><tr><td>12</td><td>4</td><td>Ethernet CRC32 der Datei</td></tr><tr><td>16</td><td>2</td><td>Datei Flags</td></tr><tr><td>18</td><td>256</td><td>Name der Datei</td></tr></table>
FT_CMD_WRITE: Array, das die Daten, die vom myDatanet-Server erhalten wurden, enthält.
FT_CMD_RETR: Name einer Datei (ASCII), die zu einem Datei-Knoten gehört und vom Server angefordert wurde</td></tr>
<tr><td>len</td><td>Dieser Parameter ist nur beim Empfang der folgenden File Transfer Kommandos relevant:
FT_CMD_READ: Anzahl der vom myDatanet-Server angeforderten Bytes (max. 4kB)
FT_CMD_STORE: Größe des vom myDatanet-Server erhaltenen Dateieigenschaftenblocks
FT_CMD_WRITE: Anzahl der vom myDatanet-Server erhaltenen Bytes
FT_CMD_RETR: Länge des Dateinamens (Anzahl der Zeichen ohne abschießende '\0')</td></tr>
<tr><td>ofs</td><td>Dieser Parameter ist nur beim Empfang der folgenden File Transfer Kommandos relevant:
FT_CMD_READ: Byteoffset innerhalb der Datei des an den myDatanet-Server zu übertragenden Datenblocks
FT_CMD_WRITE: Byteoffset innerhalb der Datei des vom myDatanet-Server erhaltenen Datenblocks
FT_CMD_ENUM: Aufzählungsindex (beginnend mit 0), wenn der Server die Eigenschaften einer Datei anfordert, die zu einem Datei-Knoten gehört[1)]</td></tr>
</table>

[1)]Beim Empfang des File Transfer Kommandos "FT_CMD_ENUM" können mittels der Funktion "FT_SetPropsExt()" die Eigenschaften einer Datei, die dem aktuellem Datei-Knoten zugeordnet werden soll, gesetzt werden. D.h. es wird eine Datei dem Datei-Konten zugeordnet. Nach dem ersten "FT_CMD_ENUM" Kommando sendet das System weitere "FT_CMD_ENUM" Kommandos, bis der Device Logic Entwickler anzeigt, dass er dem aktuellem Datei-Knoten keine weiteren Dateien mehr zuordnen will. Dies muss er signalisieren, indem er beim Setzen der Datei-Eigenschaften mittels der Funktion "FT_SetPropsExt()" die Länge für die "TFT_Info"-Struktur (d.h. der Parameter "len") auf 0 setzt.

13.3.14.4 Funktionen

native FT_Register(const name{}, id, funcidx);

registriert eine Datei, die durch die Device Logic zur Verfügung gestellt wird.

Parameter	*Erklärung*
name	*eindeutiger Dateiname*
id	*eindeutige Identifikation, mit der die Datei später referenziert wird (frei wählbar)*
funcidx	*Index der öffentlichen Funktion, die aufgerufen werden soll, wenn ein File Transfer Kommando empfangen wurde* *Typ der Funktion: public func(id, cmd, const data{}, len, ofs);* ***Wichtiger Hinweis:** Alle File Transfer Kommandos (siehe "File Transfer Kommandos" im Kapitel "Konstanten" auf Seite 180) müssen von dieser öffentlichen Funktion behandelt werden.*

	Erklärung
Rückgabewert	• *OK, wenn erfolgreich* • *< OK, wenn ein Fehler auftritt (siehe "Returncodes für allgemeine Zwecke" im Kapitel "Konstanten" auf Seite 108)*

native FT_RegisterEnum(id, funcidx, props[TFT_Info], len=sizeof props);

registriert einen Datei-Knoten, der durch die Device Logic zur Verfügung gestellt wird. Über einen Datei-Knoten können mehrer Dateien verwaltet werden.

Parameter	***Erklärung***
id	*eindeutige Identifikation, mit der der Datei-Knoten später referenziert wird (frei wählbar)*
funcidx	*Index der öffentlichen Funktion, die aufgerufen werden soll, wenn ein File Transfer Kommando empfangen wurde* *Typ der Funktion: public func(id, cmd, const data{}, len, ofs);* ***Wichtiger Hinweis:** Alle File Transfer Kommandos (siehe "File Transfer Kommandos" im Kapitel "Konstanten" auf Seite 180) müssen von dieser öffentlichen Funktion behandelt werden.*
props	*Struktur, die die Eigenschaften eines Dateieintrags für den Datei-Knoten enthält (siehe "TFT_Info" im Kapitel "Arrays mit symbolischen Indizes" auf Seite 180)* *.name: Jene Teile des Datei-Knoten-Namens, bei denen es sich nicht um eine Wildcard handelt, müssen auch in den Namen der Dateien vorkommen, die mit dem Datei-Knoten verwaltet werden sollen (Wildcard-Matching), damit die Lese- und Schreiboperationen akzeptiert werden.* *.flags: Lese/Schreib-Flags je nach Bedarf, Node-Flag zwingend erforderlich*
len	*Größe (in Cells) der Struktur, die die Eigenschaften eines Dateieintrags enthält - OPTIONAL*

	Erklärung
Rückgabewert	• *OK, wenn erfolgreich* • *< OK, wenn ein Fehler auftritt (siehe "Returncodes für allgemeine Zwecke" im Kapitel "Konstanten" auf Seite 108)*

native FT_Unregister(id);

entfernt eine Datei aus der Registrierung. Die Datei steht für den File Transfer nicht mehr zur Verfügung.

Parameter	***Erklärung***
id	*eindeutige Identifikation, mit der die Datei referenziert wird (wurde bei der Registrierung festgelegt)*

	Erklärung
Rückgabewert	• *OK, wenn erfolgreich* • *< OK, wenn ein Fehler auftritt (siehe "Returncodes für allgemeine Zwecke" im Kapitel "Konstanten" auf Seite 108)*

native FT_SetProps(id, stamp, size, crc, flags);

setzt die Eigenschaften einer Datei

Wichtiger Hinweis: *Diese Funktion muss nach dem Empfang eines "FT_CMD_LIST" Befehls aufgerufen werden.*

Wichtiger Hinweis: *Diese Funktion wird zwar weiterhin zugunsten der Abwärtskompatibilität unterstützt, sollte aber bei neuen Projekten nicht mehr verwendet werden. Alternativ sollte die Funktion "FT_SetPropsExt()" verwendet werden.*

Parameter	***Erklärung***
id	*eindeutige Identifikation, mit der die Datei referenziert wird (wurde bei der Registrierung festgelegt)*
stamp	*Zeitstempel der Datei (Sekunden seit 31.12.1999)*
size	*Dateigröße in Byte*
crc	*Ethernet CRC32 der Datei*
flags	*Datei Flags (siehe "Datei Flags" im Kapitel "Konstanten" auf Seite 180)*

	Erklärung
Rückgabewert	• *OK, wenn erfolgreich* • *< OK, wenn ein Fehler auftritt (siehe "Returncodes für allgemeine Zwecke" im Kapitel "Konstanten" auf Seite 108)*

native FT_SetPropsExt(id, props[TFT_Info], len=sizeof props);

setzt die Eigenschaften einer Datei (erweitertes Format)

Wichtiger Hinweis: *Diese Funktion muss nach dem Empfang eines "FT_CMD_LIST" Befehls aufgerufen werden.*

Parameter	***Erklärung***
id	*eindeutige Identifikation, mit der die Datei referenziert wird (wurde bei der Registrierung festgelegt)*
props	*Struktur, die die Eigenschaften eines Dateieintrags enthält (siehe "TFT_Info" im Kapitel "Arrays mit symbolischen Indizes" auf Seite 180)*
len	*Größe (in Cells) der Struktur, die die Eigenschaften eines Dateieintrags enthält - OPTIONAL*

	Erklärung
Rückgabewert	• *OK, wenn erfolgreich* • *< OK, wenn ein Fehler auftritt (siehe "Returncodes für allgemeine Zwecke" im Kapitel "Konstanten" auf Seite 108)*

native FT_Read(id, const data{}, len);

übergibt die Daten an das System, um sie zum myDatanet-Server zu übertragen. Bereitgestellt werden müssen die Daten durch die mittels "FT_Register()" festgelegte Callback Funktion.

> ***Wichtiger Hinweis:*** *Diese Funktion muss nach dem Empfang eines "FT_CMD_READ" Befehls aufgerufen werden.*

Parameter	***Erklärung***
id	*eindeutige Identifikation, mit der die Datei referenziert wird (wurde bei der Registrierung festgelegt)*
data	*Array, das die Daten enthält, die an das System zur Übertragung an den myDatanet-Server übergeben werden sollen*
len	*Anzahl der zu übergebenden Bytes*

	Erklärung
Rückgabewert	• *OK, wenn erfolgreich* • *< OK, wenn ein Fehler auftritt (siehe "Returncodes für allgemeine Zwecke" im Kapitel "Konstanten" auf Seite 108)*

native FT_Accept(id, newid=-1);

akzeptiert die Datei, die der myDatanet-Server schreiben will. Falls die übergebene eindeutige Identifikationsnummer (Parameter "id") auf einen Dateiknoten verweist, handelt es sich um eine neue Datei. In dem Fall muss für die neue Datei eine eindeutige Identifikationsnummer (Parameter "newid") vergeben werden. Die neue Datei muss außerdem mittels der Funktion "FT_Register()" registriert werden. Die Dateieigenschaften, die vom System an die Callback Funktion übergeben wurden (siehe "Callback Funktionen" auf Seite 180), müssen mittels der Funktion "FT_SetProps()" gespeichert werden.

> ***Wichtiger Hinweis:*** *Diese Funktion muss nach dem Empfang eines "FT_CMD_STORE" Befehls aufgerufen werden.*

Parameter	***Erklärung***
id	*eindeutige Identifikation, mit der die Datei referenziert wird (wurde bei der Registrierung festgelegt)*
newid	*eindeutige Identifikation für die neue Datei, die angelegt werden soll (-1 falls es sich um keine neue Datei handelt) - OPTIONAL*

	Erklärung
Rückgabewert	• *OK, wenn erfolgreich* • *< OK, wenn ein Fehler auftritt (siehe "Returncodes für allgemeine Zwecke" im Kapitel "Konstanten" auf Seite 108)*

native FT_Written(id, len);

bestätigt das Schreiben der Daten, die vom myDatanet-Server erhalten wurden. Der eigentliche Schreibvorgang muss durch die mittels "FT_Register()" festgelegte Callback Funktion erfolgen. Der Callback Funktion werden vom System die zu schreibenden Daten (siehe "Callback Funktionen" auf Seite 180) übergeben.

> ***Wichtiger Hinweis:*** *Diese Funktion muss nach dem Empfang eines "FT_CMD_WRITE" Befehls aufgerufen werden.*

Parameter	***Erklärung***
id	*eindeutige Identifikation, mit der die Datei referenziert wird (wurde bei der Registrierung festgelegt)*
len	*Anzahl der geschriebenen Bytes*

	Erklärung
Rückgabewert	• *OK, wenn erfolgreich* • *< OK, wenn ein Fehler auftritt (siehe "Returncodes für allgemeine Zwecke" im Kapitel "Konstanten" auf Seite 108)*

native FT_Error(id);

dient zum Anzeigen eines Fehlers im Dateihandling und beendet jeglichen Datei-Befehl.

Parameter	***Erklärung***
id	*eindeutige Identifikation, mit der die Datei referenziert wird (wurde bei der Registrierung festgelegt)*

	Erklärung
Rückgabewert	• *OK, wenn erfolgreich* • *< OK, wenn ein Fehler auftritt (siehe "Returncodes für allgemeine Zwecke" im Kapitel "Konstanten" auf Seite 108)*

13.4 Device Logic Fehlercodes

Sollte beim Durchlaufen der Device Logic ein Fehler auftreten, wird der entsprechende Fehlercode ins Gerätelog eingetragen und die Device Logic neu gestartet. Kommt es innerhalb von 24h mehr als 3 Mal zu einem derartiger Fehler, wird die Device Logic deaktiviert und das Errorhandling aktiviert (siehe "Errorhandling" auf Seite 36). Bei allen Log-Einträgen bis auf "SCRIPT_ERR" enthält der Parameter den 32-Bit Instruction Pointer der Abstract machine (AMX). Da im Parameter eines Log-Eintrags nur 16-Bit Werte gespeichert werden können, werden jeweils 2 Einträge im Gerätelog erzeugt. Der erste Eintrag enthält Bit31-Bit16 und der zweite Eintrag enthält Bit15-Bit0 des 32-Bit Instruction Pointers. Eine Anleitung zum Auswerten des Gerätelogs finden Sie im Kapitel "Karteireiter "Log""(siehe "Karteireiter "Log"" auf Seite 89).

Log-Eintrag		Parameter		Beschreibung
Code	**Klartext**	**Code**	**Klartext**	
3000 (1/2)	SCRIPT_ERR (1/2)	0	NO SCRIPT	keine gültige Device Logic vorhanden
		1	SCRIPT UPDATE	neue Device Logic erhalten
		2	SCRIPT EXCEPT LOOP	Exception Loop erkannt (4 Systemstarts nach Exeption innerhalb von 10min.) Die Device Logic wird deaktiviert und das Errorhandling aktiviert (siehe "Errorhandling" auf Seite 36). Es erfolgt auch eine Neu-Formatierung des Filesystems. D.h. alle bisher aufgezeichneten Daten sowie Log-Einträge gehen verloren. Dies wird durch den zusätzlichen Log-Eintrag "LOG REFORMATFILE" signalisiert.
		3	SCRIPT SOFT ERROR 1	Erstmaliges Auftreten eines Laufzeitfehlers innerhalb von 24h. Device Logic wurde neu gestartet.
		4	SCRIPT SOFT ERROR 2	Laufzeitfehler zum zweiten Mal innerhalb von 24h aufgetreten. Device Logic wurde neu gestartet.
		5	SCRIPT SOFT ERROR 3	Laufzeitfehler zum dritten Mal innerhalb von 24h aufgetreten. Device Logic wurde neu gestartet. Sollte innerhalb von 24h ein weiterer Laufzeitfehler auftreten, wird die Device Logic deaktiviert und das Errorhandling aktiviert (siehe "Errorhandling" auf Seite 36).
		6	SCRIPT UPDATE ERROR	Fehler beim Installieren der im Zuge des Downloads erhaltenen Device Logic. Die Verbindungsart „Intervall & Wakeup“ wurde aktiviert und alle 1h erfolgt eine Verbindung zum Server.
		7	SCRIPT SYSTEM SHUTDOWN	reserviert für Erweiterungen
		8	SCRIPT DOWNLOAD ERROR	Verbindungsabbruch während des Downloads der Device Logic Die bestehende Device Logic kann dadurch, dass sie sich im RAM befindet, bis zum nächsten PowerOn weiterhin ausgeführt werden. Nach einem PowerOn ist die Ausführung nicht mehr möglich und es wird der Fehler "SCRIPT_ERR, NO SCRIPT" ins Gerätelog eingetragen.

Log-Eintrag		Parameter		Beschreibung
Code	**Klartext**	**Code**	**Klartext**	
3000 (2/2)	SCRIPT_ERR (2/2)	9	SCRIPT DELETED	Die Device Logic wurde manuell (z.B. mittels rapidM2M Studio) gelöscht. Die Verbindungsart „Intervall & Wakeup“ wurde aktiviert und alle 24h erfolgt eine Verbindung zum Server.
3001	AMX_ERR_EXIT	##	---	Abbruch z.B. Max. Anzahl der Befehle (100.000) pro Durchlauf erreicht
3002	AMX_ERR_ ASSERT	##	---	Assertion fehlgeschlagen
3003	AMX_ERR_ STACKERR	##	---	Stack / Heap Kollision (unzureichende Stack-Größe)
3004	AMX_ERR_ BOUNDS	##	---	Array-Index außerhalb des gültigen Bereichs
3005	AMX_ERR_ MEMACCESS	##	---	ungültiger Speicherzugriff z.B. Verwechslung zwischen Cell (32Bit Element) Zugriff [] und Byte Zugriff {}
3006	AMX_ERR_ INVINSTR	##	---	ungültige Anweisung
3007	AMX_ERR_ STACKLOW	##	---	Stack-Unterlauf
3008	AMX_ERR_ HEAPLOW	##	---	Heap Unterlauf
3009	AMX_ERR_ CALLBACK	##	---	keine (ungültige) native Callback-Funktion
3010	AMX_ERR_NATIVE	##	---	Native-Funktion ist fehlgeschlagen
3011	AMX_ERR_DIVIDE	##	---	Division durch Null
3012	AMX_ERR_SLEEP	##	---	Sleep-Modus
3013	AMX_ERR_ INVSTATE	##	---	ungültiger Zustand
3014	reserviert			
3015	reserviert			
3016	AMX_ERR_ MEMORY	##	---	out of memory
3017	AMX_ERR_ FORMAT	##	---	ungültiges/nicht unterstütztes P-Code Dateiformat
3018	AMX_ERR_ VERSION	##	---	Datei ist für eine neuere Version des AMX

Log-Eintrag		Parameter		Beschreibung
Code	**Klartext**	**Code**	**Klartext**	
3019	AMX_ERR_ NOTFOUND	##	---	Datei oder Funktion nicht gefunden
3020	AMX_ERR_INDEX	##	---	ungültiger Index-Parameter (ungültiger Einstiegspunkt)
3021	AMX_ERR_DEBUG	##	---	Debugger kann nicht ausgeführt werden
3022	AMX_ERR_INIT	##	---	AMX nicht initialisiert (oder doppelt initialisiert)
3023	AMX_ERR_ USERDATA	##	---	Benutzer-Datenfeld kann nicht gesetzt werden (Tabelle voll)
3024	AMX_ERR_INIT_ JIT	##	---	JIT kann nicht initialisiert werden.
3025	AMX_ERR_ PARAMS	##	---	fehlerhafter Parameter
3026	AMX_ERR_ DOMAIN	##	---	Domain-Fehler. Das Ergebnis des Ausdrucks ist nicht im gültigen Bereich.
3027	AMX_ERR_ GENERAL	##	---	allgemeiner Fehler (unbekannter oder nicht spezifizierter Fehler)
3028	AMX_ERR_ OVERLAY	##	---	Overlays werden nicht unterstützt (JIT) oder sind nicht initialisiert.
3050	LOG_NOSCRIPT_ ERR	1	SCRIPT File not available	Meta-Informationen zur Device Logic nicht verfügbar
		2	SCRIPT File not fully downloaded	Inkonsistenz zwischen den Meta-Informationen zur Device Logic und der Device Logic selbst erkannt
		3	SCRIPT Error reading file	Fehler beim Lesen der Information, um welchen Device Logic Typ es sich handelt
		4	SCRIPT Marked faulty	Device Logic ist als Fehlerhaft gekennzeichnet (Es sind 4 Laufzeitfehler innerhalb von 24h aufgetreten)
		5	SCRIPT Invalid type	Ungültiger Device Logic Typ erkannt
		6	SCRIPT Error loading program	Fehler beim initialisieren der Device Logic
		7	SCRIPT Exception loop detected	Exception Loop erkannt (4 Systemstarts nach Exception innerhalb von 10min.)
		8	SCRIPT Invalid filesize	Wenn die Größe des Binarys <= 1 Byte ist

13.5 Syntax

13.5.1 Allgemeine Syntax

13.5.1.1 Format

Bezeichner, Zahlen und Zeichen werden durch Leerzeichen, Tabulatoren, Zeilenumbrüche und "Form Feed" getrennt. Eine Serie von einer oder mehreren dieser Separatoren wird als Leerraum erkannt.

13.5.1.2 Optionale Semikolons

Semikolons (um ein Statement zu beenden) sind optional, wenn sie am Ende einer Zeile auftreten. Semikolons sind notwendig, um mehrere Statements in einer Zeile zu trennen. Ein Ausdruck kann auf mehrere Zeilen aufgeteilt werden, jedoch müssen Postfix-Operatoren in derselben Zeile wie der Operand stehen.

13.5.1.3 Kommentare

Text zwischen den Symbolen /* und */ (beide Symbole können auf derselben oder auf unterschiedlichen Zeilen stehen) und Text nach // (bis zum Ende einer Zeile) sind Kommentare. Kommentare dürfen nicht verschachtelt werden. Der Compiler betrachtet Kommentare als Leerzeichen. Ein Kommentar, der mit "/** " (zwei Sterne und ein Leerzeichen nach dem zweiten Stern) beginnt und mit einem "*/" endet, ist ein Dokumentationskommentar. Ein Kommentar, der mit "/// " (drei Schrägstriche und ein Leerzeichen nach dem dritten Schrägstrich) beginnt, ist ebenfalls ein Dokumentationskommentar. Der Parser kann den Dokumentationskommentar in unterschiedlicher Weise unterstützen, zum Beispiel könnte er eine Online-Hilfe daraus generieren.

13.5.1.4 Bezeichner

Namen von Variablen, Funktionen und Konstanten. Bezeichner bestehen aus den Zeichen a...z, A...Z, 0...9, _ oder @. Das erste Zeichen darf keine Ziffer sein. Die Zeichen @ und _ alleine sind keine gültigen Bezeichner, z.B. "_Up" ist ein gültiger Bezeichner, aber "_" ist es nicht. Es wird zwischen Groß- und Kleinschreibung unterschieden. Der Parser schneidet Bezeichner ab einer bestimmten Länge ab. Es werden standardmäßig nur die ersten 16 Zeichen für die Unterscheidung herangezogen.

13.5.1.5 Reservierte Schlüsselworte

Statements	Operator	Direktiven	Andere
assert	defined	defined	defined
break	sizeof	sizeof	sizeof
case	state	state	state
continue	tagof	tagof	tagof
default			
do			
else			
exit			
for			
goto			
if			
return			
sleep			
state			
switch			
while			

13.5.1.6 Numerische Konstanten

13.5.1.6.1 Numerische Integer-Konstanten

Binär
0b gefolgt von einer Serie von 0 und 1

Dezimal
eine Serie von Ziffern zwischen 0 und 9

Hexadezimal
0x gefolgt von einer Serie von Ziffern zwischen 0 und 9 und den Buchstaben a bis f

13.5.1.6.2 Numerische Gleitkomma-Konstanten

Eine Gleitkommazahl ist eine Zahl mit einem Nachkommateil. Eine Gleitkommazahl beginnt mit einer oder mehreren Ziffern, beinhaltet einen Dezimaltrennpunkt und hat zumindest eine Ziffer nach dem Dezimaltrennpunkt. z.B. "12.0" und "0.75" sind gültige Gleitkommazahlen. Optional kann noch ein Exponent angehängt werden. Die Notation ist der Buchstabe "e" (Kleinbuchstabe), gefolgt von einer ganzzahligen numerischen Konstante. Z.B. "3.12e4" oder "12.3e-3" sind gültige Gleitkommazahlen mit Exponent.

13.5.2 Variablen

13.5.2.1 Deklaration

Das Schlüsselwort "new" deklariert eine neue Variable. Für spezielle Deklarationen wird das Schlüsselwort "new" durch "static" ersetzt (siehe "Statische lokale Deklaration" auf Seite 192). Sofern sie nicht explizit initialisiert wird, ist der Wert der neuen Variablen Null.

Eine Variablendeklaration kann auftreten

- an jeder Position, an der ein Ausdruck gültig ist - lokale Variable
- an jeder Position, an der eine Funktionsdeklaration oder eine Implementation der Funktion gültig ist - globale Variablen;
- im ersten Ausdruck einer "for" Schleife (siehe "for (Ausdruck 1 ; Ausdruck 2 ; Ausdruck 3) Statement " auf Seite 203) - lokale Variable

Beispiel:

```
new a;     // ohne Initialisierung (Wert ist 0)
new b = 3; // mit Initialisierung (Wert ist 3)
```

13.5.2.2 Lokale Deklaration

Eine lokale Deklaration erscheint innerhalb eines Anweisungs-Blocks. Auf eine Variable kann nur innerhalb dieses Blocks und der darin enthaltenen Blöcke zugegriffen werden. Eine Deklaration innerhalb des ersten Ausdrucks einer Schleifenanweisung ist ebenfalls eine lokale Deklaration.

13.5.2.3 Globale Deklaration

Eine globale Deklaration erscheint außerhalb einer Funktion und eine globale Variable kann in jeder Funktion verwendet werden. Globale Variablen können nur mit konstanten Ausdrücken initialisiert werden.

13.5.2.4 Statische lokale Deklaration

Eine lokale Variable wird zerstört, wenn die Ausführung den Block verlässt, in dem die Variable geschaffen wurde. Lokalen Variablen in einer Funktion existieren nur während der Laufzeit der genannten Funktion. Jede neuer Aufruf der Funktion erstellt und initialisiert neue lokale Variablen. Wenn eine lokale Variable mit dem Schlüsselwort "static" anstatt "new" deklariert ist, bleibt die Variable auch nach dem Ende einer Funktion im Speicher. Dies bedeutet, dass statische lokale Variablen eine private, dauerhafte Speicherung bereitstellen, die nur in einer einzigen Funktion (oder einem Block) zugänglich sind. Wie globale Variablen, können statische lokale Variablen nur mit konstanten Ausdrücken initialisiert werden.

13.5.2.5 Statische globale Deklaration

Eine statische globale Variable verhält sich wie eine globale Variable, mit dem Unterschied, dass die Variable nur in der Datei gültig ist, in der sie deklariert wurde. Um eine globale Variable statisch zu deklarieren, ersetzen Sie das Schlüsselwort "new" mit "static".

13.5.2.6 Gleitkommawerte

Gleitkommawerte werden unterstützt. Diese können an jeder Stelle eingesetzt werden, an der eine Variablendeklaration gültig ist.

Beispiel:

```
new Float:a;       // ohne Initialisierung (Wert ist 0.0)
new Float:b = 3.0; // mit Initialisierung (Wert ist 3.0)
```

13.5.3 Konstante Variablen

Es ist manchmal notwendig eine Variable zu erstellen, die einmal initialisiert wird und dann nicht mehr verändert werden soll. Eine solche Variable verhält sich ähnlich wie eine symbolische Konstante, aber sie ist

dennoch eine Variable. Um eine konstante Variable zu deklarieren, legen Sie das Schlüsselwort "const" zwischen das Schlüsselwort, das die Variablendeklaration ("new", "static") startet und den Namen der Variablen.

Beispiel:

```
new const address[4] = { 192, 0, 168, 66 }
static const status /* initialized to zero */
```

Typische Situationen, in denen man eine konstante Variable nutzen könnte, sind:

- Um eine "array"-Konstante zu erstellen. Auf symbolische Konstanten kann nicht per Index zugegriffen werden.
- Ein besonderer Fall ist, wenn die Array-Argumente in einer Funktion als "const" markiert werden. Array-Argumente werden immer per Referenz übergeben. Wenn sie als "const" deklariert werden, schützt sie das vor ungewollten Änderungen. Siehe Beispiele von "const-Funktionsargumenten" im Kapitel "Funktionsargumente ("call-by-value" versus "call-by-reference")" auf Seite 206.

13.5.4 Array Variablen

13.5.4.1 Eindimensionales Array

Die Syntax name[constant] deklariert "name" als ein Array aus “constant” Elementen, wobei jedes Element ein Eintrag ist. "name" ist ein Platzhalter für den Namen der Variable und "constant" ist ein positiver Wert ungleich Null. "constant" ist optional und kann weggelassen werden. Wenn kein Wert zwischen den Klammern steht, ist die Anzahl von Elementen gleich der Anzahl der Initialwerte. Der Array-Index-Bereich ist "Null-basierend", das bedeutet, dass das erste Element "name[0]" und das letzte Element "name[constant-1]" ist.

13.5.4.2 Initialisierung

Datenobjekte können bei ihrer Deklaration initialisiert werden. Der initialisierte Wert von globalen Datenobjekten muss ein konstanter Wert sein. Arrays, global oder lokal, müssen ebenfalls mit konstanten Werten initialisiert werden. Nicht initialisierte Daten sind standardmäßig Null.

Beispiele:

Auflistung: gültige Deklaration

```
new i = 1
new j                                   /* j ist 0 */
new k = 'a'                             /* k hat den Zeichencode von 'a' */
new a[] = [1,4,9,16,25]                 /* a hat 5 Elemente */
new s1[20] = ['a','b']                  /* die restlichen 18 Elemente sind 0 */
new s2[] = ''Hello world...''           /* ein unpacked string */
```

Auflistung: ungültige Deklaration

```
new c[3] = 4                            /* Ein Array kann nicht auf einen einzelnen
                                           Wert gesetzt werden */
new i = "Good-bye"                      /* Nur ein Array kann einen String halten. */
new q[]                                 /* Unbekannte Größe für ein Array */
new p[2] = { i + j, k - 3 }             /* Arrayinitialisierer müssen Konstanten sein. */
```

13.5.4.3 Progressive Initialisierung für Arrays

Der Punkte-Operator führt die Initialisierung des Arrays aufgrund der letzten beiden initialisierten Werte weiter. Der Punkte-Operator (drei Punkte, "...") initialisiert das Array bis zur Arraygrenze.

Beispiel: Auflistung: Arrayinitialisierer

```
new a[10] = { 1, ... }                  // setzt alle Elemente auf 1
new b[10] = { 1, 2, ... }               // b = 1, 2, 3, 4, 5, 6, 7, 8, 9, 10
new c[8]  = { 1, 2, 40, 50, ... }       // c = 1, 2, 40, 50, 60, 70, 80, 90
new d[10] = { 10, 9, ... }              // d = 10, 9, 8, 7, 6, 5, 4, 3, 2, 1
```

13.5.4.4 Mehrdimensionale Arrays

(Es werden nur Arrays mit bis zu 3 Dimensionen unterstützt)

Mehrdimensionale Arrays sind Arrays, die Referenzen zu weiteren Sub-Arrays enthalten. Zum Beispiel ist ein zweidimensionales Array ein "Array auf Eindimensionale Arrays".

Beispiele für die Deklaration von zweidimensionalen Arrays:

```
new a[4][3]
new b[3][2] = [ [ 1, 2 ], [ 3, 4 ], [ 5, 6 ] ]
new c[3][3] = [ [ 1 ], [ 2, ...], [ 3, 4, ... ] ]
new d[2]{10} = [ "agreement", "dispute" ]
new e[2][] = [ ''OK'', ''Cancel'' ]
new f[][] = [ ''OK'', ''Cancel'' ]
```

Wie die beiden letzten Deklarationen (Variablen "e" und "f") zeigen, hat die letzte Dimension eine nicht spezifizierte Länge. In diesem Fall wird die Länge des Sub-Arrays aus dem dazugehörigen Initialisierer erkannt. Jedes Sub-Array hat eine unterschiedliche Länge. In diesem speziellen Beispiel enthält "e[1][5]" den Buchstaben "l" des Wortes "Cancel", aber "e[0][5]" ist ungültig, da das Sub-Array e[0] nur drei Einträge (die Buchstaben "O", "K", und den Null-Terminator) beinhaltet. Der Unterschied zwischen den Deklarationen der Arrays "e" und "f" ist, dass wir bei "f" den Compiler die Anzahl der höheren Dimension ermitteln lassen. "sizeof f" ist 2 genauso wie "sizeof e" (siehe "Arrays und der "sizeof"-Operator" auf Seite 194).

13.5.4.5 Arrays und der "sizeof"-Operator

Der "sizeof"-Operator gibt die Anzahl der Elemente einer Variablen zurück. Für eine einfache (nicht Array) Variable ist das Ergebnis von "sizeof" immer 1.

Ein Array mit einer Dimension enthält eine Anzahl von Elementen und der "sizeof"-Operator gibt diese Anzahl zurück. Der Codeausschnitt unterhalb würde deshalb "5" ausgeben, da das Array 4 Zeichen und den Null-Terminator enthält.

```
new msg[] = ''Help''
printf(''%d'', sizeof msg);
```

Der "sizeof"-Operator gibt immer die Anzahl der Einträge, auch für ein "packed" Array, zurück. Der Codeausschnitt unterhalb gibt auch "5" aus, da die Variable 5 Einträge enthält, auch wenn diese im Speicher weniger Platz benötigt.

```
new msg{} = "Help"
printf(''%d'', sizeof msg);
```

Bei mehrdimensionalen Arrays kann der "sizeof"-Operator die Anzahl der Elemente jeder Dimension zurückgeben. Für die letzte (niedrigste) Dimension ist ein Element ein einzelner Eintrag, jedoch für die höchste Dimension ist es ein Sub-Array. Beachten Sie, dass im nachfolgenden Codeausschnitt die Syntax "sizeof matrix" die Anzahl der Elemente der höhere Dimension zurückgibt, und dass die Syntax "sizeof matrix[]" die niedrigere Dimension des zweidimensionalen Arrays ausgibt. Der Codeausschnitt gibt 3 (höhere Dimension) und 2 (niedrigere Dimension) aus.

```
new matrix[3][2] = { { 1, 2 }, { 3, 4 }, { 5, 6 } }
printf(''%d %d'', sizeof matrix, sizeof matrix[]);
```

Die Anwendung des "sizeof"-Operators auf mehrdimensionale Arrays ist besonders praktisch, wenn er als Standardwert für Funktionsargumente verwendet wird.

13.5.5 Operatoren und Ausdrücke

13.5.5.1 Zeichenerklärung

Die Anwendung von einigen Operatoren hängt von der jeweiligen Art des Operanden ab. Aus diesem Grund wird in diesem Kapitel folgende Notation angewendet:

e
beliebiger Ausdruck (eng. expression)

v
beliebiger Ausdruck, dem ein Wert zugewiesen werden kann (“lvalue” Ausdruck - Variable)

a
ein Array

f
eine Funktion

s
ein Symbol - dies kann eine Variable, eine Konstante oder eine Funktion sein

13.5.5.2 Ausdrücke

Ein Ausdruck besteht aus ein oder mehreren Operanden mit einem Operator. Der Operand kann eine Variable, eine Konstante oder ein anderer Ausdruck sein. Ein Ausdruck gefolgt von einem Semikolon ist ein Statement.

Beispiele für Ausdrücke:

```
v++ f(a1, a2)
v = (ia1 * ia2) / ia3
```

13.5.5.3 Arithmetik

Operator	Beispiel	Erklärung
+	e1 + e2	Ergebnis der Addition von e1 und e2
-	e1 - e2	Ergebnis der Subtraktion von e1 und e2
	-e	Ergebnis der arithmetischen Negation von e (Zweierkomplement)
*	e1 * e2	Ergebnis der Multiplikation von e1 und e2
/	e1 / e2	Ergebnis der Division e1 durch e2. Das Ergebnis wird zum nächstgelegenen ganzzahligen Wert, der kleiner oder gleich dem Quotienten ist, abgeschnitten. Sowohl positive als auch negative Werte werden abgerundet (Richtung unendlich).
%	e1 % e2	Ergebnis ist der Rest der Division e1 durch e2. Das Vorzeichen ist dasselbe wie bei e2
++	v++	erhöht v um 1. Das Ergebnis des Ausdrucks ist der Wert vor der Erhöhung.
	++v	erhöht v um 1. Das Ergebnis des Ausdrucks ist der Wert nach der Erhöhung.
--	v--	verringert v um 1. Das Ergebnis des Ausdrucks ist der Wert vor der Verringerung.
	--v	verringert v um 1. Das Ergebnis des Ausdrucks ist der Wert nach der Verringerung.

Hinweis: *Das unäre + ist nicht definiert. Die Operatoren ++ und -- ändern den Operanden. Der Operand muss ein "lvalue" sein.*

13.5.5.4 Bit-Manipulation

Operator	Beispiel	Erklärung
~	~e	Ergebnis ist das Einerkomplement von e.
>>	e1 >> e2	Ergebnis der arithmetischen Verschiebung nach rechts von e1 durch e2 Bits. Die Verschiebung ist vorzeichenbehaftet: Das Bit ganz links wird auf die freien Bits des Ergebnisses kopiert.
>>>	e1 >>> e2	Ergebnis der logischen Verschiebung nach rechts von e1 durch e2 Bits. Die Verschiebung ist vorzeichenlos. Die freien Bits des Ergebnisses werden mit 0 aufgefüllt.
<<	e1 << e2	Ergebnis: Verschiebung nach links von e1 durch e2 Bits. Die freien Bits des Ergebnisses werden mit 0 aufgefüllt. Es gibt keinen Unterschied zwischen einer arithmetischen und einer logischen Verschiebung nach links.
&	e1 & e2	Ergebnis ist das bitweise logische "und" von e1 und e2.
|	e1 | e2	Ergebnis ist das bitweise logische "oder" von e1 und e2.
^	e1 ^ e2	Ergebnis ist das bitweise "exklusiv oder" von e1 und e2.

13.5.5.5 Zuweisung

Das Ergebnis eines Zuweisungsausdrucks ist der Wert des Operanden nach der Zuweisung.

Operator	Beispiel	Erklärung
=	v = e	weist den Wert von e der Variable v zu
	v = a	weist das Array a der Variable v zu. v muss ein Array mit derselben Größe und denselben Dimensionen sein wie a. a kann eine Zeichenkette oder ein Array sein.

Hinweis: *Die folgenden Operatoren kombinieren eine Zuweisung mit einer arithmetischen oder bitweisen Operation. Das Ergebnis des Ausdrucks ist der Wert des linken Operanden nach der arithmetischen oder bitweisen Operation.*

Operator	Beispiel	Erklärung
+=	v += e	erhöht v um e
-=	v -= e	vermindert v um e
*=	v *= e	multipliziert v mit e
/=	v /= e	dividiert v mit e
%=	v %= e	weist v den Rest der Division von v und e zu
>>=	v >>= e	verschiebt v arithmetisch um e Bits nach rechts
>>>=	v >>>= e	verschiebt v logisch um e Bits nach rechts
<<=	v <<= e	verschiebt v um e Bits nach links
&=	v &= e	führt ein bitweises "und" von v und e aus und weist das Ergebnis v zu
|=	v |= e	führt ein bitweises "oder" von v und e aus und weist das Ergebnis v zu
^=	v ^= e	führt ein bitweises "exklusiv oder" von v und e aus und weist das Ergebnis v zu

13.5.5.6 Vergleichsoperatoren

Ein logisches "false" wird durch einen Integer-Wert von 0 repräsentiert; ein logisches "true" durch einen Wert, der nicht 0 ist. Ergebnisse eines Vergleichs-Ausdrucks sind entweder 0 oder 1 und ihr "tag" wird auf "bool" gesetzt.

Operator	Beispiel	Erklärung
==	e1 == e2	Das Ergebnis ist "true", wenn e1 und e2 gleich sind.
!=	e1 != e2	Das Ergebnis ist "true", wenn e1 nicht gleich e2 ist.

Hinweis: *Die folgenden Operatoren können verkettet werden, wie im Ausdruck "e1 <= e2 <= e3". Dies bedeutet, dass das Ergebnis "1" ist, wenn jeder einzelne Vergleich zutrifft und "0", wenn zumindest ein Vergleich nicht zutrifft.*

Operator	Beispiel	Erklärung
<	e1 < e2	Das Ergebnis ist ein logisches "true", wenn e1 kleiner ist als e2.
<=	e1 <= e2	Das Ergebnis ist ein logisches "true", wenn e1 kleiner oder gleich e2 ist.
>	e1 > e2	Das Ergebnis ist ein logisches "true", wenn e1 größer ist als e2.
>=	e1 >= e2	Das Ergebnis ist ein logisches "true", wenn e1 größer oder gleich e2 ist.

13.5.5.7 Boolean

Ein logisches "false" wird durch einen Integer-Wert von 0 repräsentiert, ein logisches "true" durch einen Wert, der nicht 0 ist. Ergebnisse eines Vergleichs-Ausdrucks sind entweder 0 oder 1 und ihr "tag" wird auf "bool" gesetzt.

Operator	Beispiel	Erklärung
!	!e	Das Ergebnis ist eine logisches "true", wenn e logisch "false" ist.
||	e1 || e2	Das Ergebnis ist "true", wenn entweder e1 oder e2 (oder beide) logisch "true" sind. Der Ausdruck e2 wird nur ausgewertet, wenn e1 logisch "false" ist.
&&	e1 && e2	Das Ergebnis ist "true", wenn e1 und e2 logisch "true" sind. Der Ausdruck e2 wird nur ausgewertet, wenn e1 logisch "true" ist.

13.5.5.8 Sonstiges

Operator	Beispiel	Erklärung
[ ]	a[e]	Array Index: Das Ergebnis ist der Eintrag an der Position e des Arrays a.
{ }	a{e}	Array Index: Das Ergebnis ist das Zeichen an der Position e des "packed" Arrays a.
( )	f(e1, e2, ... eN)	Das Ergebnis ist der Wert, der von der Funktion f zurückgegeben wird. Die Funktion wird mit den Parametern e1, e2, ... eN aufgerufen. Die Reihenfolge der Auswertung der Parameter ist nicht definiert. (Die Implementation der Script-Engine wertet die Parameter möglicherweise in umgekehrter Reihenfolge aus.)
? :	e1 ? e2 : e3	Das Ergebnis ist entweder e2 oder e3, abhängig vom Wert von e1. Der bedingte Ausdruck ist ein zusammengesetzter Ausdruck mit einem zweiteiligen Operator, "?" und ":". Der Ausdruck e2 wird ausgewertet, wenn e1 logisch "true" ist, e3 wird ausgewertet, wenn e1 logisch "false" ist.
:	tagname: e	"tag" Überschreibung: Der Wert des Ausdrucks ändert sich nicht, jedoch ändert sich der "tag".
defined	defined s	Ergebnis ist "1", wenn das Symbol definiert wurde. Das Symbol kann eine Konstante oder eine globale oder lokale Variable sein. Der "tag" des Ausdrucks ist "bool".
sizeof	sizeof s	Das Ergebnis ist die Anzahl der Elemente der angegeben Variable. Für einfache Variablen und für eindimensionale Arrays ist ein Element ein Eintrag. Für mehrdimensionale Arrays ist das Ergebnis die Anzahl der Elemente (Sub Arrays) in der höchsten Dimension. Fügen Sie [] zum Namen des Arrays hinzu, um eine niedrigere Dimension anzugeben. Wenn die Größe der Variable nicht bekannt ist, dann ist das Ergebnis 0. Wenn dieser Operator in einem "default"-Wert einer Funktion verwendet wird, dann wird der Ausdruck zum Zeitpunkt des Aufrufs der Funktion und nicht zum Zeitpunkt der Definition ausgeführt.
tagof	tagof s	Das Ergebnis ist eine eindeutige Zahl, die den "tag" der Variablen, der Konstanten, des Rückgabewerts einer Funktion oder des Names der "tag"-Bezeichnung repräsentiert. Wenn dieser Operator in einem "default"-Wert einer Funktion verwendet wird, dann wird der Ausdruck zum Zeitpunkt des Aufrufs der Funktion und nicht zum Zeitpunkt der Definition ausgeführt.

13.5.5.9 Priorität der Operatoren

Die nachfolgende Tabelle gruppiert Operatoren mit derselben Priorität, beginnend mit der höchsten Priorität.

Wenn die Auswertung eines Ausdrucks nicht explizit durch Klammern begründet wird, wird sie von den Assoziationsregeln bestimmt. Zum Beispiel: a*b/c ist gleich (a*b)/c auf Grund der links zu rechts Assoziation, und a=b=c ist gleichzusetzen mit a=(b=c).

Operator	Erklärung	Lesefolge
( ) [ ] { }	Funktionsaufruf Array Index (Element) Array Index (Zeichen)	links-nach-rechts
! ~ - ++ -- : defined sizeof tagof	logisches Nicht Einerkomplement Zweierkomplement (unäres Minus) Erhöhung Verringerung "tag"-Überschreibung Symbol Definitions-Status Symbol Größe in "Elementen" eindeutige Zahl des "tags"	rechts-nach-links
* / %	Multiplikation Division Modulo	links-nach-rechts
+ -	Addition Subtraktion	links-nach-rechts
>> >>> <<	arithemtische Verschiebung nach rechts logische Verschiebung nach rechts Verschiebung nach links	links-nach-rechts
&	bitweises "und"	links-nach-rechts
^	bitweises "exklusiv oder"	links-nach-rechts
|	bitweises "oder"	links-nach-rechts
< <= > >=	kleiner als kleiner oder gleich als größer als größer oder gleich als	links-nach-rechts
== !=	gleich ungleich	links-nach-rechts
&&	logisches "und"	links-nach-rechts
||	logisches "oder"	links-nach-rechts
? :	bedingte Ausführung	rechts-nach-links
=	Zuweisung *= /= %= += -= >>= >>>= <<= &= ^= |=	rechts-nach-links
,	Komma	links-nach-rechts

13.5.6 Anweisungen

Ein Statement kann aus einer oder mehreren Zeilen bestehen. Eine Zeile kann zwei oder mehrere Statements enthalten.

Statements zur Ablaufsteuerung (if, if-else, for, while, do-while und switch) können geschachtelt werden.

13.5.6.1 Statement-Etikett

Ein Etikett besteht aus einem Identifizierer gefolgt von einem ":". Ein Etikett ist ein "Sprung-Ziel" eines "goto" Statements.

Jede Anweisung kann mit einem Etikett versehen werden. Es muss dem Etikett ein Statement folgen, dies kann auch ein "leeres Statement" sein.

Der Gültigkeitsbereich eines Etiketts ist die Funktion in der es deklariert wurde d.h. ein "goto"-Statement kann nicht aus der aktuellen Funktion in eine andere Funktion springen.

13.5.6.2 Zusammengesetzte Anweisungen

Eine zusammengesetzte Anweisung (auch Block genannt) ist eine Serie von Null oder mehreren Anweisungen, welche durch Klammern ("{" und "}") umgeben ist. Die schließende Klammer ("}") darf nicht mit einem Semikolon abgeschlossen werden. Jede Anweisung kann durch einen Block ersetzt werden. Eine zusammengesetzte Anweisung, die keine Anweisungen enthält, ist ein Spezialfall und wird "leeres Statement" genannt.

13.5.6.3 Ausdrucksanweisung

Jeder Ausdruck wird zu einem Statement, wenn ein Semikolon (";") angehängt wird. Ein Ausdruck wird auch zu einem Statement, wenn dem Ausdruck bis zum Ende der Zeile nur Leerzeichen folgen und der Ausdruck nicht in der nächsten Zeile weitergeführt werden kann.

13.5.6.4 Leeres Statement

Eine leeres Statement führt keine Anweisungen aus und besteht aus einem Block-Statement ohne Anweisungen, d.h. es besteht aus dem Symbol "{ }". Leere Statements werden in Kontrollflussanweisungen ohne Aktionen eingesetzt (z.B. "while (!iskey()) {}") oder, wenn ein Etikett genau vor einer schließenden Klammer eines Block-Statements definiert wird. Ein leeres Statement endet nicht mit einem Semikolon.

13.5.6.5 assert Ausdruck

bricht das Programm mit einem Laufzeitfehler ab, wenn der Ausdruck logisch "false" ergibt

> ***Hinweis:*** *Dieser Ausdruck schützt vor "unmöglich" oder ungültigen Bedingungen. Im folgenden Beispiel ist eine negative Fibonacci-Zahl ungültig. Die assert-Anweisung markiert diesen Fehler als Programmierer-Fehler. assert-Anweisungen sollten nur Programmierer-Fehler kennzeichnen und niemals Benutzereingaben.*

Beispiel:

```
fibonacci(n)
{
  assert n > 0

  new a = 0, b = 1
  for (new i = 2; i < n; i++)
  {
    new c = a + b
    a = b
    b = c
  }
  return a + b
}
```

13.5.6.6 break

beendet und verlässt das kleinste, umschließende "do"-, "for"- oder "while"-Statement an jedem beliebigen Punkt in der Schleife. Das "break"-Statement bewegt den Programmfluss zum nächsten Statement außerhalb der Schleife.

Beispiel:

```
example(n)
{
  new a = 0

  for(new i = 0; i < n ; i++ )
  {
    a += i

    if(i>10)
      break

    a += 1
  }
  return a
}
```

13.5.6.7 continue

beendet die aktuelle Iteration der kleinsten umschließenden "do"-, "for"- oder "while"-Anweisung und bewegt die Programmsteuerung an den Bedingungsteil der Schleife.

Beispiel

```
example(n)
{
  new a = 0

  for(new i = 0; i < n ; i++ )
  {
    a += i

    if(i>10)
      continue

    a += 1
  }
  return a
}
```

13.5.6.8 do Statement while (Ausdruck)

führt ein Statement aus, bevor der Bedingungsteil (die "while"-Bedingung) evaluiert wird. Das Statement wird wiederholt, solange die Bedingung logisch "true" ist. Das Statement wird zumindest einmal ausgeführt.

Beispiel:

```
example(n)
{
  new a = 0

  do
  {
    a++
  }
  while(n >= 0)

  return a
}
```

13.5.6.9 exit Ausdruck

bricht das Programm ab. Der Ausdruck ist optional, aber wenn er vorhanden ist, muss er in der selben Zeile wie das "exit"-Statement beginnen und enden. Die "exit"-Anweisung gibt den Wert des Ausdrucks zurück an die Hauptanwendung oder gibt Null zurück, wenn kein Ausdruck angegeben wird.

13.5.6.10 for (Ausdruck 1 ; Ausdruck 2 ; Ausdruck 3) Statement

Alle drei Ausdrucke sind optional.

Ausdruck 1:

wird nur einmal ausgewertet, vor Eintritt in die Schleife. Dieser Ausdruck kann zum Initialisieren einer Variablen genutzt werden. Dieser Ausdruck hält auch die Variablendeklaration mittels der "new"-Syntax. Eine Variable, die an dieser Stelle deklariert wird, ist nur innerhalb der Schleife gültig. Es ist nicht möglich einen Ausdruck (mit bereits vorhandenen Variablen) und eine Deklaration von neuen Variablen in diesem Feld zu kombinieren. Es müssen entweder alle Variablen in diesem Feld bereits vorhanden sein, oder sie müssen alle in diesem Bereich deklariert werden.

Ausdruck 2:

Dieser Ausdruck wird vor jedem Durchlauf der Schleife ausgeführt und beendet die Schleife, wenn der Ausdruck logisch "false" zurückgibt. Wenn dieser Ausdruck weggelassen wird, wird das Ergebnis des Ausdrucks 2 als logisch "true" angenommen.

Ausdruck 3:

Dieser Ausdruck wird nach jeder Ausführung des Statements ausgeführt. Die Programmsteuerung bewegt sich von Ausdruck 3 zum Ausdruck 2 für die nächste (bedingte) Iteration der Schleife.

Beispiel:

```
example(n)
{
  new a = 0

  for(new i = 0; i < n; i++)
  {
    a++
  }

  return a
}
```

Das Statement "for (; ;)" ist gleich dem Statement "while (true)".

13.5.6.11 goto Etikett

bewegt die Programmsteuerung (unbedingt) zu der Anweisung, die dem angegebenen Etikett folgt. Das Etikett muss innerhalb der gleichen Funktion wie die "goto"-Anweisung sein. Eine "goto"-Anweisung kann nicht aus einer Funktion springen.

13.5.6.12 if (Ausdruck) Statement 1 else Statement 2

führt das Statement 1 aus, wenn das Ergebnis des Ausdrucks logisch "true" ergibt. Die "else"-Klausel des "if"-Statements ist optional. Wenn das Ergebnis des Ausdruck logisch "false" ergibt und eine "else"-Klausel existiert, dann wird das Statement, das mit der "else"-Klausel assoziiert ist, (Statement 2) ausgeführt.

Beispiel:

```
example(n)
{
  if(n < 0)
    return -1
  else if (n == 0)
    return 0
  else
    return 1
}
```

13.5.6.13 return Ausdruck

beendet die aktuelle Funktion und bewegt die Programmsteuerung zum nächsten Statement nach dem Funktionsaufruf. Der Wert des Ausdrucks wird als Funktionsergebnis zurückgeliefert. Der Ausdruck kann ein Array oder eine Zeichenfolge sein. Der Ausdruck ist optional, wenn er jedoch vorhanden ist, muss er an der selben Zeile beginnen, wie das "return"-Statement. Wenn kein Ausdruck angegeben ist, dann wird Null zurückgeliefert.

13.5.6.14 switch (Ausdruck) { case Liste }

überträgt die Ablaufsteuerung an unterschiedliche Statements innerhalb des "switch" in Abhängigkeit vom Wert des "switch"-Ausdrucks. Der Hauptteil der "switch"-Anweisung ist eine zusammengesetzte Anweisung, die eine Reihe von "case"-Klauseln enthält. Jede "case"-Klausel beginnt mit dem Schlüsselwort "case", gefolgt von einer Liste von Konstanten und einem Statement. Die Liste der Konstanten ist eine Serie von Ausdrücken, getrennt durch Kommas, die jeweils zu einem konstanten Wert ausgewertet werden. Diese Liste endet mit einem Doppelpunkt. Um einen Bereich in dieser Liste anzugeben, trennen Sie die untere und obere Grenze des Bereichs mit einem doppelten Punkt (".."). Ein Beispiel für einen Bereich ist: "case 1..9:".

Das "switch"-Statement bewegt die Ablaufsteuerung zu einer "case"-Klausel, wenn ein Wert der Liste dem Wert des "switch"-Ausdrucks entspricht.

Die "default"-Klausel besteht aus dem Schlüsselwort "default" und einem Doppelpunkt. Die "default"-Klausel ist optional, aber wenn sie angegeben wird, muss sie als letzter Eintrag in der "case"-Liste eingetragen sein. Das "switch"-Statement bewegt die Ablaufsteuerung zur "default"-Klausel, wenn keine der "case"-Klauseln mit dem "switch"-Ausdruck übereinstimmt.

Beispiel:

```
example(n)
{
  new a = 0

  switch (n)
  {
    case 0..3:
      a = 0
    case 4,6,8,10:
      a = 1
    case 5,7:
      a = 2
    case 9:
      a = 3
    default:
      a = -1
  }

  return a
}
```

13.5.6.15 while (Ausdruck) Statement

wertet den Ausdruck aus und führt das Statement aus, wenn das Ergebnis des Ausdrucks logisch "true" ergibt. Nachdem die Anweisung ausgeführt wurde, kehrt die Programmsteuerung erneut zu dem Ausdruck zurück. Das Statement wird daher ausgeführt, solange der Ausdruck logisch "true" ist.

Beispiel:

```
example(n)
{
  new a = 0

  while(n >= 0)
  {
    a++
  }

  return a
}
```

13.5.7 Funktionen

Eine Funktionsdeklaration spezifiziert den Namen der Funktion und die formalen Parameter, die in Klammern eingeschlossen sind. Eine Funktion kann auch einen Wert zurückliefern. Eine Funktion muss global definiert, d.h. außerhalb einer anderen Funktion deklariert werden und ist global verfügbar.

Wenn ein Semikolon der Funktionsdeklaration folgt (anstatt einer Anweisung), dann ist dies eine Vorwärtsdeklaration einer Funktion.

Die "return"-Anweisung setzt den Rückgabewert der Funktion. Zum Beispiel, hat die Funktion "sum" (siehe unten) als Rückgabewert die Summe der beiden Parameter. Der "return"-Ausdruck ist optional.

```
sum(a, b)
{
  return a + b
}
```

Argumente einer Funktion sind (implizit deklarierte) lokale Variablen für diese Funktion. Der Funktionsaufruf bestimmt die Werte der Argumente. Ein weiteres Beispiel für eine vollständige Definition einer Funktion ist "leapyear", die "true" für ein Schaltjahr und "false" für kein Schaltjahr zurückgibt.

```
leapyear(y)
{
  return y % 4 == 0 && y % 100 != 0 || y % 400 == 0
}
```

Die Anweisungen, die in diesem Beispiel verwendet wurden, werden im Kapitel "Operatoren und Ausdrücke" auf Seite 195 behandelt.

Normalerweise beinhalten Funktionen lokale Variablendeklarationen und bestehen aus einer Block-Anweisung.

> ***Hinweis:*** *Im nächsten Beispiel verhindert die "assert"-Anweisung negative Werte für den Exponenten*

```
power(x, y)
{
  /* gibt x hoch y zurück*/
  assert y >= 0

  new r = 1
  for (new i = 0; i < y; i++)
    r *= x

  return r
}
```

Eine Funktion kann mehrere "return"-Anweisungen enthalten, eine wird z.B. benutzt um schnell eine Funktion zu beenden, wenn ungültige Parameter übergeben werden, oder wenn sich herausstellt, dass die Funktion nichts zu tun hat. Wenn eine Funktion ein Array zurückgibt, müssen alle "return" Anweisungen ein Array mit derselben Anzahl von Einträgen zurückgeben.

13.5.7.1 Funktionsargumente ("call-by-value" versus "call-by-reference")

Die "faculty"-Funktion im nächsten Beispiel hat einen Parameter, der in der Schleife benutzt wird um die Fakultät dieser Zahl zu berechnen. Was Aufmerksamkeit verdient ist, dass die Funktion das Argument modifiziert.

```
main()
{
  new v = 5
  new f = faculty(v)
}

faculty(n)
{
  assert n >= 0

  new result = 1
  while (n > 0)
    result *= n--

  return result
}
```

Egal welchen (positiven) Wert die Variable "n" am Beginn der "while"-Schleife hatte, am Ende der Funktion wird "n" Null sein. Am Beispiel der Funktion "faculty" wird der Parameter als Wert ("by value") übergeben, damit ist eine Änderung der Variable "n" nur lokal in der Funktion "faculty" gültig. Mit anderen Worten, die Variable "v" in der Funktion "main()" hat vor und nach dem Aufruf der Funktion denselben Wert.

Argumente können als Wert ("by value") oder als Referenz ("by reference") übergeben werden. Einem Funktionsargument, das als Referenz übergeben werden soll, muss als Präfix "&" dem Namen vorangestellt werden. Als Standard werden der Funktion die Argumente als Wert übergeben.

Beispiel:

```
swap(&a, &b)
{
  new temp = b
  b = a
  a = temp
}
```

Um eine Array einer Funktion zu übergeben, fügen Sie ein Klammernpaar ("[]") dem Namen des Arguments an. Es kann auch zusätzlich die Anzahl der Einträge angegeben werden. Dies verbessert die Fehlererkennung des Parsers des Compilers.

Beispiel:

```
addvector(a[], const b[], size)
{
  for (new i = 0; i < size; i++)
    a[i] += b[i]
}
```

Arrays werden immer als Referenz übergeben.

> ***Hinweis:*** *Das Array "b" im oben gezeigten Beispiel wird in der Funktion nicht verändert. Dieses Funktionsargument wurde als "const" deklariert, um dies explizit zu machen. Zusätzlich zur verbesserten Fehlererkennung, erlaubt es dem Compiler einen effizenteren Code zu generieren.*

Das folgende Codebeispiel ruft die Funktion "addvector" auf und addiert zu jedem Element der Variablen "vect" den Wert 5:

```
new vect[3] = [ 1, 2, 3 ]

addvector(vect, [5, 5, 5], 3)

/* vect[] beinhaltet nun die Werte 6, 7 und 8 */
```

13.5.7.2 Benannte Parameter versus positionsgebundene Parameter

In den vorangegangenen Beispielen war die Reihenfolge der Parameter bei einem Funktionsaufruf wichtig, da jeder Parameter zu dem Funktionsparameter mit derselben Position kopiert wurde. Zum Beispiel bei der "weekday" Funktion, die unterhalb definiert wird, würde der Aufruf "weekday(12, 31, 1999)" lauten, um den Wochentag des letzten Tages des letzten Jahrhunderts zu erhalten.

```
weekday(month, day, year)
{
  /* gibt den Tag der Woche zurück: 0=Samstag, 1=Sonntag, etc. */
  if (month <= 2)
    month += 12, --year

  new j = year % 100
  new e = year / 100
  return (day + (month+1)*26/10 + j + j/4 + e/4 - 2*e) % 7
}
```

Das Datumsformat unterscheidet sich je nach Kultur und Nation, während in den Vereinten Staaten von Amerika das Format Monat/Tag/Jahr verbreitet ist, verwenden europäische Staaten oft das Format Tag/Monat/Jahr und in technischen Publikationen wird Jahr/Monat/Tag (ISO/IEC 8824) verwendet. Mit anderen Worten, keine Reihenfolge der Parameter ist "standardisiert" oder "normal". Aus diesem Grund gibt es eine alternative Möglichkeit, um Parameter an eine Funktion zu übergeben: die "benannten Parameter". Diese wird im nächsten Beispiel gezeigt (die Funktion wurde genau wie im vorherigen Beispiel deklariert).

```
new wkday1 = weekday( .month = 12, .day = 31, .year = 1999)
new wkday2 = weekday( .day = 31, .month = 12, .year = 1999)
new wkday3 = weekday( .year = 1999, .month = 12, .day = 31)
```

Bei "benannten Parametern" wird ein Punkt (".") dem Namen des Arguments vorangestellt. Das Argument der Funktion kann auf einen beliebigen Ausdruck gesetzt werden, der gültig für das Argument ist. Das Gleichheitszeichen ("=") hat im Falle eines benannten Parameters nicht die Bedeutung einer Zuordnung, sondern verknüpft den Ausdruck mit einem Funktionsargument.

Es können positionsgebundene und benannte Parameter vermischt werden, jedoch müssen die positionsgebundenen vor den benannten Parametern angegeben werden.

13.5.7.3 Standardwerte von Funktionsargumenten

Ein Funktionsargument kann einen Standardwert haben. Der Standardwert eines Funktionsarguments muss eine Konstante sein. Um einen Standardwert anzugeben, fügen Sie an den Namen des Parameters ein Gleichheitszeichen ("=") und den Wert an.

Wenn bei einem Funktionsaufruf ein Platzhalter anstelle eines gültigen Funktionsparameters angegeben wird, wird der Standardwert übernommen. Der Platzhalter ist das Unterstrichzeichen ("_"). Der Argumentplatzhalter ist nur für Parameter mit einem Standardwert gültig.

Die rechten Argumentplatzhalter können von der Argumentenliste entfernt werden.

Zum Beispiel, wenn die Funktion "increment" wie folgt definiert ist:

```
increment(&value, incr=1)
{
  value += incr
}
```

sind die folgenden Funktionsaufrufe alle gleich:

```
increment(a)
increment(a, _)
increment(a, 1)
```

Standardwerte für Argumente, die als Referenz übergeben werden, sind hilfreich um diese Parameter optional zu machen. Zum Beispiel, wenn die Funktion "divmod" geschrieben wurde, um sowohl den Quotienten als auch den Rest als Parameter zu übergeben.

```
divmod(a, b, &quotient=0, &remainder=0)
{
  quotient = a / b
  remainder = a % b
}
```

Mit der vorangegangenen Definition der Funktion "divmod" sind die folgenden Funktionsaufrufe alle gültig:

```
new p, q

divmod(10, 3, p, q)
divmod(10, 3, p, _)
divmod(10, 3, _, q)
divmod(10, 3, p)
divmod 10, 3, p, q
```

Das nächste Beispiel addiert die Werte von einem Array zu einem anderen. Wenn nur ein Parameter angegeben wird, dann werden die Werte des Arrays um 1 erhöht:

```
addvector(a[], const b[] = {1, 1, 1}, size = 3)
{
  for (new i = 0; i < size; i++)
    a[i] += b[i]
}
```

13.6 Unterschiede zu C

- Der Eingabe-Mechanismus von C ist nicht vorhanden. Es handelt sich um eine "integer-only" Variante von C. Es gibt keine Strukturen oder Unions. Floating Point-Unterstützung muss mit benutzerdefinierten Operatoren und der Hilfe von nativen Funktionen implementiert werden.

- Die Syntax für Gleitkommawerte ist strenger als die in C. Im Gegensatz zu C werden Werte wie ".5" und "6." nicht akzeptiert. Es ist zwingend die Schreibweise "0.5" und "6.0" zu verwenden. In C ist der Dezimalpunkt optional, wenn ein Exponent enthalten ist, so kann man in C "2E8" schreiben; Auch der Großbuchstaben "E" wird nicht akzeptiert. Verwenden Sie den Kleinbuchstaben "e". Zudem ist das Komma erforderlich: z.B. "2.0e8"(siehe "Numerische Konstanten" auf Seite 191).
- "Zeiger" werden nicht unterstützt. Für die Übergabe von Funktionsparametern als Referenz existiert ein "Referenz"-Argument (siehe "Funktionsargumente ("call-by-value" versus "call-by-reference")" auf Seite 206). Das "Platzhalter"-Argument ersetzt einige Verwendungen des NULL-Zeigers (siehe "Standardwerte von Funktionsargumenten" auf Seite 208).
- Zahlen können mit Hexadezimal-, Dezimal-, oder Binärbasis angegeben werden. Die Oktale Basis wird nicht unterstützt (siehe "Numerische Konstanten" auf Seite 191). Hexadezimale Zahlen müssen mit "0x" ("x" in Kleinbuchstaben) beginnen. Das Präfix “0X” ist ungültig.
- "Cases" in einem "switch"-Statement sind nicht "durchfallend". Es muss dem "case"-Label zumindest eine Anweisung folgen. Um mehrere Anweisungen auszuführen, müssen Sie ein zusammengesetztes Statement (mit {}) erstellen (siehe "switch (Ausdruck) { case Liste }" auf Seite 204). Die "switch"-Anweisung ist als ein strukturiertes "if" zu betrachten. In C/C++ hingegen ist die "switch"-Anweisung ein "bedingtes goto".
- Eine "break"-Anweisung beendet nur Schleifen. In C/C++ beendet die "break"-Anweisung auch ein "case" in einer "switch"-Anweisung.
- "array Zuweisungen" werden mit der Limitation unterstützt, dass beide Arrays die gleiche Länge haben müssen. Zum Beispiel, wenn "a" und "b" Arrays mit 6 Zeilen sind, dann ist der Ausdruck "a=b" gültig. Neben Zeichenketten, werden auch literale Arrays und somit Ausdrücke wie "a = {0,1,2,3,4,5}" unterstützt, wobei "a" eine Array Variable mit 6 Elementen ist.
- "defined" ist ein Operator und keine Präprozessor-Direktive. Der "defined" Operator arbeitet mit Konstanten (deklariert mit "const"), globalen Variablen, lokalen Variablen und Funktionen.
- Der "sizeof"-Operator gibt die Größe von Variablen in "Elementen" zurück und nicht in "Bytes". Ein Element ist ein Eintrag oder ein Sub-Array. Weitere Details finden Sie im Kapitel "Sonstiges" auf Seite 198.
- Eine leere Anweisung ist ein leerer Block (mit {}), nicht ein Semikolon (siehe "Zusammengesetzte Anweisungen" auf Seite 200). Diese Änderung verhindert häufige Fehler.
- Eine Division erfolgt in der Weise, dass der Rest der Division das gleiche Vorzeichen hat (oder hätte) wie der Nenner. Bei der Division (Operator "/") erfolgt die Rundung immer zum kleineren ganzzahligen Wert (wobei -2 kleiner ist als -1). D.h. 5/2=2 (2,5 wird zu 2 abgerundet), -5/2=-3 (-2,5 wird zu -3 abgerundet). Der "%"-Operator ergibt immer ein positives Ergebnis unabhängig vom Vorzeichen des Zählers (siehe "Operatoren und Ausdrücke" auf Seite 195).
- Es gibt keinen unären Operator "+", da dieser sowieso ein "no-operation"-Operator ist ("a = +1" ist nicht gültig; korrekt: "a = 1").
- Drei der bitweisen Operatoren haben andere Prioritäten als in C. Die Prioritätsstufe des "&", "^" und "|" Operators ist höher als die relationalen Operatoren. Dennis Ritchie erklärte, dass diese Operatoren in C ihre niedrigen Prioritätsstufen bekamen, weil frühe C-Compiler noch nicht über die logischen Operatoren "&&" und "| |" verfügten, so dass stattdessen bitweise "&" und "|" verwendet wurden.
- Das Schlüsselwort "const" implementiert die "enum" Funktionalität von C.

- In den meisten Fällen sind Vorwärts-Deklarationen von Funktionen (d.h. Prototypen) nicht notwendig da ein 2-Pass-Compiler verwendet wird. Er erkennt alle Funktionen beim ersten Durchlauf und verwendet diese beim zweiten Durchlauf. Benutzerdefinierte Operatoren müssen jedoch vor der Benutzung deklariert werden. Falls vorhanden, müssen Vorwärts-Deklarationen genau mit der Definition der Funktion übereinstimmen. Die Parameternamen in den Prototypen und den Definitionen der Funktionen müssen ident sein. Aufgrund der "benannte Parameter"-Funktion sind die Parameter-Namen im Prototyp von Bedeutung. Prototypen werden verwendet, um vorwärts deklarierte Funktionen aufzurufen. Um diese dabei mit benannten Parametern zu verwenden, muss der Compiler bereits die Namen der Parameter (und ihre Position in der Parameterliste) kennen. Aus diesem Grund müssen die Parameternamen in den Prototypen mit jenen in den Definitionen übereinstimmen.
- Variable werden automatisch mit „0“ initialisiert. Es ist daher nicht notwendig, diese explizit auf „0“ zu setzen.

Kapitel 14 Data Descriptor

Das Grundprinzip des myDatalogC33x ist eine sogenannte Storage-2-Storage-Datenübertragung. Für diese Art der Datenübermittlung müssen weder das myDatalogC33x noch der Server über den logischen Inhalt der Datenblöcke Bescheid wissen. Es geht also nur darum, einen Block Daten von A nach B zu transportieren.

Die von einem myDatalogC33x an den Server übermittelten Daten sind somit frei gestaltbar. Es stehen 1023 Byte pro Datensatz zur Verfügung, die beliebig genutzt werden können. Zudem stehen auch noch 10 voneinander unabhängige Speicherblöcke für Konfigurationsdaten mit je 4000 Byte zur Verfügung, die ebenfalls beliebig genutzt werden können.

Um die von einem myDatalogC33x erhaltenen Daten und Konfigurationen auch in Verbindung mit der myDatanet Oberfläche nutzen zu können (Auswertungen, Visualisierungen, Grafiken, etc.), muss eine Beschreibung des Datenblock- bzw. Konfigurationsblock-Inhalts am Server erfolgen. Der Data Descriptor beinhaltet sowohl das Werkzeug für die Beschreibung der Daten, als auch die korrekte Bereitstellung der Daten zur Verwendung mit der Oberfläche.

14.1 Datenstruktur

Für die unterschiedlichen Arten von Daten (Messdaten, Konfigurationen usw.) stehen folgende Container zur Verfügung:

- **Messdaten:** "#histdata0" - "#histdata9"
- **Konfigurationen:** "#config0" - "#config9"
- **Nur am Server verfügbare Konfigurationen:** "#configA" - "#configC"
- **Alarmmeldungen:** #alerts
- **Aloha-Daten:** #aloha

Dieser Abschnitt beschreibt wie strukturierte Messdatenkanäle ("#histdata0" - "#histdata9"), Konfigurationsspeicherblöcke ("#config0" - "#config9" bzw. "#configA" - "#configC") sowie die Aloha-Daten ("'aloha") für die Verwendung am myDatanet-Server in einzelne Datenfelder aufgeteilt werden. Die Struktur der Alarmmeldungen ("#alerts") ist im System fest verankert und muss durch den Anwender nicht spezifiziert werden bzw. kann nicht angepasst werden.

> ***Wichtiger Hinweis:*** *Sollen ein strukturierter Messdatenkanal, ein Konfigurationsblock oder die Aloha-Daten am myDatanet-Server oder über die REST-API verfügbar sein, müssen alle Datenfelder mittels Data Descriptor definiert werden.*

Ein erweitertes Beispiel, in dem die meisten der verfügbaren Attribute verwendet werden, finden Sie im Kapitel "Beispiel" auf Seite 221.

14.1.1 Aufteilung eines strukturierten Messdatenkanals in einzelne Datenfelder

```
#histdata0 Measurements up
BatVoltage      s16  title="Battery Voltage"  decpl=2  units="V"    vscale=0.001
InputVoltage    s16  title="USB Voltage"      decpl=2  units="V"    vscale=0.001
```

Mit der ersten Zeile im obigen Beispiel wird der für die Messdaten zu verwendende Container spezifiziert:

#histdata0:	Die Messdaten sollen im histdata-Kanal 0 gespeichert werden.
Measurements:	"Measurements" soll als Name für den histdata-Kanal verwendet werden.
up:	Die Daten werden nur vom Gerät zum Server übertragen.

Hinweis: *Nach Angabe der Übertragungsrichtung könnten noch weitere Attribute (z.B. "title") angefügt werden.*

Die zweite Zeile im obigen Beispiel beschreibt den ersten Messwert im verwendeten Messdaten-Container:

BatVoltage:	"BatVoltage" soll als Name für den Messwert verwendet werden.
s16:	Als Datentyp für den Messwert soll ein 16Bit signed Integer verwendet werden.
title:	Bezeichnung des Messwerts, die am Server angezeigt wird
decpl:	Anzahl der anzuzeigenden Dezimalstellen
units:	Einheit des Messwertes, die am Server angezeigt wird
vscale:	Virtuelle Skalierung des Werts (siehe "Attribute der Feld-Definition" auf Seite 216)

Hinweis: *Name und Datentyp müssen immer angegeben werden. Attribute sind optional. Es können noch weitere Attribute angeführt werden.*

Die dritte Zeile im obigen Beispiel beschreibt den zweiten Messwert im verwendeten Messdaten-Container.

14.1.2 Aufteilung eines Konfigurationsspeicherblocks in einzelne Datenfelder

```
#config0 BasicCfg down title="Basic configuration"
RecordItv        u32  title="Record Interval"        units="sec"  min=10  default=10
TransmissionItv  u32  title="Transmission Interval"  units="min"  min=10  default=60
```

Mit der ersten Zeile im obigen Beispiel wird der für die Konfiguration zu verwendende Container spezifiziert:

#config0:	Die Parameter sollen im Konfigurationsspeicherblock 0 gespeichert werden.
BasicCfg:	"BasicCfg" soll als Name für den Konfigurationsspeicherblock verwendet werden.
down:	Der Konfigurationsspeicherblock wird nur vom Server zum Gerät übertragen.
title:	Bezeichnung für den Konfigurationsabschnitt, die am Server angezeigt wird

> ***Hinweis:*** *Name und Übertragungsrichtung müssen immer angegeben werden. Attribute sind optional. Es können noch weitere Attribute (z.B. "edit" oder "view") angeführt werden.*

Die zweite Zeile im obigen Beispiel beschreibt den ersten Parameter im Konfigurationsspeicherblock:

RecordItv:	"RecordItv" soll als Name für den Parameter verwendet werden
u32:	Als Datentyp für den Parameter soll ein 32Bit unsigned Integer verwendet werden.
title:	Bezeichnung des Parameters, die am Server angezeigt wird
units:	Einheit des Parameters, die am Server angezeigt wird
min:	kleinster gültiger Wert für den Parameter
default:	Defaultwert für den Parameter

> ***Hinweis:*** *Name und Datentyp müssen immer angegeben werden. Attribute sind optional. Es können noch weitere Attribute (z.B. "vscale") angeführt werden.*

Die dritte Zeile im obigen Beispiel beschreibt den zweiten Parameter im Konfigurationsspeicherblock.

14.1.3 Aufteilung der Aloha-Daten in einzelne Datenfelder

```
#aloha up
BatVoltage      s16  title="Battery Voltage"  decpl=2  units="V"    vscale=0.001
InputVoltage    s16  title="USB Voltage"      decpl=2  units="V"    vscale=0.001
```

Mit der ersten Zeile im obigen Beispiel wird der für die Aloha-Daten zu verwendende Container spezifiziert:

#aloha:	Die Messwerte sollen im Aloha-Daten-Container gespeichert werden.
up:	Die Daten werden nur vom Gerät zum Server übertragen.

Hinweis: *Nach Angabe der Übertragungsrichtung könnten noch weitere Attribute (z.B. "title") angefügt werden.*

Die zweite Zeile im obigen Beispiel beschreibt den ersten Messwert im verwendeten Aloha-Daten-Container:

BatVoltage:	"BatVoltage" soll als Name für den Messwert verwendet werden.
s16:	Als Datentyp für den Messwert soll ein 16Bit signed Integer verwendet werden.
title:	Bezeichnung des Messwerts, die am Server angezeigt wird
decpl:	Anzahl der anzuzeigenden Dezimalstellen
units:	Einheit des Messwertes, die am Server angezeigt wird
vscale:	Virtuelle Skalierung des Werts (siehe "Attribute der Feld-Definition" auf Seite 216).

Hinweis: *Name und Datentyp müssen immer angegeben werden. Attribute sind optional. Es können noch weitere Attribute angeführt werden.*

Die dritte Zeile im obigen Beispiel beschreibt den zweiten Messwert im verwendeten Aloha-Daten-Container.

14.1.4 Attribute der Feld-Definition

title

Alphanumerisch. Titel des Feldes. Diese Bezeichnung findet sich danach in allen Auswertungen, Grafiken, etc. für diesen Kanal.
Die Länge von title sollte 16 Zeichen nach Möglichkeit nicht überschreiten, da sonst Darstellungsprobleme an der Oberfläche auftreten können.

units

Alphanumerisch. Einheiten des Werts.
Die Länge von units sollte 8 Zeichen nach Möglichkeit nicht überschreiten, da sonst Darstellungsprobleme an der Oberfläche auftreten können.

bitmask

Hexadezimal, ohne führendes "0x". Bitmaske, um die tatsächlich zu verwendenden Bits aus dem Datenblock heraus zu maskieren, hexadezimal. Nach der Maskierung wird der extrahierte Wert auf das LSB ausgerichtet (LSB-aligned).
Beispiel: Ein u16-Feld wird aus zwei Bytes mit dem Inhalt 0xF3A7 erzeugt. Als Bitmaske wird 0FF0 angegeben. Die Bits werden extrahiert und ans LSB ausgerichtet. Der resultierende HEX-Wert ist also 0x3A (=58 dezimal).

editmask

Formatanweisung für die Anzeige des Feldinhalts auf der Oberfläche des myDatanet-Servers bzw. die Eingabe über die Oberfläche des myDatanet-Servers

- *verwendbar für Strings (Datentyp "astr", "nstr", "wstr" und "ustr", jedoch nicht für "cstr")*

Formatanweisung	***Erklärung***
"%COLORPICKER%"	*Erstellt einen Button, der die aktuell ausgewählte Farbe anzeigt. Beim Klicken des Buttons wird ein Dialogfeld eingeblendet, über das die gewünschte Farbe festgelegt werden kann. Das Dialogfeld liefert die gewählte Farbe als String im Format "RRGGBB", wobei die Farbanteile jeweils in Hex angegeben sind.*
"%HEX%"	*Erstellt ein Eingabefeld, in dem der String im Hex-Format angezeigt wird. Jedes Byte des Stings wird dabei durch zwei Zeichen repräsentiert. Der Buchstabe 'a' wird z.B. als "61" dargestellt*
"%MASKED%"	*Erstellt ein Eingabefeld, dessen Inhalt mit "Sternchen" maskiert wird (z.B. für Passwörter und Tokens). Neben dem Eingabefeld wird ein Button (in Form eines Auges) zum Umschalten zwischen Klartext und Maskierung eingeblendet.*
"%MULTILINE%30%75"	*Erstellt ein Textfeld mit 30 Zeilen und je 75 Zeichen pro Zeile*

- *verwendbar für numerische Felder (Datentyp "u8", "s8", "f32", ...)*

Formatanweisung	*Erklärung*
"%2.x0"	*Erstellt ein Eingabefeld, in dem der numerische Wert im Hex-Format angezeigt wird. Jedes Byte des Datentyps wird dabei durch zwei Zeichen repräsentiert. Die Zahl nach dem "%" muss passend zur erforderlichen Zeichenanzahl für die Repräsentation des Datentyps angegeben werden (z.B. '4' für einen "u16").* ***Hinweis:** Bei Verwendung dieser Formatanweisung muss auch das Attribut "decpl" auf "-1" gesetzt werden (d.h. decpl=-1).*
"0=off;1=on"	*Es wird eine Dropdown-Liste erstellt, in der für jeden Eintrag anstelle des Wertes der Text nach dem "=" angezeigt wird. Die Einträge sind durch ";" zu trennen.* ***Hinweis:** Die Einträge dürfen NICHT mit '+' oder '~' beginnen*
"%CHECKBOX%"	*Es wird eine Checkbox erstellt.* *1 = Häkchen gesetzt* *ungleich 1 = Häkchen nicht gesetzt* ***Hinweis:** Wird die Checkbox zur Datenanzeige genutzt und das Gerät liefert einen Wert ungleich 1, wird das Häkchen auch als "nicht gesetzt" angezeigt. Wird die Checkbox zur Dateneingabe genutzt und ist das Häkchen nicht gesetzt, liefert die Checkbox 0.*
"%CHECKBOX%5;10"	*Wie bei editmask="%CHECKBOX%" wird eine Checkbox erstellt, nur dass hier die Werte für "Häkchen gesetzt" (10 in diesem Beispiel) und "Häkchen nicht gesetzt" (5 in diesem Beispiel bzw. ungleich 10) selbst gewählt werden können. Der oben angeführte Hinweis gilt auch hier in ähnlicher Form.*
"%TIME%s%hh:mm:ss"	*Erstellt ein Eingabefeld, in dem der numerische Wert im angegebenen Zeitformat (z.B. hh:mm:ss) angezeigt wird.* *Nach "%TIME" muss angegeben werden, ob der Wert in Sekunden (%s) oder in Minuten (%m) gespeichert wird. Nach dieser Angabe folgt das Zeitformat. Möglich sind "%hh:mm:ss", %hh:mm" oder "%mm:ss". Achtung "%mm:ss" ist nicht zulässig, wenn der Wert in Minuten (%m) gespeichert werden soll.* ***Hinweis:** Es empfiehlt sich über das "units" Attribut anzugeben, in welchem Zeitformat (z.B. hh:mm:ss) der Wert dargestellt wird.*

- *verwendbar für Zeitstempel (Datentyp "stamp32" und "stamp40")*

Formatanweisung	Erklärung
%DATETIMEPICKER%	Erstellt ein Eingabefeld, in dem der Zeitstempel als Datum/Zeit angezeigt wird. Beim Klicken auf das Eingabefeld wird ein Dialogfeld eingeblendet, über das Datum und Zeit ausgewählt werden können.

decpl

Numerisch, ganzzahlig positiv. Anzahl der anzuzeigenden Dezimalstellen.

vscale

Numerisch, Fließkomma. Virtuelle Skalierung des Werts. Der extrahierte Wert wird mit diesem Faktor multipliziert und dann erst weiterverwendet.
*Dieser Wert stellt den Wert k aus der Formel k*x+d dar.*

vofs

Numerisch, Fließkomma. Virtueller Offset des Werts. Der extrahierte Wert wird erst mit vscale multipliziert, danach wird vofs zum Wert addiert.
*Dieser Wert stellt den Wert d aus der Formel k*x+d dar.*

min

Der Minimalwert für die weitere Darstellung am Server (z.B. Grafik).

max

Der Maximalwert für die weitere Darstellung am Server (z.B. Grafik). Beim Datentyp string (bzw. char) wird dieser Wert für die Länge der Zeichenfolge herangezogen. Damit ist die Angabe des max-Wertes bei string (bzw. char) verpflichtend.

chmode

Der Kanalmodus. Die hier gewählte Einstellung nimmt Einfluss auf die weitere Verarbeitung und Darstellung des Kanals in den einzelnen Servermodulen.
Es sind folgende Kanalmodi verfügbar:

Modus	Name	Erklärung
1	Digital	Der Kanal wird als Digitalkanal verarbeitet. Um Probleme zu vermeiden, sollte der enthaltene Wert 0 oder 1 sein.
2	Tageszähler	ein Zähler, dessen Wert einmal täglich zurückgesetzt wird. Diese Rücksetzung ist durch das Steuerprogramm vorzunehmen!
3	Intervallzähler	Ein Zähler, der bei jeder Erstellung eines Messdatensatzes zurückgesetzt wird. Diese Rücksetzung ist durch das Steuerprogramm vorzunehmen!
6 (Standard)	Analog	ein "einfacher" Messwert, z.B. Temperatur
12	Endloszähler	ein Zähler, dessen Wert nie zurückgesetzt wird, z.B. Wasserzähler, Stromzähler

index

wird für die benutzerdefinierte Sortierung der Felder in den Auswahllisten verwendet. Der Standardwert für die 1000 verfügbaren Kanäle ist -1, dadurch wird der Kanal ausgeblendet.
Sobald ein Kanal in der Datenstruktur-Beschreibung verwendet wird (Es reicht ein einfacher Field-Tag mit dem Attribut name), wird der Wert Index im Hintergrund automatisch auf den Feld-Index gesetzt (Bsp.: bei ch2 wird Index=2).
Diese Standard-Sortierung kann durch die Angabe des Index-Feldes übersteuert werden.

view

Numerisch, ganzzahlig, positiv. Gibt an, ab welchem Benutzerlevel das Feld auf der Oberfläche des myDatanet-Servers sichtbar ist.

edit

Numerisch, ganzzahlig, positiv. Gibt den Benutzerlevel an, der erforderlich ist, um den Feldinhalt über die Oberfläche des myDatanet-Servers verändern zu können. Wird dieses Attribut nicht angegeben oder ist der angegebene Wert kleiner als jener für das Attribut "view", ist zum Ändern des Feldinhalts der für das Attribut "view" angegebene Benutzerlevel erforderlich.
Soll ein Ändern des Feldinhalts über die REST-API verhindert werden, muss der Wert dieses Attributs auf 99 gesetzt werden.

14.2 Beispiel

```
#histdata0 Measurements up
Delay  u16  title="Delay"           units="sec" min=10  max=2000 vofs=10      chmode=3  index=1
Height f32  title="Height"  decpl=2  units="cm"  min=0   max=2000 vscale=0.01  chmode=6  index=0
Pump   u8  view=99  edit=99
@Pump
Pump_MSK  u8  dlorw=skip  title="Pump"  bitmask=$01  min=0  max=1              chmode=1
Info      astr.50         title="Info"                                                  index=10
```

Mit der ersten Zeile wird der für die Messdaten zu verwendende Container spezifiziert:

#histdata0:	Die Messdaten sollen im histdata-Kanal 0 gespeichert werden.
Measurements:	"Measurements" soll als Name für den histdata-Kanal verwendet werden
up:	Die Daten werden nur vom Gerät zum Server übertragen

Die zweite Zeile beschreibt den ersten Messwert "Delay" im verwendeten Messdaten-Container:

Delay:	"Delay" soll als Name für den Messwert verwendet werden.
u16:	Als Datentyp für den Messwert soll ein 16Bit unsigned Integer (d.h. 2 Byte) verwendet werden.
title:	Bezeichnung des Messwerts, die am Server angezeigt wird
units:	Einheit des Messwertes, die am Server angezeigt wird
min:	Minimalwert für die weitere Darstellung am Server (z.B. Grafik)
max:	Maximalwert für die weitere Darstellung am Server (z.B. Grafik)
vofs:	virtueller Offset des Werts (siehe "Attribute der Feld-Definition" auf Seite 216). Beim aktuellen Beispiel wird zum extrahierten Wert 10 addiert.
chmode:	Kanalmodus 3 =^ Intervallzähler (Zähler, der bei jeder Erstellung eines Messdatensatzes zurückgesetzt wird)
index:	wird für die benutzerdefinierte Sortierung der Felder in den Auswahllisten verwendet

Die dritte Zeile beschreibt den zweiten Messwert "Height" im verwendeten Messdaten-Container:

Height:	"Height" soll als Name für den Messwert verwendet werden.
f32:	Als Datentyp für den Messwert soll ein 32Bit Float (d.h. 4 Bytes) verwendet werden.
title:	Bezeichnung des Messwerts, die am Server angezeigt wird
decpl:	Anzahl der anzuzeigenden Dezimalstellen
units:	Einheit des Messwertes, die am Server angezeigt wird
min:	Minimalwert für die weitere Darstellung am Server (z.B. Grafik)

max:	Maximalwert für die weitere Darstellung am Server (z.B. Grafik)
vscale:	virtuelle Skalierung des Werts (siehe "Attribute der Feld-Definition" auf Seite 216). Beim aktuellen Beispiel wird der extrahierte Wert mit 0.01 multipliziert.
chmode:	Kanalmodus 6 =^ Analogkanal (ein "einfacher" Messwert, z.B. Temperatur)
index:	wird für die benutzerdefinierte Sortierung der Felder in den Auswahllisten verwendet

Die Zeilen 4-6 beschreiben den dritten Messwert "Pump" im verwendeten Messdaten-Container:

Pump:	"Pump" soll als Name für den Messwert verwendet werden.
u8:	Als Datentyp für den Messwert soll ein 8Bit unsigned Integer (d.h. 1Byte) verwendet werden.
view:	gibt an, ab welchem Benutzerlevel das Feld auf der Oberfläche des Servers sichtbar ist 99 =^ Feld ist für niemanden sichtbar (nötig, da im aktuellen Beispiel das Schattenfeld "Pump_MSK" anstelle des Feldes "Pump" verwendet werden soll. Mittels Schattenfeldern lässt sich beispielsweise ein Byte-Feld auf mehrere Bit-Felder aufteilen).
edit:	gibt den Benutzerlevel an, der erforderlich ist, um den Feldinhalt über die Oberfläche des Servers verändern zu können 99 =^ Feld kann durch niemanden verändert werden (nötig, da im aktuellen Beispiel das Schattenfeld "Pump_MSK" anstelle des Feldes "Pump" verwendet werden soll).
@Pump:	gibt an, dass das mit Zeile 6 definierte Schattenfeld "Pump_MSK" den Speicherbereich des mit Zeile 4 definierten Feldes "Pump" verwenden soll
Pump_MSK:	"Pump_MSK" soll als Name für das Schattenfeld verwendet werden.
dlorw=skip:	Bei den für die Device Logic automatisch erzeugten Zugriffsfunktionen auf den Container (im aktuellen Beispiel histdata-Kanal 0) wird das Schattenfeld nicht berücksichtigt. D.h. innerhalb der Device Logic muss das Schreiben/Lesen des Schattenfeldes durch manuelles Maskieren des gewünschten Bits im Feld "Pump" erfolgen.
bitmask:	Bitmaske, um die tatsächlich zu verwendenden Bits aus dem Datenblock heraus zu maskieren, hexadezimal Im aktuellen Beispiel wird das niedrigstwertige Bit (LSB) aus dem Feld "Pump" heraus maskiert.
min:	Minimalwert für die weitere Darstellung am Server (z.B. Grafik)
max:	Maximalwert für die weitere Darstellung am Server (z.B. Grafik)
chmode:	Kanalmodus. 1 =^ Digital

Die 7te Zeile beschreibt den vierten Messwert "Info" im verwendeten Messdaten-Container:

Info:	"Info" soll als Name für den Messwert verwendet werden.
astr.50:	Als Datentyp für den Messwert soll ein ANSI-String verwendet werden. Nach dem Punkt wird die Anzahl der Zeichen (50 im aktuellen Beispiel) angegeben.
title:	Bezeichnung des Messwerts, die am Server angezeigt wird
index:	wird für die benutzerdefinierte Sortierung der Felder in den Auswahllisten verwendet.

Bei "Pump" fehlt die Angabe von index, daher erhält dieser Kanal automatsich den Index 2. Die Sortierung der Kanäle ist also "Height", "Delay", "Pump", "Info".

14.3 Spezialwerte der Datentypen

Jeder numerische Datentyp unterstützt spezielle Zustände wie z.B. NaN (Not a Number). Wird solch ein Wert am Server erkannt, so wird die in myDatanet übliche Anzeige und Weiterverarbeitung angewandt.

Übersicht der möglichen Werte (unsigned):

Wert/Typ	u8 (byte)	u16 (word)	u32 (dword)
NaN	0xFF	0xFFFF	0xFFFFFFFF
OF	0xFE	0xFFFE	0xFFFFFFFE
UF	0xFD	0xFFFD	0xFFFFFFFD
OL	0xFC	0xFFFC	0xFFFFFFFC
SC	0xFB	0xFFFB	0xFFFFFFFB

Übersicht der möglichen Werte (signed):

Wert/Typ	s8 (bint)	s16(wint)	s32 (dint)	s64 (qint)
NaN	127	32767	2147483647	0x7FFFFFFFFFFFFFFF
OF	126	32766	2147483646	0x7FFFFFFFFFFFFFFE
UF	-126	-32766	-2147483646	0x8000000000000002
OL	-127	-32767	-2147483647	0x8000000000000001
SC	-128	-32768	-2147483648	0x8000000000000000

Übersicht der möglichen Werte (float):

Wert/Typ	f32 (float32)	f64 (float64)
NaN	0x7F800001	0x7FF0000000000001
OF	0x7F800002	0x7FF0000000000002
UF	0x7F800003	0x7FF0000000000003
OL	0x7F800004	0x7FF0000000000004
SC	0x7F800005	0x7FF0000000000005

Kapitel 15 API

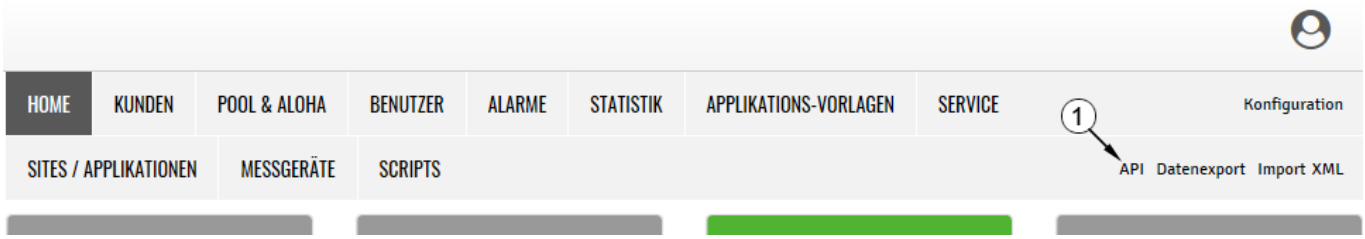
Wichtiger Hinweis: *Für die Verwendung der API (Application Programming Interface) sind die entsprechenden Lizenzen am myDatanet-Server erforderlich. Für nähere Informationen wenden Sie sich an Ihren zuständigen Vertriebspartner.*

15.1 Allgemein

Die API dient dazu, Daten aus dem myDatanet-Server zu exportieren sowie Daten in den myDatanet-Server zu importieren. Dies beschränkt sich jedoch nicht nur auf die reinen Messdaten sondern auf alle durch den myDatanet-Server bereitgestellten Daten (z.B. Konfigurationen). Dadurch ist es dem Kunden möglich, komplett auf die Oberfläche des myDatanet-Servers zu verzichten und seine eigene Benutzerschnittstelle zu erstellen. Dies kann zum Beispiel durch ein eigens entwickeltes PC-Programm oder ein Web-Interface erfolgen.

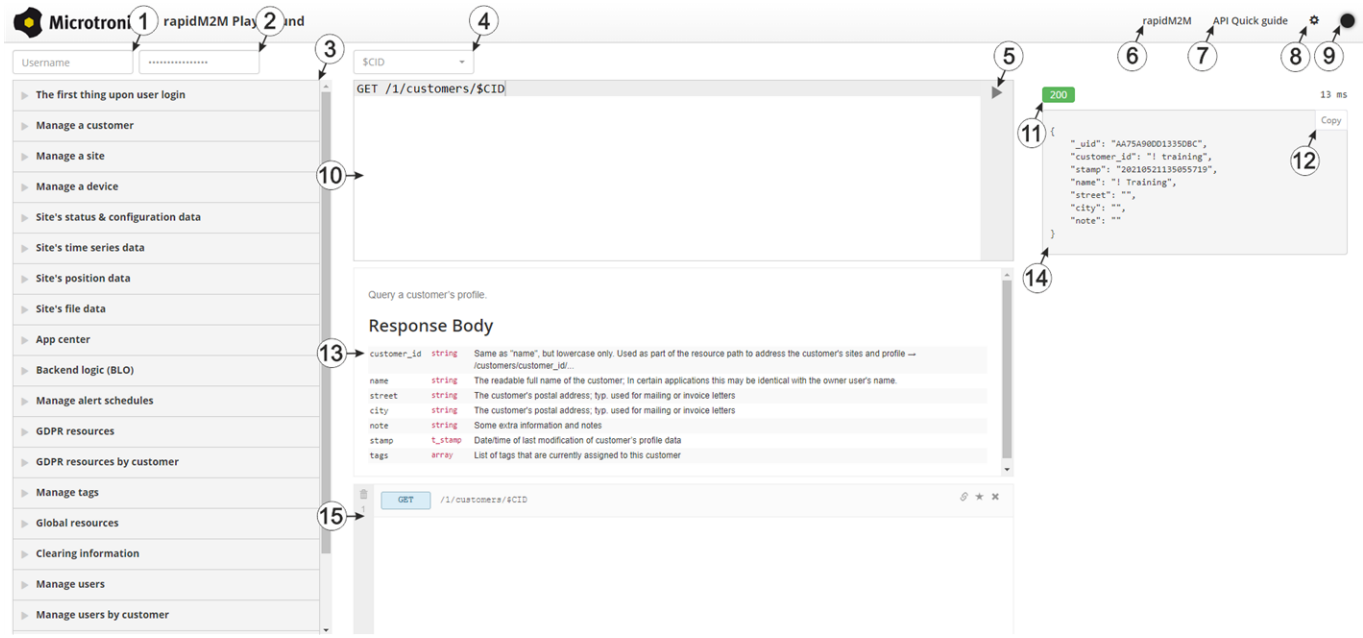
15.2 rapidM2M Playground

Der rapidM2M Playground ermöglicht es Ihnen, sich mit der API des myDatanet-Servers vertraut zu machen und die bereitgestellten Funktionen zu testen. Durch einen Klick auf die Schaltfläche "API" gelangen Sie zum rapidM2M Playground .



1	öffnet den rapidM2M Playground

15.2.1 Übersicht



rapidM2M Playground

1	Eingabefeld für den Benutzernamen
2	Eingabefeld für das Passwort
3	Auflistung der zur Verfügung stehenden HTTP-Kommandos. Die HTTP-Kommandos sind entsprechend ihrer Anwendungsgebiete gruppiert.
4	Abhängig vom gewählten HTTP-Kommando werden hier die Dropdown-Listen für die Auswahl des Kunden, des Benutzers und der Site eingeblendet, die die entsprechende Wildcard ("$CID"...Kunde , "$UID"...Benutzer, "$SID"...Site) im Ressource-Pfad des HTTP-Kommandos ersetzen sollen.
5	Button zum Ausführen des HTTP-Kommandos
6	öffnet die Webseite "http://rapidm2m.com/", die zusätzliche Informationen für Entwickler enthält
7	öffnet die Kurzanleitung für die API
8	Button zum Anzeigen des Menüs, das die globalen Einstellungen enthält
9	Button zum Wechsel des Farbschemas des rapidM2M Playground
10	Fenster, in dem das gewählte HTTP-Kommando angezeigt wird
11	Response-Code, der vom myDatanet-Server als Antwort auf das HTTP-Kommando gesendet wurde
12	kopiert das JSON-Objekt, das als Antwort auf das HTTP-Kommando erzeugt wurde, in die Zwischenablage
13	Fenster, in dem die Dokumentation für das ausgewählte HTTP-Kommando angezeigt wird. Diese enthält abhängig vom ausgewählten Kommando eine Beschreibung der Aktion, die durchgeführt wird, Hinweise, die beachtet werden müssen und eine Beschreibung des Request Bodys sowie des Response Bodys.
14	Fenster, in dem das JSON-Objekt angezeigt wird, das als Antwort auf das HTTP-Kommando erzeugt wird
15	Fenster, in dem die zuletzt ausgeführten HTTP-Kommandos angezeigt werden

Kapitel 16 Wartung

Wichtiger Hinweis: *Um Schäden am Instrument zu vermeiden, dürfen die in diesem Abschnitt der Anleitung beschriebenen Arbeiten nur von qualifiziertem Personal ausgeführt werden.*

Vor Wartungs-, Reinigungs- und/oder Reparaturarbeiten ist das Gerät unbedingt spannungsfrei zu machen.

16.1 Allgemeine Wartung

- Überprüfen Sie den myDatalogC33x regelmäßig auf mechanische Beschädigungen.
- Überprüfen Sie regelmäßig alle Kabel auf mechanische Beschädigungen.
- Reinigen Sie den myDatalogC33x mit einem weichen, feuchten Tuch. Verwenden Sie ein mildes Reinigungsmittel, falls nötig.

16.2 Sicherungswechsel

GEFAHR:
Feuergefahr. Eine falsche Sicherung kann Verletzungen, Schäden oder Immissionen verursachen. Die Sicherung befindet sich im Inneren des Gehäuses, das nur vom Hersteller geöffnet werden darf.

Sollte der Verdacht bestehen, dass die Sicherung des myDatalogC33x defekt ist (siehe "Fehlersuche und Behebung" auf Seite 231), muss das Gerät in der Originalverpackung an den Hersteller zurückgesendet werden (siehe "Rücksendung" auf Seite 42).

Kapitel 17 Demontage/Entsorgung

Durch falsche Entsorgung können Gefahren für die Umwelt entstehen.

Entsorgen Sie Gerätekomponenten und Verpackungsmaterialien entsprechend den gültigen örtlichen Umweltvorschriften für Elektroprodukte.

1. Trennen Sie die eventuell verwendete Ladespannung.
2. Lösen Sie eventuell angeschlossene Kabel mit geeignetem Werkzeug.

Logo zur WEEE-Direktive der EU

Dieses Symbol weist darauf hin, dass bei der Verschrottung des Gerätes die Anforderungen der Richtlinie 2012/19/EU über Elektro- und Elektronik-Altgeräte zu beachten sind. Die Microtronics Engineering GmbH unterstützt und fördert das Recycling bzw. die umweltgerechte, getrennte Sammlung/Entsorgung von Elektro- und Elektronik-Altgeräten zum Schutz der Umwelt und der menschlichen Gesundheit. Beachten Sie die örtlichen Entsorgungsvorschriften und Gesetze.

Die Microtronics Engineering GmbH entpflichtet in Österreich in den Verkehr gebrachte Waren über die ERA, daher können in Österreich Sammelstellen, welche mit der ERA Elektro Recycling Austria GmbH (https://www.era-gmbh.at/) kooperieren, für die Entsorgung genutzt werden.

Das Gerät enthält eine fest verlötete Lithium-Knopfzelle. Diese ist vor der Entsorgung zu entfernen bzw. ist der Entsorgungsbetrieb darüber zu informieren, dass sich noch Batterien im Gerät befinden.

Das Gerät enthält eine Batterie bzw. einen Akku (Lithium), welcher separat zu entsorgen ist.

Kapitel 18 Fehlersuche und Behebung

18.1 Allgemeine Probleme

Problem	Ursache/Lösung
Das Gerät zeigt keine Reaktion	• Kabelverbindungen überprüfen (siehe "Anschluss der Sensoren, der Aktoren und der Versorgung" auf Seite 46)
Kommunikationsprobleme	• Laden Sie das Gerätelog mit Hilfe des DeviceConfig vom myDatalogC33x (siehe "Benutzerhandbuch für DeviceConfig" 206.887).
Es sind nicht alle/keine Daten am Server vorhanden.	• Es kam zu einem Verbindungsabbruch während der Übertragung, erkennbar an einem Timeout-Eintrag in der Verbindungsliste (siehe "Benutzerhandbuch für myDatanet-Server " 206.886). Lösung: Übertragung auslösen oder auf die nächste zyklische Übertragung warten. • Der Data Descriptor wurde nicht korrekt konfiguriert (siehe "Data Descriptor " auf Seite 213). • Die Zuweisung von Gerät und Site ist nicht korrekt (siehe "Site" auf Seite 74).
Daten am Universaleingang sind nicht plausibel	• Kabelverbindungen überprüfen (siehe "Anschluss der Sensoren, der Aktoren und der Versorgung" auf Seite 46) • Prüfen Sie, ob das Ausgabesignal des von Ihnen verwendeten Sensors kompatibel mit den elektrischen Kenndaten der Universaleingänge ist (siehe "Technische Details zu den Universaleingängen" auf Seite 59). • Der Data Descriptor wurde nicht korrekt konfiguriert (siehe "Data Descriptor " auf Seite 213).
Die Daten der RS485 Schnittstelle sind nicht plausibel.	• Kabelverbindungen überprüfen (siehe "Anschluss der Sensoren, der Aktoren und der Versorgung" auf Seite 46) • Prüfen Sie, ob der von Ihnen verwendete Sensor kompatibel mit den elektrischen Kenndaten der Schnittstelle ist (siehe "Technische Details zur RS485-Schnittstelle" auf Seite 60). • Der Data Descriptor wurde nicht korrekt konfiguriert (siehe "Data Descriptor " auf Seite 213).
Die über die CAN-Schnittstelle empfangenen Daten sind nicht plausibel	• Kabelverbindungen überprüfen (siehe "Anschluss der Sensoren, der Aktoren und der Versorgung" auf Seite 46) • Prüfen Sie, ob die von Ihnen verwendeten Maschinen, Sensoren und Aktoren kompatibel mit den Kenndaten der CAN-Schnittstelle sind (siehe "Technische Details zur CAN-Schnittstelle" auf Seite 61). • Der Data Descriptor wurde nicht korrekt konfiguriert (siehe "Data Descriptor " auf Seite 213).
Die Daten der RS232 Schnittstelle sind nicht plausibel.	• Kabelverbindungen überprüfen (siehe "Anschluss der Sensoren, der Aktoren und der Versorgung" auf Seite 46) • Prüfen Sie, ob der von Ihnen verwendete Sensor kompatibel mit den elektrischen Kenndaten der Schnittstelle ist (siehe "Technische Details zur RS232-Schnittstelle" auf Seite 63). • Der Data Descriptor wurde nicht korrekt konfiguriert (siehe "Data Descriptor " auf Seite 213).

Problem	Ursache/Lösung
Der potentialfreie Schaltkontakt funktioniert nicht.	• Ausfall der Spannung, die über den potentialfreien Schaltkontakt geführt wird
Die Device Logic wird nicht korrekt ausgeführt.	• Prüfen Sie, ob bei der Konfiguration der Steuerung der korrekte Device Logic Typ ausgewählt wurde (siehe "Steuerung" auf Seite 75). • Laden Sie das Gerätelog mit Hilfe des DeviceConfig vom myDatalogC33x (siehe "Benutzerhandbuch für DeviceConfig" 206.887). Eine Auflistung aller möglichen Fehlercodes der Device Logic finden Sie im Kapitel „Pawn Script Fehlercodes“ (siehe "Device Logic Fehlercodes" auf Seite 186). • Durch einen Kontextwechsel (Zuweisung einer anderen Site / Applikation) wurde die bisherige Device Logic durch jene der neu zugewiesenen Site / Applikation ersetzt. • Durch die Zuweisung einer neuen oder anderen Site / Applikation wurde die per USB eingespielte Device Logic durch jene der neu zugewiesenen Site / Applikation ersetzt.

18.2 Log-Einträge und Fehlercodes

Log-Eintrag		Parameter		Beschreibung
Code	**Klartext**	**Code**	**Klartext**	
1000	POWER ON	0	---	Neustart nach einem Spannungsausfall
		4	---	Watchdog Reset (z.B. aufgrund einer Exception)
		6	---	Reset wurde vom Gerät selbst ausgelöst (z.B. bei Firmwareupdate)
		##	--	Neustart aus einem anderen Grund. Sollte der "POWER ON" Log-Eintrag mehrmals mit einem Parameter-Code ungleich 0 oder 6 im Gerätelog enthalten sein, liegt unter Umständen ein Hardwareproblem vor. Kontaktieren Sie in diesem Fall den Hersteller (siehe "Kontaktinformationen" auf Seite 253).
1030	UV LOCKOUT	---	---	Das Gerät schaltet aufgrund einer zu niedrigen Akku-/Batteriespannung in den Energiesparmodus und stellt alle Operationen ein. Nur die Laderegelung, falls vorhanden, bleibt aktiv.
1031	UV RECOVER	---	---	Die Akku-/Batteriespannung reicht wieder aus, um einen sicheren Betrieb zu gewährleisten. Das Gerät nimmt nun den normalen Betrieb entsprechend der Konfiguration wieder auf.
1034	CONTROLLER UPDATE	##	---	Update der Firmware des Controllers wurde erfolgreich durchgeführt Dieser Eintrag ist immer doppelt im Gerätelog enthalten. Beim ersten Eintrag gibt der Parameter die Hauptversionsnummer (z.B. 3 bei 03v011) und beim zweiten Eintrag die Nebenversionsnummer (z.B. 11 bei 03v011) an.
1035	EXCEPTION	##	---	Es wurde ein interner Systemfehler erkannt, der zu einem Neustart des Geräts führte. Der Parameter gibt den Typ des Systemfehlers an. Sollte dieser Fehler mehrmals mit demselben Parameter-Code im Gerätelog enthalten sein, kontaktieren Sie den Hersteller (siehe "Kontaktinformationen" auf Seite 253).
1038	UV MODEM LOCKOUT	---	---	Das Gerät deaktiviert aufgrund einer zu niedrigen Akku-/Batteriespannung das Modem. Das Herstellen einer Verbindung ist nicht mehr möglich.

Log-Eintrag		Parameter		Beschreibung
Code	**Klartext**	**Code**	**Klartext**	
1039	UV MODEM RECOVER	---	---	Die Akku-/Batteriespannung reicht wieder aus, um eine stabile Verbindung herzustellen.
1161	LOG REFORMATFILE	##	---	Fehler im Filesystem wurden behoben. Es kann dabei zum Datenverlust (Daten und/oder Log-Einträge) kommen. Der Parameter enthält nähere Informationen zu dem Problem. Sollte dieser Fehler mehrmals mit demselben Parameter-Code im Gerätelog enthalten sein, kontaktieren Sie den Hersteller (siehe "Kontaktinformationen" auf Seite 253).
1192	FUTURE TIMESTAMP	##	---	interner Fehler Sollte dieser Fehler mehrmals im Gerätelog enthalten sein, kontaktieren Sie den Hersteller (siehe "Kontaktinformationen" auf Seite 253).
1200	MODEM ERROR			Modemfehler (siehe "Modemfehler" auf Seite 238)
1201	MODEM NOT FOUND	---		interner Fehler Sollte dieser Fehler mehrmals im Gerätelog enthalten sein, kontaktieren Sie den Hersteller (siehe "Kontaktinformationen" auf Seite 253).
1202	MODEM CMME ERROR	##	---	Das GPRS-Modem meldet einen +CME-Fehler. Der Parameter gibt an, um welchen Fehler es sich handelt.
1203	SELECTED NETWORK	##	---	Ein neues GSM-Netzwerk wurde gewählt. Der Parameter gibt den MCC (Mobile Country Code) und den MNC (Mobile Network Code) des gewählten GSM-Netzwerks an.
1207	GSM NETWORK REGISTRATION	0	NOT REGISTERED	nicht registriert, Modem sucht derzeit keinen neuen Betreiber zur Registrierung
		1	HOME	registriert, Heimnetzwerk
		2	SEARCHING	nicht registriert, aber Modem sucht derzeit nach einem neuen Betreiber, bei dem es sich registrieren kann
		3	DENIED	Registrierung verweigert
		4	UNKNOWN	unbekannt (z. B. außerhalb der GERAN/UTRAN/E-UTRAN-Abdeckung)
		5	ROAMING	registriert, Roaming

Log-Eintrag		Parameter		Beschreibung
Code	**Klartext**	**Code**	**Klartext**	
1208	GPRS NETWORK REGISTRATION	0	NOT REGISTERED	nicht registriert, Modem sucht derzeit keinen neuen Betreiber zur Registrierung
		1	HOME	registriert, Heimnetzwerk
		2	SEARCHING	nicht registriert, aber Modem sucht derzeit nach einem neuen Betreiber, bei dem es sich registrieren kann
		3	DENIED	Registrierung verweigert
		4	UNKNOWN	unbekannt (z. B. außerhalb der GERAN/UTRAN/E-UTRAN-Abdeckung)
		5	ROAMING	registriert, Roaming
1212	ERROR MODEM IRREGULAR OFF	##	---	Zeigt eine fehlerhafte Verbindung an. Der Parameter enthält dabei einen Zähler, der angibt wie viele Verbindungen hintereinander nicht funktioniert haben.
1219	LTE NETWORK REGISTRATION	0	NOT REGISTERED	nicht registriert, Modem sucht derzeit keinen neuen Betreiber zur Registrierung
		1	HOME	registriert, Heimnetzwerk
		2	SEARCHING	nicht registriert, aber Modem sucht derzeit nach einem neuen Betreiber, bei dem es sich registrieren kann
		3	DENIED	Registrierung verweigert
		4	UNKNOWN	unbekannt (z. B. außerhalb der GERAN/UTRAN/E-UTRAN-Abdeckung)
		5	ROAMING	registriert, Roaming
1252	MODEM TO CON	##	---	Timeout während des Verbindungsaufbaus. Der Parameter gibt den Grund für den Timeout an. Sollte dieser Fehler mehrmals mit demselben Parameter-Code im Gerätelog enthalten sein, kontaktieren Sie den Hersteller (siehe "Kontaktinformationen" auf Seite 253).
1281	ZLIB STREAMPROCESS ERR	##	---	interner Fehler Sollte dieser Fehler mehrmals im Gerätelog enthalten sein, kontaktieren Sie den Hersteller (siehe "Kontaktinformationen" auf Seite 253).
1282	ZLIB STREAMFINISH ERR	##	---	interner Fehler Sollte dieser Fehler mehrmals im Gerätelog enthalten sein, kontaktieren Sie den Hersteller (siehe "Kontaktinformationen" auf Seite 253).
1300	USB CONNECTED	---	---	USB-Verbindung zu einem PC hergestellt.

Log-Eintrag		Parameter		Beschreibung
Code	**Klartext**	**Code**	**Klartext**	
1310	USB DISCONNECTED	---	---	USB-Verbindung wurde getrennt.
1335	LOG_SHT2X_ STATE	0	SHT2X SENSOR OK	Der interne Temperatur-und Luftfeuchtigkeitssensor liefert wieder gültige Werte
		1	SHT2X RH ERROR	Beim Lesen des Luftfeuchtikeitswertes vom internen Temperatur-und Luftfeuchtigkeitssensor kam es zu einen Kommunikationsfehler.
		2	SHT2X TEMP ERROR	Beim Lesen des Temperaturwertes vom internen Temperatur-und Luftfeuchtigkeitssensor kam es zu einen Kommunikationsfehler.
		3	SHT2X RH+TEMP ERROR	Beim Lesen des Messwertes vom internen Temperatur-und Luftfeuchtigkeitssensor kam es zu einen Kommunikationsfehler.
		4	SHT2X PLAUSIBILITY ERROR	Die vom internen Temperatur-und Luftfeuchtigkeitssensor empfangenen werte sind nicht plausibel (rH <0% rH oder >100% rH oder Temperatur <-40°C oder >125°C)
1336	SHT2X COM ERR	---	---	Kommunikation mit dem internen Temperatur- und Luftfeuchtigkeitssensor nicht möglich (Sensor nicht vorhanden oder defekt)
1337	SHT2X COM ERR1	---	---	Starten der internen Temperaturmessung schlug fehl
1338	SHT2X COM ERR2	---	---	Starten der internen Luftfeuchtigkeitsmessung schlug fehl
1339	SHT2X TEMP RAW	##	---	Rohwert für die Temperatur (Registerwert vom internen Temperatur-und Luftfeuchtigkeitssensor) wenn ein Plausibilitätsfehler (SHT2X PLAUSIBILITY ERROR) erkannt wurde
1340	SHT2X RH RAW	##	---	Rohwert für die Luftfeuchtikeit (Registerwert vom internen Temperatur-und Luftfeuchtigkeitssensor) wenn ein Plausibilitätsfeher (SHT2X PLAUSIBILITY ERROR) erkannt wurde

Log-Eintrag		Parameter		Beschreibung
Code	**Klartext**	**Code**	**Klartext**	
1601	SIM_STATE	0	NONE	SIM-Status wurde auf "NONE" geändert (Initialzustand).
		1	PRODUCTION	SIM-Status wurde auf "PRODUCTION" geändert (ein neu produziertes Gerät liegt auf Lager).
		2	HOT	SIM-Status wurde auf "HOT" geändert (gültiger Vertrag).
		3	COLD	SIM-Status wurde auf "COLD" geändert (Vertragsende oder Fair-Use-Verletzung).
		4	DISCARDED	SIM-Status wurde auf "DISCARDED" geändert (Gerät wurde außer Dienst gestellt).
1910	ACCU 0 E2PROM ERROR	0	---	Akku nicht verfügbar
		1	---	Ungültige Länge der Datenstruktur im EEPROM des Akkus
		2	---	Kein Ladeprofil im EEPROM vorhanden (Nur bei Li-Ion Akkus)
		3	---	Fehler beim Lesen des SoC-Wertes
		4	---	Fehler beim Schreiben des SoC-Wertes
		5	---	Die Ladeprofile der eingesetzten Akkus stimmen nicht überein (Nur bei Geräten, die das gleichzeitige Einsetzen mehrerer Akkus unterstützen)
		6	---	• Zulässige Ladezeit überschritten • Beim Neustart des Geräts wurde erkannt, dass beim aktuell eingesetzten Akku die zulässige Ladezeit bereits einmal überschritten wurde. Der Akku ist vermutlich defekt und sollte vom Hersteller überprüft werden.
2000 - 2199	MODULE ERR	##	---	Bereich für die kundenspezifischen kritischen Fehlercodes, die mittels der Funktion "rM2M_ WriteLog()" in das Gerätelog geschrieben werden können
2200 - 2399	MODULE WARNING	##	---	Bereich für die kundenspezifischen unkritischen Fehlercodes, die mittels der Funktion "rM2M_ WriteLog()" in das Gerätelog geschrieben werden können

Log-Eintrag		Parameter		Beschreibung
Code	**Klartext**	**Code**	**Klartext**	
2400 - 2599	MODULE INFO	##	---	Bereich für die kundenspezifischen Informationen über den aktuellen Betriebszustand, die mittels der Funktion "rM2M_WriteLog()" in das Gerätelog geschrieben werden können
2600 - 2799	MODULE DEBUG	##	---	Bereich für die kundenspezifischen Debug-Informationen, die mittels der Funktion "rM2M_WriteLog()" in das Gerätelog geschrieben werden können
3000 - 3099	SCRIPT ERROR	##	--	Fehlercodes der Scriptausführung (siehe "Device Logic Fehlercodes" auf Seite 186).

18.2.1 Modemfehler

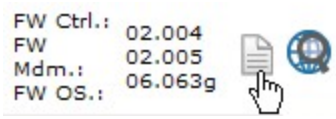
Log-Eintrag		Parameter		Beschreibung
Code	**Klartext**	**Code**	**Klartext**	
GPRS-Fehler				
1200	BEARER GPRS FAILED	-988	---	GPRS Setup-Fehler • Versuchen Sie die Antennenposition zu verbessern. • Überprüfen Sie, ob sich das Gerät im Versorgungsbereich befindet (www.microtronics.com/footprint).
1200	BAND SEL FAILED	-969	---	Es konnte weder auf dem GSM900/1800, noch auf dem GSM850/1900-Band ein Netzwerk gefunden werden. • Versuchen Sie die Antennenposition zu verbessern. • Überprüfen Sie, ob sich das Gerät im Versorgungsbereich befindet (www.microtronics.com/footprint).

Log-Eintrag		Parameter		Beschreibung
Code	Klartext	Code	Klartext	
1200	NETLOCK ERROR	-966		Fehler bei der Netzauswahl. Überprüfen Sie, ob sich das Gerät im Versorgungsbereich befindet. interner SIM-Chip: siehe www.microtronics.com/footprint
TCP Channel Fehler				
1200	CHANNEL ABORTED	-965	---	Es wird versucht auf einen/von einem nicht mehr verfügbaren TCP-Client zu schreiben/lesen. später erneut versuchen
	TCP DNS FAILURE	-958	---	Der Name konnte nicht in eine IP-Adresse aufgelöst werden. interner Fehler
	CHANNEL REFUSED	-955	---	Die TCP-Verbindung wurde vom Server abgelehnt. später erneut versuchen
	CHANNEL HOST UNREACHABLE	-954	---	keine Route zum Host später erneut versuchen
	CHANNEL NETWORK UNREACHABLE	-953	---	kein Netz erreichbar später erneut versuchen
	CHANNEL PIPE BROKEN	-952	---	TCP-Verbindung unterbrochen später erneut versuchen
	CHANNEL TIMEOUT	-951	---	Timeout (DNS-Request, TCP-Verbindung, Ping-Response,..) später erneut versuchen
	MODEM POSITION UPDATE ERROR	-943	---	Timeout bei der Ermittlung der GSM-Positionsdaten

18.3 Auswerten des Gerätelogs

18.3.1 Auswerten des Gerätelogs am myDatanet-Server

Am myDatanet-Server sind die letzten 300 Log-Einträge über den unten abgebildeten Button, der sich in der Messgeräteliste befindet, abrufbar. Da die Log-Einträge genau wie die Messdaten im Übertragungsintervall zum Server gesendet werden, sind immer nur die Log-Einträge bis zur letzten Serververbindung verfügbar.



Eine genauere Beschreibung zur Auswertung des Gerätelogs am myDatanet-Server finden Sie im Handbuch des Servers ("Benutzerhandbuch für myDatanet-Server " 206.886).

18.3.2 Auswerten des Gerätelogs mittels DeviceConfig

Mit Hilfe des Programms DeviceConfig können alle gespeicherten Logeinträge, auch jene, die noch nicht zum myDatanet-Server übertragen wurden, direkt über die USB-Schnittstelle aus des myDatalogC33x gelesen werden.

Eine genauere Beschreibung zur Auswertung des Gerätelogs mittels DeviceConfig finden Sie im Kapitel "Karteireiter "Log"" auf Seite 89

Kapitel 19 Ersatzteile und Zubehör

19.1 Montagesets

Beschreibung	Menge	Bestellnummer
Gehäuse für rapidM2M C3xx	1	301219
Wandmontageset für Gehäuse 300x200x150mm	1	301185

19.2 Antennen

Beschreibung	Menge	Bestellnummer
Antennenverlängerung SMA-M/SMA-F 2,5m	1	206.807
Flachantenne Disc SMA-M 2,5m	1	206.816
Winkeladapter SMA-M/SMA-F	1	300318
Stabantenne Multiband 2G/3G SMA-M	1	301075
Flachantenne Disc Multi Band 2xSMA-M 2m	1	301090
Kuppelantenne Multiband SMA-M 3m	1	301212
Kombiantenne Cellular/Wifi SMA-M 3m	1	300924

19.3 Versorgung

Beschreibung	Menge	Bestellnummer
Netzgerät 24V 0,63A für Hutschienenmontage [1)]	1	301066
Netzgerät 24V 1,5A 36W für Hutschienenmontage	1	301251

[1)] nicht empfohlen bei Verwendung von Erweiterungsmodulen

19.4 Adapter

Beschreibung	Menge	Bestellnummer
Gender changer 9pol. D-Sub male/male	1	206.684
Nullmodemadapter 9pol. D-Sub female/male [1)]	1	206.686

[1)] Bitte beachten Sie, dass der Pin 9 des Adapters nicht durchgeschliffen wird.

19.5 Erweiterungsmodule

Beschreibung	Menge	Bestellnummer
Eingangserweiterungen		
myDatalogC3e 12UI/2Rel	1	301058
Ausgangserweiterungen		
myDatalogC3e 3mA/6Rel	1	301112

19.6 Sonstiges Zubehör

Beschreibung	Menge	Bestellnummer
myDatanet Tool Pen	1	206.646

Kapitel 20 Dokumentenhistorie

Rev.	Datum	Änderungen
01	21.08.2020	**Erste Version**
02 (1/2)	25.03.2021 (1/2)	**Kapitel "Konformitätserklärung" auf Seite 11** *Konformitätserklärung aktualisiert* **Kapitel "Technische Daten" auf Seite 13** *Angabe des systembedingten Overheads pro Datensatz von 10Bytes auf 11Bytes angepasst* *Angabe der Bandbreite der WiFi-Kanäle hinzugefügt* *Angabe der unterstützten WiFi-Kanäle hinzugefügt* **Kapitel "Registrierungsspeicherblöcke" auf Seite 36** *Die Erklärungen der Registry-Einträge "latestAppVersion", "installedAppVersion" und "appId", die in Verbindung mit den Applikations-Vorlagen stehen, wurden entfernt. Die Applikations-Vorlagen werden nicht mehr weiterentwickelt sondern durch die IoT Apps ersetzt.* *Erklärungen für die Registry-Einträge "pipAppId" und "pipAppVer" wurden an die neue Verwendung im Zusammenhang mit den IoT Apps angepasst* **Kapitel "Site" auf Seite 74** *Erklärung des Feldes "Applikation Version" hinzugefügt, welches die Versionsnummer der IoT Applikation angibt, die aktuell auf der Site installiert ist* **Kapitel "Alarmierung" auf Seite 76** *Erklärung des Konfigurationsparameters "Offline Alarm nach" hinzugefügt* **Kapitel "DeviceConfig " auf Seite 81** *Kapitel hinzugefügt* **Kapitel "Übersicht" auf Seite 93** *Kapitel hinzugefügt* **Kapitel "Bereich "Kunden"" auf Seite 94** *Kapitel hinzugefügt* **Kapitel "Bereich "Sites / Applikationen" auf Kundenebene" auf Seite 96** *Kapitel hinzugefügt* **Kapitel "Anlegen der Site" auf Seite 97** *Der Verweis auf das Benutzerhandbuch des Servers wurde durch eine detaillierte Beschreibung der durchzuführenden Schritte ersetzt.* **Kapitel "rapidM2M Studio " auf Seite 101** *Detailliertere Aufschlüsselung der Bestandteile des rapidM2M Studio hinzugefügt* **Kapitel "Projekt Dashboard " auf Seite 103** *Kapitel hinzugefügt* **Kapitel "CODEbed" auf Seite 104** *Kapitel hinzugefügt* **Kapitel "TESTbed" auf Seite 105** *Kapitel hinzugefügt* **Kapitel "Datenstruktur" auf Seite 213** *Kapitel hinzugefügt*

Rev.	Datum	Änderungen
02 (2/2)	25.03.2021 (2/2)	**Kapitel "Beispiel" auf Seite 221** *Kapitel hinzugefügt* **Kapitel "Spezialwerte der Datentypen" auf Seite 223** *Kapitel hinzugefügt* **Kapitel "Demontage/Entsorgung" auf Seite 229** *Hinweis auf die im Gerät enthaltene Lithium-Knopfzelle und die damit verbundenen Auswirkungen betreffend des Recycling bzw. der umweltgerechten Entsorgung des Geräts hinzugefügt* **Kapitel "Glossar" auf Seite 251** *Erklärungen der Begriffe "App Center", "App Model", "IoT App" und "rapidM2M Store " hinzugefügt*
03 (1/3)	19.08.2022 (1/3)	**Kapitel "Technische Daten" auf Seite 13** *Hinweis auf die Reset Taste hinzugefügt* *Angabe der max. Größe eines Datensatzes von 1024Bytes auf 1023Bytes angepasst* *Angaben zum WiFi-Interface entfernt.* *Angabe der unterstützten Frequenzbänder für rapidM2M C32x 2G/M1/NB1 World , rapidM2M C32x 3G World und rapidM2M C32x 2G/4G EU hinzugefügt.* *Angabe zum rapidM2M C32x WIFI/3G EU entfernt* **Kapitel "Übersicht" auf Seite 20** *"Taste" auf "MDN Taste" umbenannt* **Kapitel "Systemarchitektur " auf Seite 21** *Kapitel hinzugefügt* **Kapitel "Blockschaltbild" auf Seite 22** *Kapitel hinzugefügt* **Kapitel "Funktionsweise des internen Datenspeichers " auf Seite 31** *Kapitel hinzugefügt* **Kapitel "Speicherorganisation" auf Seite 33** *Kapitel hinzugefügt* **Kapitel "Vorgehensweise bei Verbindungsabbrüchen" auf Seite 34** *Kapitel hinzugefügt* **Kapitel "Timeout-Überwachung im Online-Modus " auf Seite 35** *Kapitel hinzugefügt* **Kapitel "Ermittlung der GSM/UMTS/LTE-Signalstärke" auf Seite 36** *Kapitel hinzugefügt* **Kapitel "Ermittlung der GSM-Positionsdaten" auf Seite 36** *Kapitel hinzugefügt* **Kapitel "File Transfer" auf Seite 38** *Kapitel hinzugefügt* **Kapitel "Anschluss der Erweiterungsmodule" auf Seite 51** *Kapitel hinzugefügt* **Kapitel "Technische Details zu den Universaleingängen" auf Seite 59** *Kapitel hinzugefügt* **Kapitel "Technische Details zur RS485-Schnittstelle" auf Seite 60** *Kapitel hinzugefügt*

Rev.	Datum	Änderungen
03 (2/3)	19.08.2022 (2/3)	**Kapitel "Technische Details zur CAN-Schnittstelle" auf Seite 61** *Kapitel hinzugefügt* **Kapitel "Technische Details zur RS232-Schnittstelle" auf Seite 63** *Kapitel hinzugefügt* **Kapitel "Technische Details zu den Ausgängen" auf Seite 65** *Kapitel hinzugefügt* **Kapitel "Technische Details zum integrierten Pufferakku" auf Seite 65** *Kapitel hinzugefügt* **Kapitel "Technische Details zur Energieversorgung" auf Seite 67** *Kapitel hinzugefügt* **Kapitel "Technische Details zur Systemzeit" auf Seite 67** *Kapitel hinzugefügt* **Kapitel "Bedienelemente" auf Seite 73** *"Taste" auf "MDN Taste" umbenannt* **Kapitel "Grundeinstellungen" auf Seite 77** *Erklärung des Parameters zur Auswahl der Auswertungs-Vorlage, die zur Darstellung der Daten verwendet wird, überarbeitet (Wenn keine Auswertungs-Vorlage ausgewählt wurde, wird das Symbol zur Anzeige der Messdaten in der Liste der Sites/Applikationen nicht angezeigt.)* **Kapitel "Messgerät" auf Seite 78** *Erklärung der nicht mehr verwendeten Felder "Modem Version" und "OS Version" entfernt* **Kapitel "Bereich "Kunden"" auf Seite 94** *Screenshots der Benutzeroberfläche des myDatanet-Servers an Version 50v007 angepasst* **Kapitel "Bereich "Sites / Applikationen" auf Kundenebene" auf Seite 96** *Screenshots der Benutzeroberfläche des myDatanet-Servers an Version 50v007 angepasst* **Kapitel "Konstanten" auf Seite 108** *Erklärung des Returncodes "ERROR_SENSOR_DISABLED" überarbeitet* **Kapitel "Uplink" auf Seite 114** *Erklärung des Arrays mit symbolischen Indizes "TrM2M_GSMInfo" um die Beschreibung der Elemente "act", "lac" und "cid" erweitert* *Erklärung des Arrays mit symbolischen Indizes "TrM2M_TxItfStats" hinzugefügt* *Erklärung der Konstanten für die Mobile Radio AcT (Access Technology) hinzugefügt* *Erklärung der Konstanten für die Signalstärkemessungs-Flags hinzugefügt* *Erklärung der vom Device Logic Entwickler bereitzustellenden Funktion, die nach dem Lesen eines Datensatzes aus dem internen Flash-Speicher aufgerufen wird, um die Beschreibung des Parameters "timestamp256" erweitert* *Erklärung der Funktionen "rM2M_TxItfGetStats()" und "rM2M_SetTCPKeepAlive()" hinzugefügt* *Erklärung der Funktionen "rM2M_GSMGetRSSI()" und "rM2M_GetRSSI()" um die Beschreibung des Parameters "flags" erweitert* **Kapitel "Position" auf Seite 143** *Erklärung der Funktion "rM2M_PosUpdate()" hinzugefügt*

Rev.	Datum	Änderungen
03 (3/3)	19.08.2022 (3/3)	**Kapitel "Char & String" auf Seite 160** *Hinweis zu den Funktionen "strchr", "strrchr", "strspn", "strcspn" , "strpbrk" , "strstr" , "strtol" und "atof" hinzugefügt, dass Strings > 128 Bytes nicht unterstützt werden* *Erklärung der Funktionen "memcpy_native", "memset_native" und "memcomp_ native" hinzugefügt* **Kapitel "Verschiedene Funktionen" auf Seite 170** *Erklärung der Funktion "delay_us()" hinzugefügt* **Kapitel "File Transfer" auf Seite 180** *Erklärung der File Transfer Kommandos "FT_CMD_ENUM" und "FT_CMD_ RETR" hinzugefügt* *Erklärung der vom Script-Entwickler bereitzustellenden Callback-Funktion um die Beschreibung, wie die File Transfer Kommandos "FT_CMD_ENUM" und "FT_ CMD_RETR" zu behandeln sind, erweitert.* *Erklärung der Funktion "FT_RegisterEnum()" hinzugefügt* **Kapitel "Device Logic Fehlercodes" auf Seite 186** *Erklärung des Fehlercodes "SCRIPT_ERR, SCRIPT UPDATE ERROR" überarbeitet* *Erklärung der Fehlercodes "SCRIPT_ERR, SCRIPT SYSTEM SHUTDOWN", "SCRIPT_ERR, SCRIPT DOWNLOAD ERROR" und "SCRIPT_ERR, SCRIPT DELETED" hinzugefügt* *Erklärung der Fehlercodes "LOG_NOSCRIPT_ERR, SCRIPT xxx" hinzugefügt* **Kapitel "Data Descriptor " auf Seite 213** *Angabe der max. Größe eines Datensatzes von 1024Bytes auf 1023Bytes angepasst* **Kapitel "rapidM2M Playground " auf Seite 225** *Screenshot und Beschreibung des rapidM2M Playgrounds aktualisiert (Button "System Console" wurde entfernt, Button für die Global Settings wurde hinzugefügt)* **Kapitel "Log-Einträge und Fehlercodes" auf Seite 233** *Erklärung der Fehlercodes "GSM NETWORK REGISTRATION", "GPRS NETWORK REGISTRATION" ,"LTE NETWORK REGISTRATION", "SHT2X SENSOR OK", "SHT2X RH ERROR", "SHT2X TEMP ERROR", "SHT2X RH+TEMP ERROR", "SHT2X PLAUSIBILITY ERROR", "SHT2X COM ERR", "SHT2X COM ERR1", "SHT2X COM ERR2", "SHT2X TEMP RAW" und "SHT2X RH RAW" hinzugefügt* **Kapitel "Montagesets" auf Seite 241** *Kapitel hinzugefügt* **Kapitel "Antennen" auf Seite 241** *Zubehörliste überarbeitet* **Kapitel "Versorgung" auf Seite 241** *Zubehörliste überarbeitet* **Kapitel "Adapter" auf Seite 241** *Kapitel hinzugefügt* **Kapitel "Erweiterungsmodule" auf Seite 242** *Kapitel hinzugefügt* **Kapitel "Glossar" auf Seite 251** *Erklärungen der Begriffe "Device Logic", "Hardware ID String", "Produktrevision" und "rapidM2M Timestamp" hinzugefügt*

Rev.	Datum	Änderungen
04	26.09.2022	**Kapitel "Konformitätserklärung" auf Seite 11** *Konformitätserklärung der Variante myDatalogC33x WIFI/2G/3G/4G World hinzugefügt* **Kapitel "Technische Daten" auf Seite 13** *Angabe der unterstützten Frequenzbänder für myDatalogC33x WIFI/2G/3G/4G World hinzugefügt.* **Kapitel "Übersicht" auf Seite 20** *Abbildung der Vorderseite aktualisiert (VIN und GND waren vertauscht)* **Kapitel "Blockschaltbild" auf Seite 22** *Abschlusswiderstände der CAN-Schnittstelle ins Blockschaltbild eingezeichnet* **Kapitel "Gerätekennzeichnung" auf Seite 25** *Typenschild aktualisiert* **Kapitel "Lieferumfang" auf Seite 41** *Auflistung der verfügbaren Varianten aktualisiert* **Kapitel "Abmessungen" auf Seite 43** *Abbildung aktualisiert (VIN und GND waren vertauscht)* **Kapitel "Anschluss der Sensoren, der Aktoren und der Versorgung" auf Seite 46** *Abbildung aktualisiert (VIN und GND waren vertauscht)* *Hinweis hinzugefügt, dass es sich bei den Ausgängen um eine Parallelschaltung eines PhotoMOS-Relais (hohe Schaltfrequenzen) und eines mechanischen Relais (hohe Schaltströme) handelt* **Kapitel "Anschlussbeispiele" auf Seite 49** *Abbildungen aktualisiert (VIN und GND waren vertauscht)* **Kapitel "Anschluss der GSM-Antenne" auf Seite 50** *Verweis auf die im Falle einer niedrigen Funksignalstärke alternativ zu verwendenden Antennen auf die Kuppelantenne Multiband SMA-M 3m (301211) geändert.* **Kapitel "Technische Details zur CAN-Schnittstelle" auf Seite 61** *Fehler in der Tabelle, die angibt, mit welchen Schaltern die Abschlusswiderstände geschaltet werden, behoben. Mit S1 und S2 wird der 2k Abschlusswiderstand aktiviert und nicht der 120Ω Abschlusswiderstand.* **Kapitel "Bedienelemente" auf Seite 73** *Abbildung aktualisiert (VIN und GND waren vertauscht)* **Kapitel "Montagesets" auf Seite 241** *Wandmontageset für Gehäuse 300x200x150mm (301185) hinzugefügt*
05	17.10.2022	**Kapitel "Anschluss der GSM-Antenne" auf Seite 50** *Bestellnummer der Kuppelantenne Multiband SMA-M 3m korrigiert von (301211) auf (301212)*

Rev.	Datum	Änderungen
06 (1/2)	22.09.2023 (1/2)	**Kapitel "Konformitätserklärung" auf Seite 11** *Variante rapidM2M C33x WIFI/3G World entfernt* *Variante rapidM2M C33x WIFI/2G/4G EU entfernt* **Kapitel "Technische Daten" auf Seite 13** *Variante rapidM2M C33x WIFI/3G World entfernt* *Variante rapidM2M C33x WIFI/2G/4G EU entfernt* **Kapitel "Lieferumfang" auf Seite 41** *Variante rapidM2M C33x WIFI/3G World entfernt* *Variante rapidM2M C33x WIFI/2G/4G EU entfernt* **Kapitel "Montage des myDatalogC33x " auf Seite 43** *Hinweis hinzugefügt, dass das Gerät nicht für den Einsatz in geschlossenen Kanälen zugelassen ist.* **Kapitel "Konstanten" auf Seite 108** *Returncode "ERROR_SENSOR_DISABLED" hinzugefügt* **Kapitel "Timer, Datum & Zeit" auf Seite 109** *Erklärung des Arrays mit symbolischen Indizes "TrM2M_DateTime" erweitert* *Erklärungen der Funktionen "rM2M_TimerAdd()" und "rM2M_TimerAddExt()" erweitert* **Kapitel "Uplink" auf Seite 114** *Erklärung der Funktion "rM2M_CfgRead()" erweitert* *Erklärung des Rückgabewerts der Funktion "rM2M_CfgWrite()" korrigiert* *Liste der Fehlercodes, die von der Funktion "rM2M_TxGetStatus()" zurückgeliefert werden können, um die Fehlercodes "RM2M_TXERR_MODEM_RESETLOOP", "RM2M_TXERR_MODEM_UNDERVOLTAGE" und "RM2M_TXERR_MODEM_OVERHEAT" erweitert.* **Kapitel "Encoding" auf Seite 130** *Erklärung der Funktion "rM2M_SetPacked()" dahingehend korrigiert, dass es in Verbindung mit Signed-Datentypen zu Problemen kommen kann und nicht wie bisher angegeben, in Verbindung mit Unsigned-Datentypen.* *Erklärung der Funktion "rM2M_Pack()" erweitert* **Kapitel "Registry" auf Seite 136** *Erklärung der Konstanten für "Indizes der Registrierungsspeicherblöcke" erweitert* *Erklärungen der Funktionen "rM2M_RegGetString()", "rM2M_RegGetValue()", "rM2M_RegSetString()", "rM2M_RegSetValue()" und "rM2M_RegOnChg()" erweitert* **Kapitel "64-Bit Signed Integer" auf Seite 157** *Kapitel hinzugefügt*

Rev.	Datum	Änderungen
06 (2/2)	22.09.2023 (2/2)	**Kapitel "Char & String" auf Seite 160** *Erklärungen der Funktionen "sprintf()", "strcat()", "strcmp()", "strcspn()", "strpbrk()", "strstr()", "strtol()" und "atof()" erweitert* *Hinweis, dass Strings > 128 Bytes nicht unterstützt werden, aus den Erklärungen der Funktionen "strchr()", "strrchr()", "strspn()", "strcspn()","strpbrk()" und "strtsr()" entfernt.* **Kapitel "CRC & Hash" auf Seite 168** *Erklärung der Funktion "MD5()" erweitert* **Kapitel "Verschiedene Funktionen" auf Seite 170** *Erklärungen der Funktionen "getapilevel()", "exists()", "rtm_start()", "funcidx()", "numargs()" und "getarg()" erweitert* **Kapitel "Console" auf Seite 176** *Erklärung der Funktion "printf()" erweitert* *Hinweis zur Funktion "setbuf()" hinzugefügt, dass der Puffer "buf" während der gesamten Nutzung durch die Firmware gültig sein muss.* **Kapitel "Unterschiede zu C" auf Seite 209** *Hinweis hinzugefügt, dass Variable automatisch mit "0" initialisiert werden* **Kapitel "Datenstruktur" auf Seite 213** *Erklärung des Attributs "editmask" überarbeitet* **Kapitel "Log-Einträge und Fehlercodes" auf Seite 233** *Erklärung des Fehlercodes "MODEM NOT FOUND" hinzugefügt* *Erklärung des Fehlercodes "ACCU 0 E2PROM ERROR" hinzugefügt*

Kapitel 21 Glossar

App Center

Bereich des myDatanet-Servers für die Installation und Verwaltung der IoT Apps. Die als Basis für die IoT Apps dienenden App Models werden über den rapidM2M Store bezogen. Bei der Installation einer IoT App am myDatanet-Server werden zunächst die bei der Entwicklung des App Models festgelegten Standardsettings übernommen. Diese Standardsettings können anschließend angepasst werden. Auf Basis eines einzelnen App Models können so durch Setzen entsprechender Standardsettings beliebig viele IoT Apps erzeugt werden.

App Model

Ein App Model wird im rapidM2M Studio entwickelt und bildet die Grundlage zum Erstellen von IoT Apps. Es enthält im Wesentlichen die ausführbaren Programmdateien (Device Logic, Backend Logic, Portal View, usw.) aus denen durch Hinzufügen von Standardsettings eine IoT App erzeugt wird. Die Verteilung an die einzelnen myDatanet-Server erfolgt über den rapidM2M Store . Angezeigt werden die verfügbaren App Models im App Center des jeweiligen myDatanet-Servers.

Footprint

Die Geräte des Herstellers sind ab Werk mit Subscriber Identitiy Modules (SIM) zur mobilen Übertragung der Daten ausgestattet. Der Footprint bezeichnet jene Länder und Regionen, in denen eine Mobilfunkverbindung zur Verfügung steht(siehe www.microtronics.com/footprint).

Device Logic

Bei der Device Logic handelt es sich um die am Gerät installierte Intelligenz durch die die lokale Funktionalität des Geräts bestimmt wird. Die Device Logic ist Bestandteil des App Models und wird mittels einer C-ähnliche Scriptsprache built on PAWN erstellt.

Hardware ID String

Gibt die im Gerät verbaute Hardwareplattform und deren Hardwareversion an (z.B. rapidM2M M2 HW1.4). Der Teil des Hardware ID Strings, der die Hardwareversion angibt, wird nur dann erhöht, wenn für die rapidM2M Firmware relevante Änderungen an der Hardwareplattform vorgenommen wurden. Bei der Entwicklung eines App Models kann angegeben werden, auf welchen Hardwareplattformen das App Model installiert werden kann und welche Version der Hardwareplattform mindestens erforderlich ist. Der Hardware ID String wird unter anderem im TESTbed des rapidM2M Studio oder im Feld „Identifikation“ der Eingabemaske zur Konfiguration des Geräts angezeigt.

IoT App

IoT Apps bilden den Grundstein zum Erstellen von Sites. Sie bestehen aus einem App Model und entsprechenden Standardsettings, die beim Anlegen der Site als Default-Werte für die Site übernommen werden. Mit Hilfe des App Centers können auf Basis eines einzelnen App Models durch Setzen entsprechender Standardsettings beliebig viele IoT Apps erzeugt werden. Dies bietet sich an, wenn mittels eines App Models mehrere Use Cases abgedeckt werden sollen, die jeweils eine unterschiedliche Default-Konfiguration der Sites erfordern (z.B. wenn ein Datenlogger mit verschiedenen externen Sensoren als Paket vertrieben werden soll).

NaN-Wert

Beim myDatanet werden spezielle Kodierungen verwendet, um verschiedene Fehlerzustände in z.B. den Messwerten anzuzeigen. Durch das Setzen eines Messwerts auf "NaN" wird dieser eindeutig als ungültig gekennzeichnet und somit nicht mehr für weitere Berechnungen verwendet. In den Messwertgrafiken wird ein auf "NaN" gesetzter Messwert durch eine Unterbrechung in der Ganglinie angezeigt. Beim Download der Daten wird ein auf "NaN" gesetzter Messwert durch ein leeres Datenfeld signalisiert.

Produktrevision

Gibt die Revision des Produktes an. Sie wird bei jeder Änderung am Produkt (d.h. Elektronik, Mechanik, usw.) erhöht und ist am Typenschild des Produktes vermerkt.

rapidM2M Store

Übernimmt die Verteilung der App Models an die einzelnen myDatanet-Server. Bei der Installation und beim Update von IoT Apps greifen die myDatanet-Server auf die im rapidM2M Store bereitgestellten App Models zu. Welche myDatanet-Server auf ein App Model zugreifen dürfen, wird vom Entwickler des jeweiligen App Models über das rapidM2M Studio festgelegt.

rapidM2M Timestamp

Je nach erforderlicher Genauigkeit kann bei rapidM2M für die Zeitstempel eine von 2 speziellen Kodierungen verwendet werden. Bei moderaten Anforderungen an die Genauigkeit kann der Datentyp „stamp32“ (Sekunden seit 1999-12-31 00:00:00 UTC) verwendet werden. Ist eine höhere Genauigkeit erforderlich, kann der Datentyp „stamp40“ (1/256 Sekunden seit 1999-12-31 00:00:00 UTC) eingesetzt werden. Die Umrechnung des Datentyp „stamp32“ in den UNIX Timestamp (Sekunden seit 1970-01-01 00:00:00 UTC) kann durch Addition von 946598400 erfolgen.

Kapitel 22 Kontaktinformationen

Support & Service:
Microtronics Engineering GmbH
Hauptstrasse 7
3244 Ruprechtshofen
Austria, Europe
Tel. +43 (0)2756 7718023
support@microtronics.com
www.microtronics.com

Microtronics Engineering GmbH
(Headquarters)
Hauptstrasse 7
3244 Ruprechtshofen
Austria, Europe
Tel. +43 (0)2756 77180
Fax. +43 (0)2756 7718033
office@microtronics.com
www.microtronics.com

Zertifiziert durch TÜV AUSTRIA: EN ISO 9001:2015, EN ISO 14001:2015, ISO/IEC 27001:2013, EN ISO 50001:2011 für myDatanet | TÜV SÜD: ATEX Richtlinie 2014/34/EU

Microtronics Engineering GmbH | www.microtronics.com
Hauptstrasse 7 | 3244 Ruprechtshofen | Austria | +43 2756 77180 | office@microtronics.com