Benutzerhandbuch
EC-CAN-00x

Gültig ab:

- **Firmware Version: 01v010**
- **Server Version: 44.1**
- **Hardware Version: 1.0**



Microtronics

Kapitel 1 Inhaltsverzeichnis

Deckblatt ... **1**
Kapitel 1 Inhaltsverzeichnis ... **3**
Kapitel 2 FCC Statement ... **11**
Kapitel 3 Technical Compliance Statement FCC ... **13**
Kapitel 4 Konformitätserklärung ... **15**
4.1 EC-CAN-001 ... 15
4.2 EC-CAN-002 ... 16
Kapitel 5 Technische Daten ... **17**
Kapitel 6 Allgemeine Angaben ... **19**
6.1 Übersetzung ... 19
6.2 Copyright ... 19
6.3 Gebrauchsnamen ... 19
6.4 Sicherheitshinweise ... 19
6.4.1 Verwendung der Gefahrenhinweise ... 20
6.4.2 Allgemeine Sicherheitshinweise ... 20
6.4.3 Sicherheits-/Vorsichtsmaßnahmen im Umgang mit GSM/GPRS-Modems ... 21
6.4.3.1 Sicherheits-/Vorsichtsmaßnahmen für den GSM/GPRS-Modemeinbau ... 21
6.4.3.2 Sicherheitsmaßnahmen für den Antenneneinbau ... 21
6.5 Übersicht ... 22
6.5.1 Blockschaltbild ... 23
6.6 Bestimmungsgemäße Verwendung ... 23
6.7 Allgemeine Produktinformationen ... 24
6.8 Gerätekennzeichnung ... 24
6.8.1 Typenschilder ... 25
6.8.2 Verpackungsetiketten ... 25
6.9 Einbau von Ersatz- und Verschleißteilen ... 25
6.10 Aufbewahrung des Produkts ... 26
6.11 Pflichten des Betreibers ... 26
6.12 Anforderungen an das Personal ... 27
Kapitel 7 Funktionsprinzip ... **29**
7.1 Empfohlene Vorgehensweise ... 31

7.1.1 Entwicklung einer M2M / IoT Anwendung ... 31
7.2 Funktionsweise des internen Datenspeichers ... 31
7.3 Speicherorganisation ... 33
7.4 Vorgehensweise bei Verbindungsabbrüchen ... 34
7.4.1 Verbindungsabbruch im "online"-Modus ... 35
7.4.2 Verbindungsabbruch während eines Scriptdownloads ... 35
7.5 Automatische Auswahl des GSM-Netzes ... 35
7.6 Errorhandling ... 35
7.7 Registrierungsspeicherblöcke ... 36
7.7.1 REG_SYS_FLASH ... 37
7.7.2 REG_APP_OTP ... 37
7.8 File Transfer ... 38
7.9 Bedeutung des SIM-Status ... 38
Kapitel 8 Lagerung, Lieferung und Transport ... 41
8.1 Eingangskontrolle ... 41
8.2 Lieferumfang ... 41
8.3 Lagerung ... 41
8.4 Transport ... 41
8.5 Rücksendung ... 42
Kapitel 9 Installation ... 43
9.1 Abmessungen ... 43
9.2 Montage des EC-CAN-00x ... 43
9.2.1 Hutschienenmontage ... 44
9.3 Sicherheitshinweise zur Verkabelung ... 45
9.3.1 Hinweise zur Vermeidung elektrostatischer Entladungen (ESD) ... 45
9.4 Elektrische Installation ... 45
9.4.1 Anschluss der Sensoren, der Aktoren und der Versorgung ... 46
9.4.2 Anschluss der GSM-Antenne ... 47
9.4.3 Technische Details zur CAN-Schnittstelle ... 48
9.4.4 Technische Details zur USB-Schnittstelle ... 49
9.4.5 Technische Details zum Energiemanagement ... 49
9.4.6 Technische Details zur Energieversorgung ... 50
9.4.7 Technische Details zur Systemzeit ... 50

Kapitel 10 Inbetriebnahme ... **51**
10.1 Hinweise an den Benutzer ... 51
10.2 Allgemeine Grundsätze ... 51
10.3 Inbetriebnahme des Systems ... 51
10.4 Kommunikation mit dem Gerät testen ... 53
Kapitel 11 Benutzerschnittstellen ... **55**
11.1 Benutzerschnittstelle am EC-CAN-00x ... 55
11.1.1 Bedienelemente ... 55
11.1.1.1 Taste ... 55
11.1.1.2 LED ... 56
11.2 Benutzerschnittstelle am myDatanet-Server ... 56
11.2.1 Messstellenkonfiguration ... 56
11.2.1.1 Site ... 56
11.2.1.2 Kommentar ... 56
11.2.1.3 Steuerung ... 57
11.2.1.4 Konfiguration 0 - Konfiguration 1 ... 57
11.2.1.5 Alarmierung ... 58
11.2.1.6 Grundeinstellungen ... 59
11.2.1.7 FTP-Export Einstellungen ... 60
11.2.2 Gerätekonfiguration ... 60
11.2.2.1 Kommentar ... 60
11.2.2.2 Messgerät ... 60
11.2.2.3 GPRS ... 62
Kapitel 12 rapidM2M Toolset ... **63**
12.1 Allgemein ... 63
12.2 Voraussetzungen ... 63
12.3 Funktionsprinzip ... 64
12.4 Installation ... 65
12.5 Übersicht ... 68
12.6 Kateireiter "Script" ... 70
12.7 Karteireiter "Config" ... 72
12.8 Karteireiter "Console" ... 73
12.9 Karteireiter "Server" ... 74

12.10 Karteireiter "Log" ... 75
12.11 Karteireiter "FW" ... 76
12.12 Hot-Keys ... 76
Kapitel 13 myDatanet-Server ... 77
13.1 Übersicht ... 77
13.1.1 Erklärung der Symbole ... 77
13.2 Bereich "Kunden" ... 78
13.3 Bereich "Messstellen" auf Kundenebene ... 80
13.3.1 Auswertungen ... 81
13.3.2 Gruppen ... 81
13.3.3 Kartendarstellung ... 81
13.4 Empfohlene Vorgehensweise ... 81
13.4.1 Anlegen der Messstelle ... 81
13.4.2 Gerät dem Kunden zuweisen ... 83
Kapitel 14 Pawn Script ... 85
14.1 Allgemein ... 85
14.1.1 Direkte Eingabe eines Pawn Scripts ... 85
14.1.2 Hochladen eines Binary-Files ... 85
14.1.3 Script-Editor des rapidM2M Toolset ... 85
14.2 Compiler-Optionen ... 85
14.3 rapidM2M API ... 86
14.3.1 Konstanten ... 86
14.3.2 Timer, Datum & Zeit ... 87
14.3.2.1 Arrays mit symbolischen Indizes ... 87
14.3.2.2 Konstanten ... 87
14.3.2.3 Funktionen ... 87
14.3.3 Uplink ... 92
14.3.3.1 Arrays mit symbolischen Indizes ... 92
14.3.3.2 Konstanten ... 92
14.3.3.3 Callback Funktionen ... 96
14.3.3.4 Funktionen ... 96
14.3.4 Encoding ... 105
14.3.4.1 Konstanten ... 105

14.3.4.2 Funktionen ... 106
14.3.5 GPIO, IRQ ... 111
14.3.5.1 Konstanten ... 111
14.3.5.2 Callback Funktionen ... 111
14.3.5.3 Funktionen ... 112
14.3.6 SPI, I2C, UART ... 114
14.3.6.1 Konstanten ... 114
14.3.6.2 Callback Funktionen ... 114
14.3.6.3 Funktionen ... 115
14.3.7 Registry ... 121
14.3.7.1 Konstanten ... 121
14.3.7.2 Callback Funktionen ... 122
14.3.7.3 Funktionen ... 123
14.3.8 Position ... 127
14.3.8.1 Arrays mit symbolischen Indizes ... 127
14.3.8.2 Konstanten ... 129
14.3.8.3 Funktionen ... 131
14.3.9 Mathematik ... 137
14.3.10 Char & String ... 140
14.3.11 CRC & Hash ... 145
14.3.11.1 Arrays mit symbolischen Indizes ... 145
14.3.11.2 Funktionen ... 145
14.3.12 Verschiedene Funktionen ... 146
14.3.12.1 Arrays mit symbolischen Indizes ... 146
14.3.12.2 Konstanten ... 147
14.3.12.3 Funktionen ... 147
14.3.13 Console ... 153
14.3.14 SMS ... 154
14.3.14.1 Callback Funktionen ... 154
14.3.14.2 Funktionen ... 154
14.3.15 File Transfer ... 155
14.3.15.1 Arrays mit symbolischen Indizes ... 155
14.3.15.2 Konstanten ... 155

14.3.15.3 Callback Funktionen ... 156
14.3.15.4 Funktionen ... 157
14.4 Pawn Script Fehlercodes ... 160
14.5 Syntax ... 163
14.5.1 Allgemeine Syntax ... 163
14.5.1.1 Format ... 163
14.5.1.2 Optionale Semikolons ... 163
14.5.1.3 Kommentare ... 163
14.5.1.4 Bezeichner ... 163
14.5.1.5 Reservierte Schlüsselworte ... 164
14.5.1.6 Numerische Konstanten ... 164
14.5.1.6.1 Numerische Integer-Konstanten ... 164
14.5.1.6.2 Numerische Gleitkomma-Konstanten ... 164
14.5.2 Variablen ... 164
14.5.2.1 Deklaration ... 164
14.5.2.2 Lokale Deklaration ... 165
14.5.2.3 Globale Deklaration ... 165
14.5.2.4 Statische lokale Deklaration ... 165
14.5.2.5 Statische globale Deklaration ... 165
14.5.2.6 Gleitkommawerte ... 165
14.5.3 Konstante Variablen ... 165
14.5.4 Array Variablen ... 166
14.5.4.1 Eindimensionales Array ... 166
14.5.4.2 Initialisierung ... 166
14.5.4.3 Progressive Initialisierung für Arrays ... 167
14.5.4.4 Mehrdimensionale Arrays ... 167
14.5.4.5 Arrays und der "sizeof"-Operator ... 167
14.5.5 Operatoren und Ausdrücke ... 168
14.5.5.1 Zeichenerklärung ... 168
14.5.5.2 Ausdrücke ... 168
14.5.5.3 Arithmetik ... 169
14.5.5.4 Bit-Manipulation ... 169
14.5.5.5 Zuweisung ... 169

14.5.5.6 Vergleichsoperatoren ... 170
14.5.5.7 Boolean ... 171
14.5.5.8 Sonstiges ... 171
14.5.5.9 Priorität der Operatoren ... 172
14.5.6 Anweisungen ... 173
14.5.6.1 Statement-Etikett ... 173
14.5.6.2 Zusammengesetzte Anweisungen ... 173
14.5.6.3 Ausdrucksanweisung ... 173
14.5.6.4 Leeres Statement ... 173
14.5.6.5 assert Ausdruck ... 174
14.5.6.6 break ... 174
14.5.6.7 continue ... 175
14.5.6.8 do Statement while (Ausdruck) ... 175
14.5.6.9 exit Ausdruck ... 175
14.5.6.10 for (Ausdruck 1 ; Ausdruck 2 ; Ausdruck 3) Statement ... 176
14.5.6.11 goto Etikett ... 176
14.5.6.12 if (Ausdruck) Statement 1 else Statement 2 ... 177
14.5.6.13 return Ausdruck ... 177
14.5.6.14 switch (Ausdruck) { case Liste } ... 177
14.5.6.15 while (Ausdruck) Statement ... 178
14.5.7 Funktionen ... 178
14.5.7.1 Funktionsargumente ("call-by-value" versus "call-by-reference") ... 179
14.5.7.2 Benannte Parameter versus positionsgebundene Parameter ... 181
14.5.7.3 Standardwerte von Funktionsargumenten ... 181
14.6 Beispiele ... 182
14.7 Unterschiede zu C ... 183
Kapitel 15 myDatanet Connector ... 185
15.1 Voraussetzungen ... 185
15.2 Split-tag ... 186
15.3 Datenstruktur ... 187
15.3.1 Attribute der Feld-Definition ... 188
15.4 Beispiel ... 191
15.5 Spezialwerte der Datentypen ... 192

Kapitel 16 API ... **193**
16.1 Allgemein ... 193
16.2 rapidM2M Playground ... 193
16.2.1 Übersicht ... 194
Kapitel 17 Wartung ... **195**
17.1 Allgemeine Wartung ... 195
17.2 Sicherungswechsel ... 195
Kapitel 18 Demontage/Entsorgung ... **197**
Kapitel 19 Fehlersuche und Behebung ... **199**
19.1 Allgemeine Probleme ... 199
19.2 Log-Einträge und Fehlercodes ... 200
19.2.1 Modemfehler ... 203
19.3 Auswerten des Gerätelogs ... 204
19.3.1 Auswerten des Gerätelogs am myDatanet-Server ... 204
19.3.2 Auswerten des Gerätelogs mittels rapidM2M Toolset ... 205
Kapitel 20 Dokumentenhistorie ... **207**
Kapitel 21 Glossar ... **211**
Kapitel 22 Kontaktinformationen ... **213**

Kapitel 2 FCC Statement

Labelling requirements:

This device complies with part 15 of the FCC Rules. Operation is subject to the following two conditions:

1. This device may not cause harmful interference, and
2. this device must accept any interference received, including interference that may cause undesired operation.

Information to user:

Caution: Any Changes or modifications not expressly approved by the party responsible for compliance could void the user's authority to operate the equipment.

Note: This equipment has been tested and found to comply with the limits for a Class B digital device, pursuant to part 15 of the FCC Rules. These limits are designed to provide reasonable protection againstharmful interference in a residential installation. This equipment generates uses and can radiate radio frequency energy and, if not installed and used in accordance with the instructions, may cause harmful interferenceto radio communications. However, there is no guarantee that interference will not occur in a particular installation. If this equipment does cause harmful interference to radio or television reception, which can be determined by turning the equipment off and on, the user is encouraged to try to correctthe interference by one or more of the following measures:

- Reorient or relocate the receiving antenna.
- Increase the separation between the equipment and receiver.
- Connect the equipment into an outlet on a circuit different from that to which the receiver is connected.
- Consult the dealer or an experienced radio/TV technician for help.

RF warning for Portable device:

- The device has been evaluated to meet general RF exposure requirement.
- The device can be used in portable exposure condition without restriction.

Kapitel 3 Technical Compliance Statement FCC

Technical Compliance Statement

Product: GSM gateway for data acquisition via CAN bus & GPS location.
Type code: **EC-CAN-001**

Microtronics
We live M2M

Hersteller:
Manufacturer :
Fabricant:

Microtronics Engineering GmbH
Elektronik Entwicklung&Produktion
Hauptstrasse 7
AT-3244 Ruprechtshofen

We hereby certify that the above product has been tested by us and complied with the FCC official limits. The product might be marketed in US in accordance with Part 15 Subpart B class B equipment regulations under FCC Rules. The test was performed according to the procedures mentioned in ANSI C63.4:2014. The test data and results are issued on the test report no. TÜV-A INE-AT/EMV-17/216.

Rules and Standards	**FCC Part 15 (01-10-16 Edition) (Class B Limit)**

Ruprechtshofen, den 13.08.2021

Place and date of issue

Hans-Peter Buber, Managing Director

name and signature of authorised person

Kapitel 4 Konformitätserklärung

4.1 EC-CAN-001

EU-Konformitätserklärung

EU Declaration of Conformity / Déclaration de conformité UE

Produktbezeichnung:
Product:
Désignation du produit:

Celluar Gateway (2G/3G/World) zur Erfassung von Daten über den CAN-Bus & GPS-Ortung.

Type :
Type code:
Type:

EC-CAN-001

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**		
		EN61326-1	
(2014/35/EU)	**LVD Directive**		
		EN61010-1	
(2014/53/EU)	**RED Directive**		
	Safety & Health 3.1a	EN62368-1 EN62311	
	EMC 3.1b	EN301489-1 V2.1.1 EN301489-52 V1.1.0	
	Radio spectrum efficiency 3.2	EN301511 V12.5.1 EN301908-1 V13.1.1 EN301908-2 V13.1.1	
(2015/863/EU)	**RoHS Directive**		
	Prevention 4.1	EN IEC 63000	

Ruprechtshofen, den 13.08.2021

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

4.2 EC-CAN-002

EU-Konformitätserklärung

EU Declaration of Conformity / Déclaration de conformité UE

Produktbezeichnung:
Product:
Désignation du produit:

Celluar Gateway (2G/M1/NB1 World) zur Erfassung von Daten über den CAN-Bus & GPS-Ortung.

Type :
Type code:
Type:

EC-CAN-002

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**		
		EN61326-1	
(2014/35/EU)	**LVD Directive**		
		EN61010-1	
(2014/53/EU)	**RED Directive**		
	Safety & Health 3.1a	EN62368-1 EN62311	
	EMC 3.1b	EN301489-1 V2.1.1 EN301489-52 V1.1.0	
	Radio spectrum efficiency 3.2	EN301511 V12.5.1 EN301908-13 V13.1.1 EN301908-1 V13.1.1	
(2015/863/EU)	**RoHS Directive**		
	Prevention 4.1	EN IEC 63000	

Ruprechtshofen, den 13.08.2021

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 5 Technische Daten

Spannungsversorgung	24VDC (+5/-10%) Zusätzliche Informationen finden Sie unter "Technische Details zur Energieversorgung" auf Seite 50.
Leistungsaufnahme	typ. 0,5W max. 3,5W
Integrierter Pufferakku	Li-Po -Akku mit 900mAh für: • Applikative Reaktion auf den Ausfall der Versorgung • Ausbuchen aus dem Mobilfunknetz bei Ausfall der Versorgung Zusätzliche Informationen finden Sie unter "Technische Details zum Energiemanagement " auf Seite 49.
Gehäuse	Material: Aluminium Gewicht: 250g Schutzart: IP67 (alle Anschlüsse gesteckt) Abmessungen (BHT): 85 x 99 x 48mm (ohne Antenne)
Betriebstemperatur	-20...+60°C
Luftfeuchtigkeit	15...90%rH nicht kondensierend
Lagertemperatur	-20...+60°C
Ladetemperatur (Pufferakku)	0...+40°C
Anzeige	frei verwendbar LED (gesteuert durch das Script)
Bedienung	frei verwendbare Taste (Auswertung durch das Script)
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 50.
Antennenanschluss	2 x FAKRA für Mobilfunk [1)] 1 x FAKRA für GNSS
Schnittstelle	1 x CAN • max. Datenübertragungsrate: 1Mbit/s • zulässige kontinuierliche Buslast: 40% • unterstützte CAN Spezifikation: V2.0B • keine galvanische Trennung Zusätzliche Informationen finden Sie unter"Technische Details zur CAN-Schnittstelle" auf Seite 48.
Positionsbestimmung	GNSS-Modul (GPS/QZSS, GLONAS, Galileo, BeiDou)

Datenspeicher	3 MB interner Flash-Speicher Die Größe der Datensätze ist variabel (max. 1024 Byte) und wird durch das vom User erstellte Script bestimmt. Der systembedingte Overhead beträgt pro Datensatz 10 Byte . Zusätzliche Informationen finden Sie unter"Funktionsweise des internen Datenspeichers " auf Seite 31.
Konfigurationsspeicher	10 unabhängige Blöcke mit je 4000 Byte
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 PAWN-Binary	256kB (unkomprimierte Größe) Zusätzliche Informationen finden Sie unter "Speicherorganisation" auf Seite 33.
Datenübertragung	EC-CAN-001 (2G/3G Welt, u-blox SARA-U201): 2G GPRS 900MHz / 1800MHz 2G GPRS 850MHz / 1900MHz UMTS B1, B2, B5, B8, B19 EC-CAN-002 (2G/M1/NB1 Welt, u-blox SARA-R422S): 2G GPRS 900MHz / 1800MHz 2G GPRS 850MHz / 1900MHz LTE B1, B2, B3, B4, B5, B8, B12, B13, B20, B25, B26, B28, B66, B71, B85
SIM	Das EC-CAN-00x ist mit einem integrierten SIM-Chip versehen.

[1)] MAIN und AUX als Vorbereitung für Modul der Kategorien LTE CAT-1 und höher

Kapitel 6 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.*

6.1 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.

6.2 Copyright

Weitergabe, Vervielfältigung dieses Dokuments sowie Verwertung und Mitteilung seines Inhalts sind verboten, soweit nicht ausdrücklich gestattet. Zuwiderhandlungen verpflichten zu Schadenersatz. Alle Rechte vorbehalten.

6.3 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.

6.4 Sicherheitshinweise

Für Anschluss, Inbetriebnahme und Betrieb des EC-CAN-00x 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: *Die Produkte des Herstellers zur Nutzung im Freien haben einen umfangreichen Schutz gegen das Eindringen von Feuchtigkeit und Staub. Dies gilt jedoch nur, wenn sämtliche Stecker mit entsprechenden Gegenstücken verbunden sind. Es liegt in der Verantwortlichkeit des Betreibers nicht verwendete Stecker auf geeignete Weise gegen das Eindringen von Feuchtigkeit und Staub zu schützen und die lokalen Sicherheitsbestimmungen zu erfüllen.*

6.4.1 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.*

6.4.2 Allgemeine Sicherheitshinweise

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 213).

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 GSM/GPRS-Modem beeinträchtigt werden kann.

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

6.4.3 Sicherheits-/Vorsichtsmaßnahmen im Umgang mit GSM/GPRS-Modems

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

VORSICHT:
Die GSM/GPRS-Modemverbindung 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 GSM/GPRS-Modem, bei dessen Verwendung Funksignale und -netzwerke zum Einsatz kommen, wird zu keiner Zeit und unter keinen Umständen gehaftet. Das GSM/GPRS-Modem muss eingeschaltet sein und in einem Gebiet betrieben werden, in dem eine ausreichende Signalstärke vorhanden ist.*

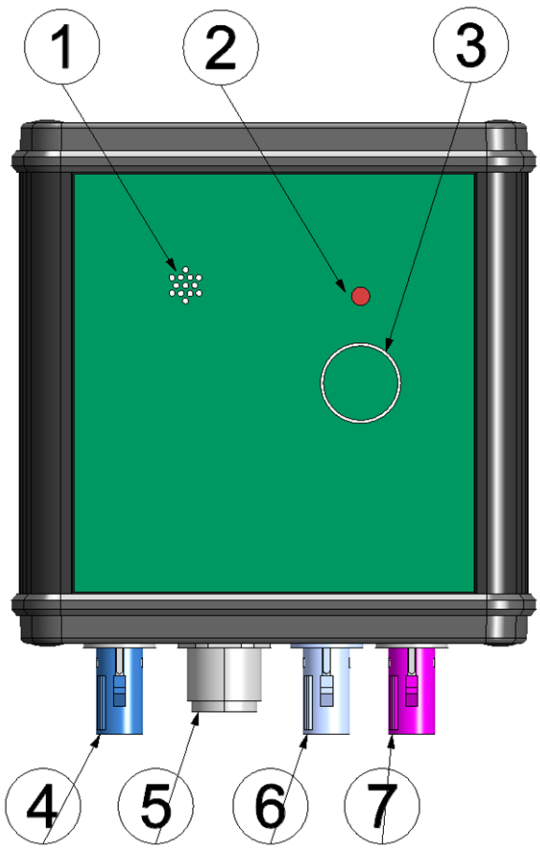
6.4.3.1 Sicherheits-/Vorsichtsmaßnahmen für den GSM/GPRS-Modemeinbau

- 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 Gerätschaften, 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 GSM/GPRS-Modem kann Störungen verursachen, wenn es sich in der Nähe von Fernsehgeräten, Radios oder Computern befindet.
- Das GSM/GPRS-Modem nicht öffnen. Eine Änderung der Gerätschaft 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 für hierdurch erfolgte Schäden und Kosten allein verantwortlich.
- Bauen Sie das Gerät nicht anders ein, als in der Bedienungsanleitung angegeben. Eine fehlerhafte Verwendung führt zum Erlöschen der Garantie.

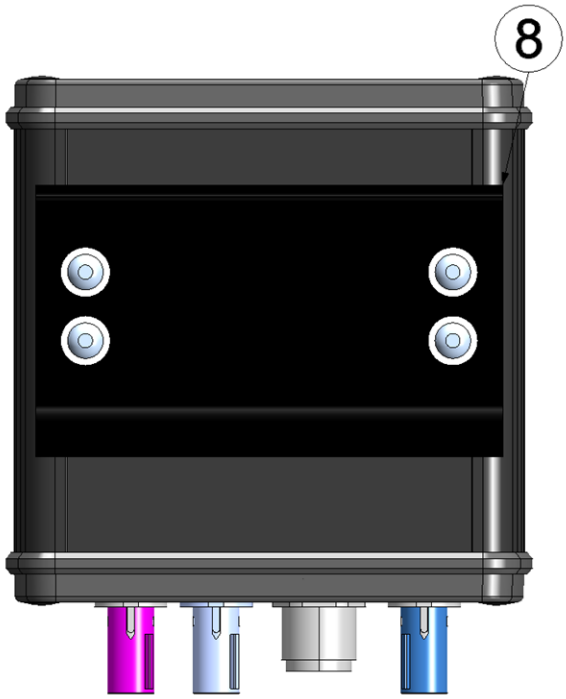
6.4.3.2 Sicherheitsmaßnahmen für den Antenneneinbau

- 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.

6.5 Übersicht



Vorderseite des EC-CAN-00x

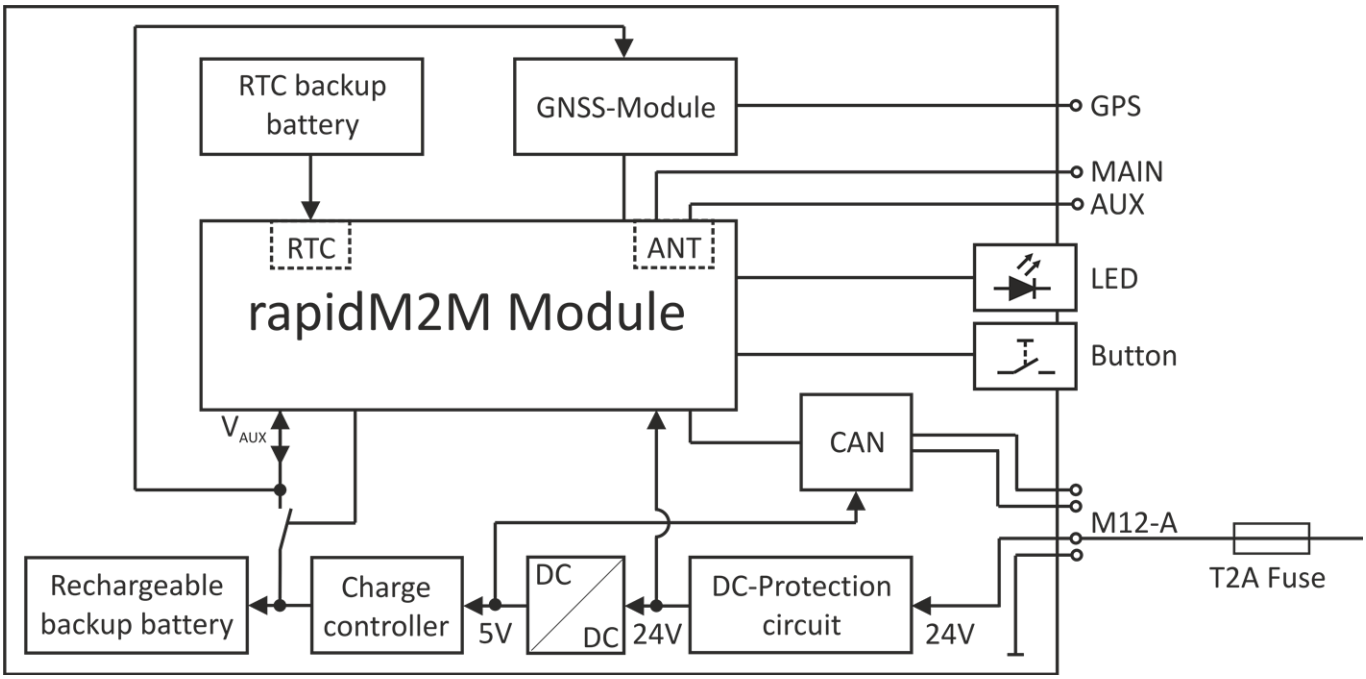


Rückseite des EC-CAN-00x

1 Druckausgleich	**5** Anschluss für die Versorgung und den CAN-Bus
2 LED	**6** Anschluss für die primäre Mobilfunkantenne (MAIN)
3 Taste	**7** Anschluss für die sekundäre Mobilfunkantenne (AUX)[1)]
4 Anschluss für die GNSS-Antenne	**8** Montagelasche (für die Befestigung an einer Hutschiene)

[1)] als Vorbereitung für Module der Kategorien LTE CAT-1 und höher

6.5.1 Blockschaltbild



Blockschaltbild des EC-CAN-00x

6.6 Bestimmungsgemäße Verwendung

Das kompakte, frei programmierbare Gerät bietet eine universelle Lösung für die Erfassung, Verarbeitung und Übertragung von Maschinendaten, welche über die CAN-Schnittstelle gelesen werden. Das Gerät benötigt eine permanente Energieversorgung. Die Speicherung der erfassten Daten erfolgt auf einem nicht flüchtigen Speichermedium. Diese gespeicherten Daten werden über das Mobilfunknetz an einen zentralen Server zur Weiterverarbeitung gesendet. Dazu ist das Gerät mit einem integrierten SIM-Chip versehen. Es sind die zulässigen maximalen Grenzwerte, aufgeführt im Kapitel "Technische Daten" auf Seite 17, 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 dazugehörige 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.*

6.7 Allgemeine Produktinformationen

Es handelt sich um ein kompaktes, frei programmierbares Gerät zur Ermittlung, Verarbeitung und Übertragung von Maschinendaten. Für die Datenerfassung steht eine galvanisch nicht getrennte CAN-Schnittstelle zur Verfügung. Es gilt zu beachten, dass sich das EC-CAN-00x nur für Systeme mit einer kontinuierlichen Buslast bis zu 40% eignet. Des Weiteren ist das EC-CAN-00x mit einem Multi-GNSS-Modul für eine exakte Bestimmung der Position ausgestattet. Für die Nutzung dieser Funktion wird allerdings eine externe GNSS-Antenne benötigt. 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 Kunde kann mittels PAWN-Script (siehe "Pawn Script" auf Seite 85) seine eigene Applikation erstellen und diese per USB-Schnittstelle oder GPRS-Verbindung ins EC-CAN-00x laden. Dies erfolgt entweder über das rapidM2M Toolset (siehe "rapidM2M Toolset " auf Seite 63) oder über den Konfigurationsabschnitt "Steuerung" des myDatanet-Servers (siehe "Steuerung" auf Seite 57). Das PAWN-Script ermöglicht den Zugriff auf die CAN-Schnittstelle, wodurch der Kunde die Möglichkeit erhält, nahezu alle mit dieser Schnittstelle kompatiblen Geräte anzubinden und die entsprechenden Kommunikationsprotokolle zu implementieren.

> ***Wichtiger Hinweis:*** *Bei der USB-Schnittstelle handelt es sich um eine Serviceschnittstelle, die nur nach Öffnen des Gehäuses zugänglich ist. Geräteschäden wie z.B. durch Feuchtigkeitseintritt, die auf das Öffnen des Gehäuses zurückzuführen sind, entfallen aus der Haftung des Herstellers.*

Das EC-CAN-00x bietet dem Kunden einen Speicherbereich für seine Daten (3 MB) sowie 10 voneinander unabhängige Speicherblöcke für Konfigurationsdaten mit je 4000 Byte . Zudem verfügt das EC-CAN-00x noch über 4 Registrierungsspeicherblöcke mit je 1kB und vordefinierten Verwendungszwecken zur Ablage gerätespezifischer Daten (siehe "Registrierungsspeicherblöcke" auf Seite 36). Für das Anstoßen der Übertragung ist der Kunde selbst verantwortlich (siehe "rM2M_TxStart()"). Die Synchronisation der Konfigurations-, Registrierungs- und Messdaten mit dem Server wird vom System selbständig durchgeführt. Dazu ist das EC-CAN-00x mit einem integrierten SIM-Chip versehen.

6.8 Gerätekennzeichnung

Die Angaben in diesem Handbuch gelten ausschließlich für den Gerätetyp EC-CAN-00x .

> ***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.

6.8.1 Typenschilder

Die Typenschilder befinden sich auf der Oberseite des Geräts und beinhalten folgende Angaben:

- Typenbezeichnung
- Seriennummer
- Angaben zur Spannungsversorgung
- Hardwarerevision
- Länderlisten-Profil des SIM-Chips
- Schutzart
- Umgebungsbedingungen im Betrieb
- CE-Kennzeichnung
- Logo zur WEEE-Direktive der EU
- Chemische Zusammensetzung der verbauten Batterie

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



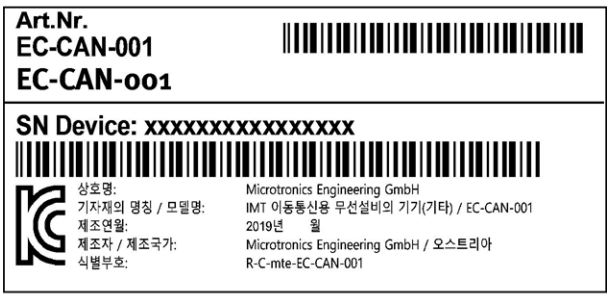
Typenschild EC-CAN-001



Typenschild EC-CAN-002

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

6.8.2 Verpackungsetiketten



Verpackungsetikett EC-CAN-001

6.9 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.

6.10 Aufbewahrung des Produkts

Zur Aufbewahrung des EC-CAN-00x stellen Sie sicher, dass alle relevanten Daten zum myDatanet-Server übertragen wurden. Gegebenenfalls lösen Sie dazu direkt am Gerät, falls Sie dies in Ihrem PAWN-Script vorgesehen haben, mittels Taste (siehe "Taste" auf Seite 55) 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. Trennen Sie zuerst das Gerät von der Versorgungsspannung indem Sie das Anschlusskabel vom M12-A Stecker lösen und entfernen Sie anschließend die Verkabelung der Antennen. Wenn möglich, sollte dabei die Versorgungsspannung abgeschaltet werden bevor Sie das Anschlusskabel vom M12-A Stecker lösen (siehe "Anschluss der Sensoren, der Aktoren und der Versorgung" auf Seite 46). Bewahren Sie das EC-CAN-00x in der Originalverpackung auf.

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 50).

6.11 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.

6.12 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

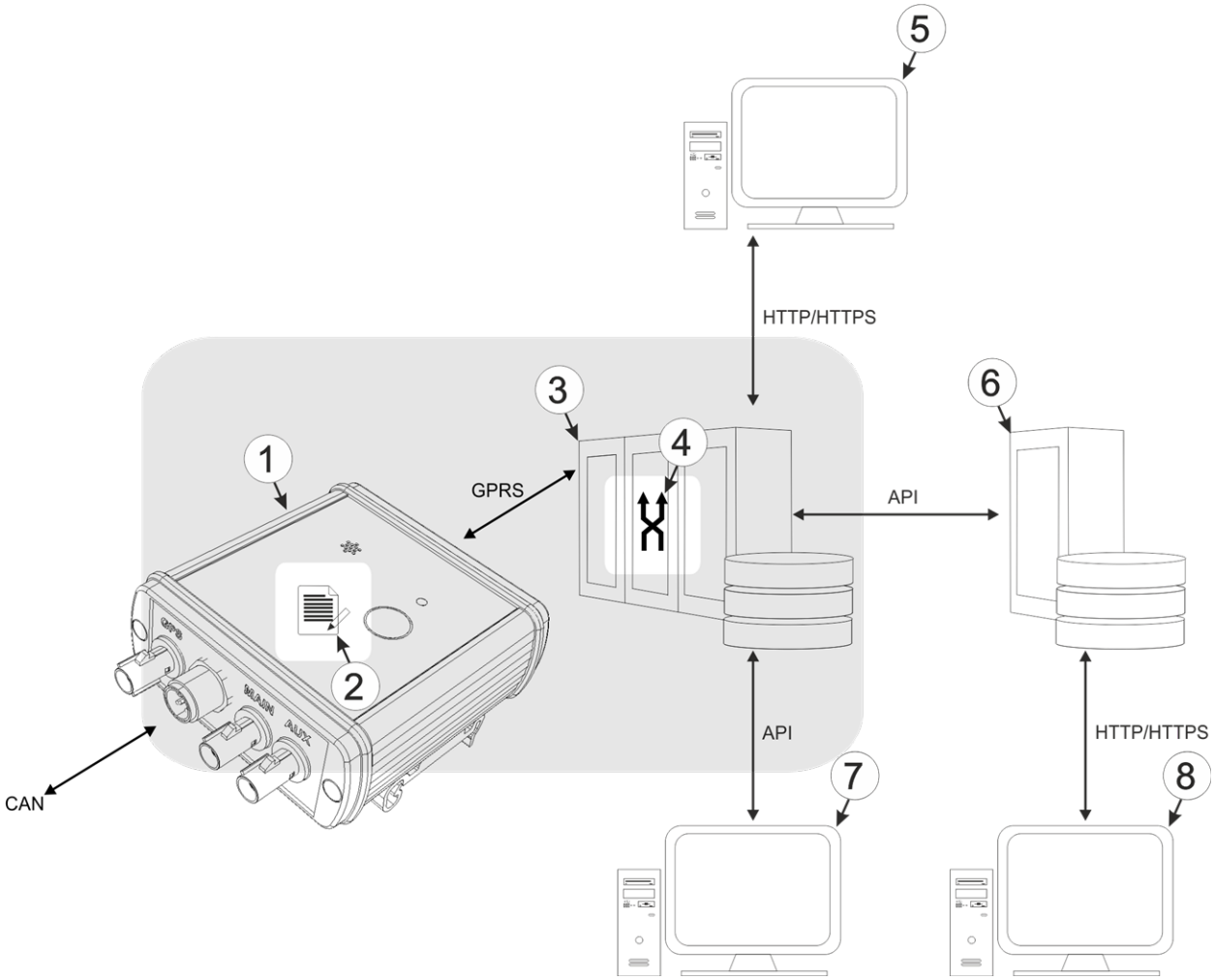
Hinweis: Qualifiziertes Fachpersonal

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 7 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	EC-CAN-00x mit integriertem Managed Service SIM-Chip (Datenübertragung inkludiert)
2	Durch den Kunden erstellte Applikation (Pawn Script), die sich um die Datenerfassung und -aufzeichnung kümmert (siehe "Pawn Script" auf Seite 85)
3	myDatanet-Server, zu dem die Daten übertragen werden
4	Durch den Kunden definierter myDatanet Connector , der die Nutzung der durch die Applikation (Pawn Script) erzeugten Messdaten und Konfigurationen in Verbindung mit der Oberfläche des myDatanet-Servers ermöglicht (siehe "myDatanet Connector " auf Seite 185)
5	Client, der mittels Web-Browser auf die Oberfläche des myDatanet-Servers zugreift
6	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 193).
7	Client, auf dem ein PC-Programm läuft, das seine Daten über die API-Schnittstelle des myDatanet-Servers (siehe "API" auf Seite 193) bezieht
8	Client, der mittels Web-Browser auf die Oberfläche des kundenspezifischen Servers zugreift

Funktionen und Komponenten, die durch myDatanet bereitgestellt werden:

- EC-CAN-00x

  Programmierbares Gerät (siehe "Pawn Script" auf Seite 85) mit integriertem Speicher und CAN-Schnittstelle (CAN) zur Anbindung von Maschinen, Sensoren und Aktoren an den myDatanet-Server (GPRS)

- Managed Service

  Das Managed Service ist die Basis für den Betrieb Ihrer Geräte und bietet Ihnen 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 des Servers (siehe "API" auf Seite 193) oder über 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 den im Kapitel "Technische Daten" (siehe "Technische Daten" auf Seite 17) aufgelisteten Spezifikationen kompatibel sind

- Applikation (Pawn Script)

  Die Firmware des EC-CAN-00x kümmert sich lediglich um die Synchronisation der Messdaten und Konfigurationen zwischen EC-CAN-00x 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 EC-CAN-00x (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.

- myDatanet Connector

  Die Basisfunktion des myDatanet-Servers beschränkt sich auf die Synchronisation der Rohmessdaten ("rm2mraw") und Konfigurationenblöcke ("config0" - "config9") zwischen EC-CAN-00x und Server. Der vom Kunden definierte myDatanet Connector muss sich um die Zuordnung der Rohmessdaten ("rm2mraw") zu den Messdatenkanälen ("histdata0" - "histdata9") sowie die Aufgliederung der Messdatenkanäle und Konfigurationsblöcke in die einzelnen Datenfelder kümmern. Die Zuordnung der Rohmessdaten ("rm2mraw") zu den Messdatenkanälen ("histdata0" - "histdata9") erfolgt mittels Split-tag (siehe "Split-tag" auf Seite 186), der Teil des myDatanet Connector ist. Eine Erklärung zur Aufgliederung der Messdatenkanäle und Konfigurationsblöcke in die einzelnen Datenfelder finden Sie im Kapitel "Datenstruktur" auf Seite 187.

- 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 193) vom myDatanet-Server gelesen.

7.1 Empfohlene Vorgehensweise

7.1.1 Entwicklung einer M2M / IoT Anwendung

Es wird empfohlen bei der Entwicklung einer M2M / IoT Anwendung mit der Definition des myDatanet Connector (siehe "myDatanet Connector " auf Seite 185) zu beginnen. Durch ihn werden die verschiedenen Datenstrukturen (Messdaten, Konfigurationen usw.) festgelegt, die sowohl für das Script 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 myDatanet Connector .

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.*

7.2 Funktionsweise des internen Datenspeichers

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

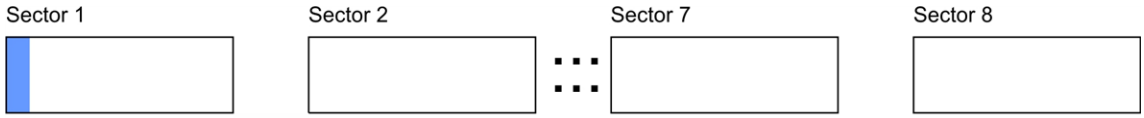
Der interne Datenspeicher des EC-CAN-00x ist als Ringspeicher mit 8 Sektoren aufgebaut. Wurde der gesamte Speicher (3 MB) zur Gänze beschrieben, wird der Sektor mit den ältesten Daten vollständig gelöscht, bevor wieder neue Daten in diesen Sektor gespeichert werden können. D.h. der interne Datenspeicher 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 10 Byte beträgt und die ersten 8 Byte eines jeden Sektors für die interne Verwaltung reserviert sind. Die 10 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.

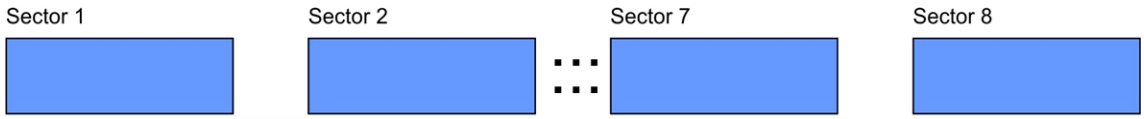
Hinweis:

Ergänzende Erklärung zur Funktionsweise des Ringspeichers

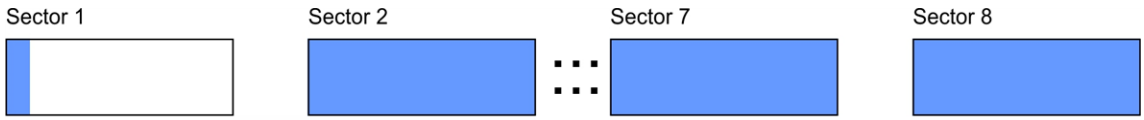
Datenspeicher nach der ersten Datenaufzeichnung:



Datenspeicher nachdem 3 MB aufgezeichnet wurden



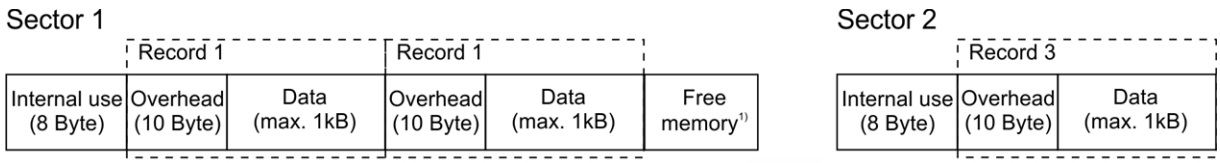
Datenspeicher, wenn nachdem bereits 3 MB aufgezeichnet wurden eine weitere Datenaufzeichnung erfolgte



Hinweis:

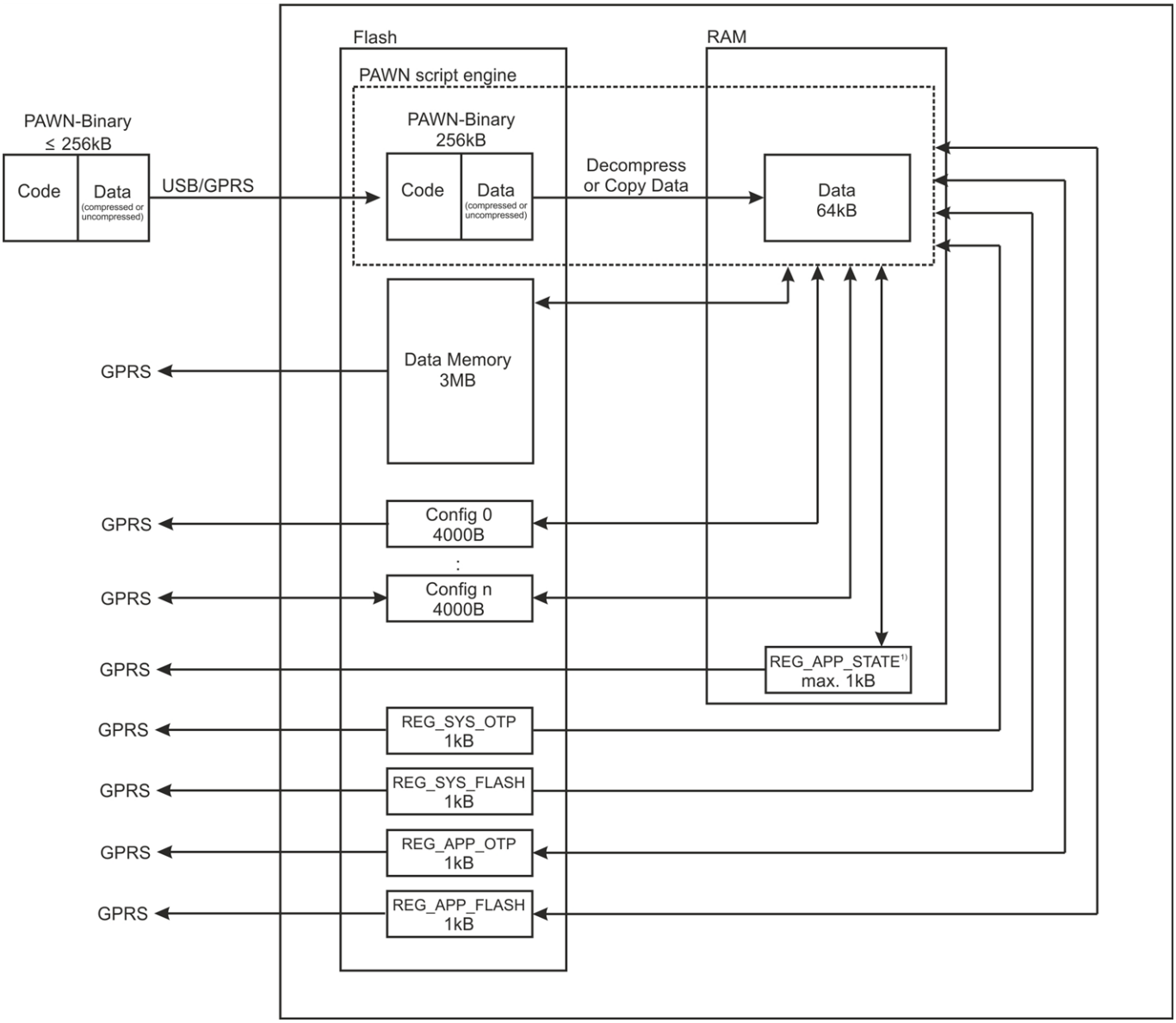
Ergänzende Erklärung zur Berechnung der zu speichernden Datenmenge:

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

7.3 Speicherorganisation



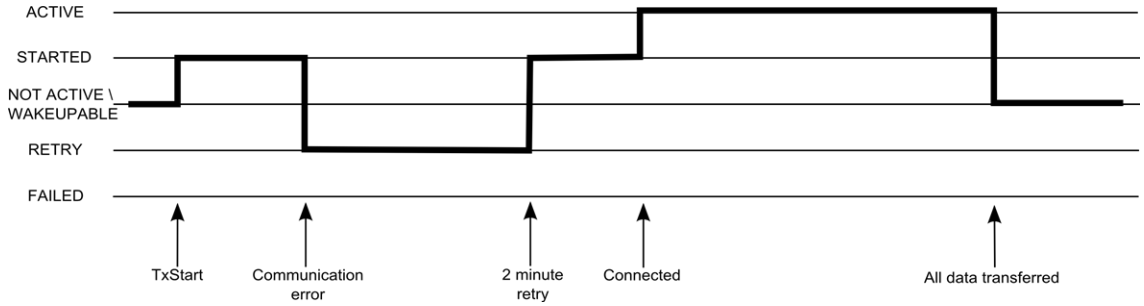
Organisation des Speichers des EC-CAN-00x

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

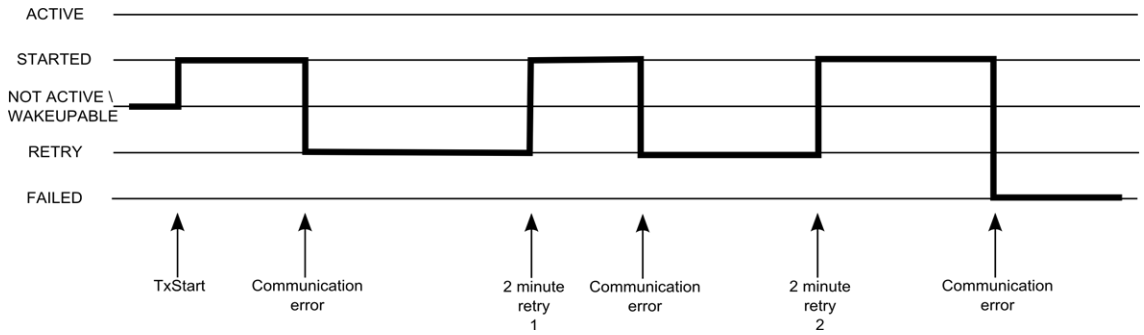
Für die Übertragung ins EC-CAN-00x darf die Größe des PAWN-Binarys 256kB nicht überschreiten. Falls nötig, kann der Data-Bereich des PAWN-Binary mittels Compiler-Anweisung (#pragma amxcompress <0-3>) komprimiert werden. Im Flash Speicher des EC-CAN-00x sind das PAWN-Binary (256kB), die 10 Konfigurationsblöcke (je 4000 Byte), die 4 Registrierungsspeicherblöcke (je 1kB) und die Messdaten (3 MB) abgelegt. Für die Ausführung durch die PAWN Script-Engine wird der Data-Bereich des PAWN-Binarys falls nötig dekomprimiert und ins RAM kopiert. Die maximale Größe für den dekomprimierten Data-Bereich des PAWN-Binarys im RAM beträgt 64kB . Im RAM werden auch die optional mittels der Funktion "rM2M_ RegInit()" initialisierbaren Registrierungsspeicherblöcke (z.B. REG_APP_STATE) abgelegt.

7.4 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.

7.4.1 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.

7.4.2 Verbindungsabbruch während eines Scriptdownloads

Auf einen Verbindungsabbruch während des Scriptdownloads reagiert das Gerät mit der Standard Retry-Sequenz (2 Verbindungsaufbauversuche im Abstand von 2min.). Zusätzlich dazu wird, sobald das Gerät den Verbindungsabburch erkannt hat, der Log-Eintrag "SCRIPT_ERR, SCRIPT UPDATE ERROR" ins Gerätelog eingetragen. Das bestehende Script kann dadurch, dass es 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 Scriptdownloads der Bereich im Flash-Speicher in dem sich das Script befindet, gelöscht wird.

7.5 Automatische Auswahl des GSM-Netzes

Da das EC-CAN-00x 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.

7.6 Errorhandling

Um zu gewährleisten, dass bei Problemen mit dem Script eine Diagnose bzw. Behebung aus der Ferne möglich ist, wurden folgende Übertragungmechanismen in die Firmware integriert. Für den Fall, dass kein Script vorhanden ist, erfolgt die Verbindung zum myDatanet-Server alle 24h . Sollte ein vorhandenes Script 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).

7.7 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:** Script, myDatanet-Server	FLASH	Systeminformationen, die einmalig im Zuge der Produktion geschrieben werden
REG_SYS_FLASH [1)]	**lesbar:** Script, myDatanet-Server	FLASH	Systeminformationen, die im Betrieb veränderbar sein müssen
Applikationsspezifische Daten			
REG_APP_OTP	**lesbar:** Script, myDatanet-Server **schreibbar:** Script	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:** Script, myDatanet-Server **schreibbar:** Script	FLASH	applikationsspezifische Informationen, die im Betrieb veränderbar sein müssen
Applikationsspezifische, flüchtige Daten			
REG_APP_STATE [2)]	**lesbar:** Script, myDatanet-Server **schreibbar:** Script	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 213) 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.

7.7.1 REG_SYS_FLASH

Sollte die Messstelle, die mit dem Gerät verknüpft ist aus einer Applikations-Vorlage (siehe Benutzerhandbuch für myDatanet-Server 206.886) erstellt worden sein, enthält dieser Registrierungsspeicherblock folgende Felder:

latestAppVersion
Die aktuellste verfügbare Version der im Gerät installierten Applikations-Vorlage [16-Bit Unsigned]

installedAppVersion
Momentan im Gerät installierte Version der Applikations-Vorlage [16-Bit Unsigned]

appId
Applikations-ID der im Gerät installierten Applikation [max. 50 Zeichen]

7.7.2 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
Applikations-ID der Applikations-Vorlage aus der die Messstelle erstellt werden soll [max. 50 Zeichen].

pipAppVer (optional)
Momentan im Gerät installierte Version des Scripts (z.B. "01v005") [6 Zeichen].

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.

7.8 File Transfer

Es ist möglich bis zu 20 Dateien für den File Transfer zu registrieren (siehe "FT_Register()"). Dabei muss der Funktion "FT_Register()" eine Callback Funktion (siehe "Callback Funktionen" auf Seite 156) 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 155) 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.

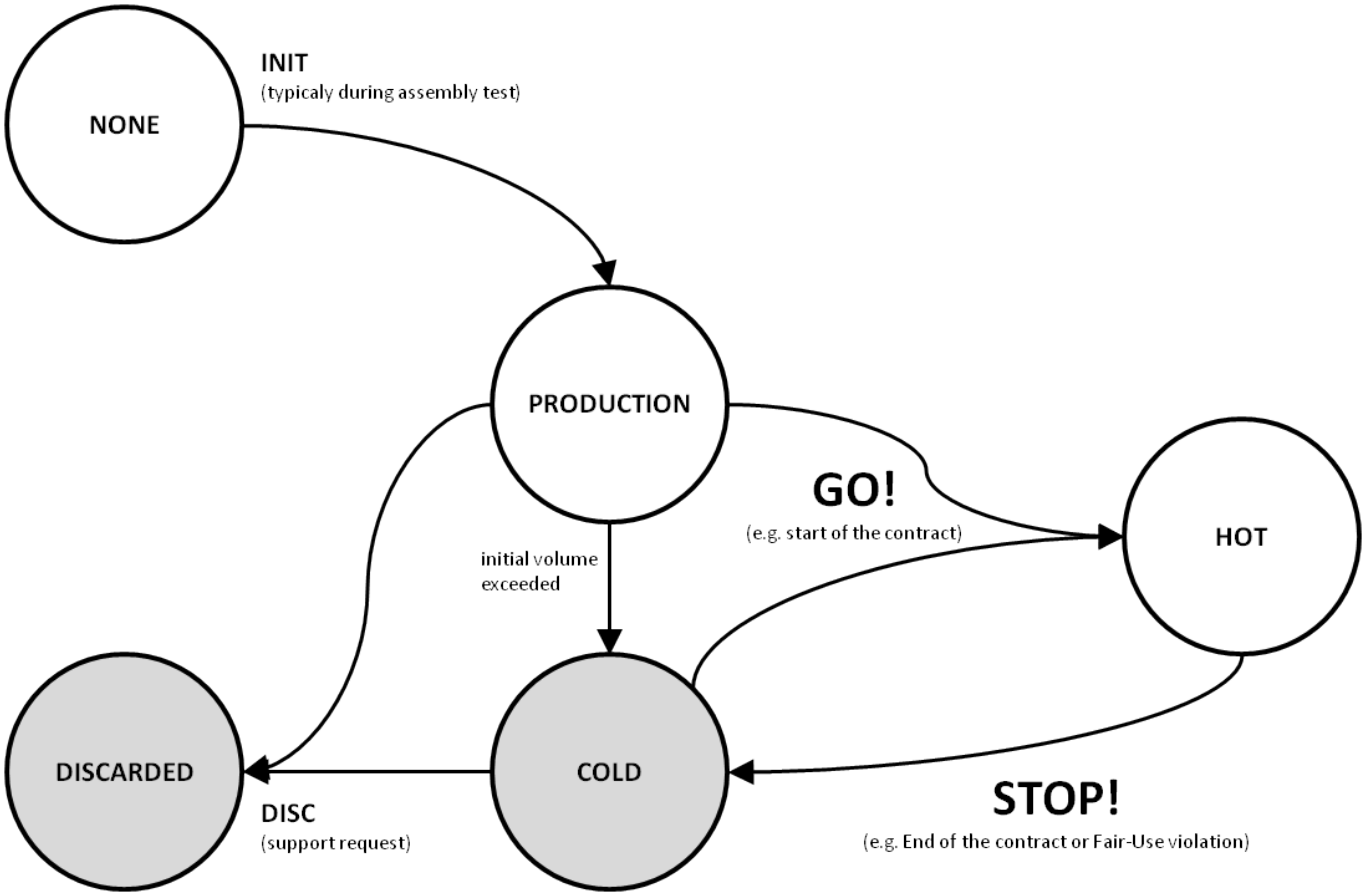
Beim Empfang eines File Transfer Kommandos wird eine Sitzung gestartet, welche nach 15sec. automatisch beendet wird, wenn das Script das Kommando nicht korrekt behandelt. Um Konflikte zu vermeiden, kann immer nur eine Sitzung nach der anderen aktiv sein.

7.9 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 Script	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 Script 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 213) 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 8 Lagerung, Lieferung und Transport

8.1 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 213) gerichtet werden.

Hinweis: *Später eingehende Reklamationen werden nicht anerkannt!*

8.2 Lieferumfang

Zum Standardlieferumfang des EC-CAN-001 (300825) gehören:

- EC-CAN-001
- Kombiantenne Mobilfunk (3G)/GNSS (GPS/GALILEO)
- Anschlusskabel 1m (M12-A Buchse auf offene Kabelenden)
- Staubschutzkappe

Zum Standardlieferumfang des EC-CAN-002 (300837) gehören:

- EC-CAN-002
- Kombiantenne Mobilfunk (4G)/GNSS (GPS/GALILEO)
- Anschlusskabel 1m (M12-A Buchse auf offene Kabelenden)
- Staubschutzkappe

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

8.3 Lagerung

Folgende Lagerbedingungen sind unbedingt einzuhalten:

EC-CAN-00x	Lagertemperatur	-20...+60°C
	Feuchte	15...90%rH

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

8.4 Transport

Schützen Sie das EC-CAN-00x vor starken Stößen, Schlägen, Erschütterungen oder Vibrationen. Der Transport muss in der Originalverpackung erfolgen.

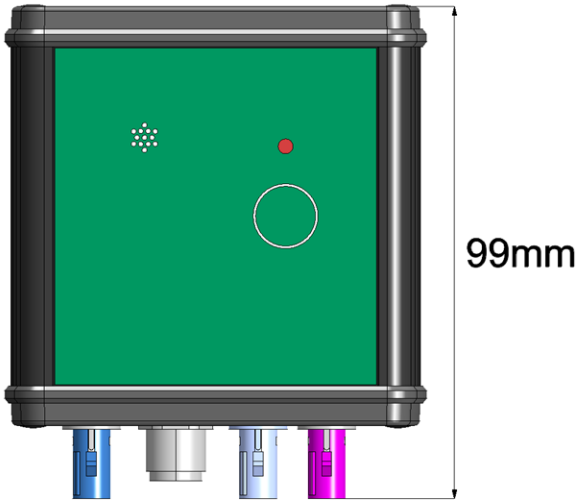
8.5 Rücksendung

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 213). Die Rücksendung des EC-CAN-00x muss in der Originalverpackung frachtfrei zu Microtronics Engineering GmbH (siehe "Kontaktinformationen" auf Seite 213) erfolgen. Nicht ausreichend frei gemachte Sendungen werden nicht angenommen!

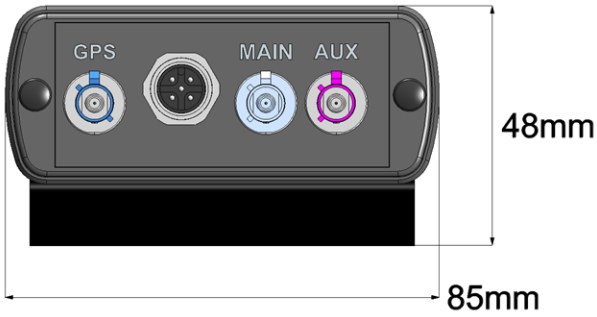
Kapitel 9 Installation

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

9.1 Abmessungen



Abmessungen: Höhe



Abmessungen: Breite und Tiefe

9.2 Montage des EC-CAN-00x

Wichtiger Hinweis:

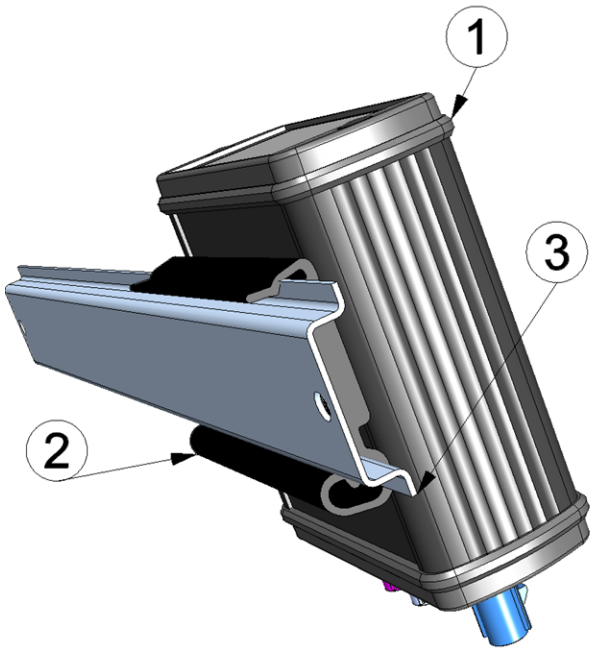
- *Achten Sie auf eine sachgemäße Montage!*
- *Befolgen Sie bestehende gesetzliche bzw. betriebliche Richtlinien!*
- *Unsachgemäße Handhabung kann zu Verletzungen und/oder Beschädigungen an den Instrumenten führen!*
- *Der Druckausgleich muss vor Verschmutzung geschützt werden.*

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 dem Gerät sollten Sie ca. 8 cm Abstand für die Kabelanschlüsse vorsehen. Weitere Informationen zu den Abmessungen für die Montage entnehmen Sie dem jeweiligen Unterkapitel.*

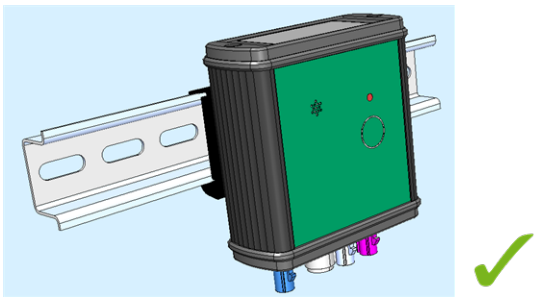
9.2.1 Hutschienenmontage

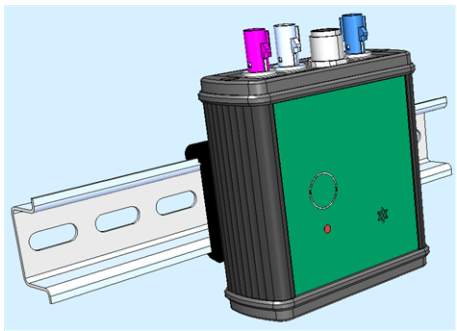


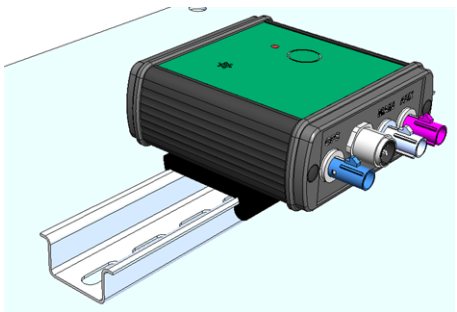
Hutschienenmontage

1 EC-CAN-00x	**3** Hutschiene
2 Montagelasche	

Wichtiger Hinweis: *Das EC-CAN-00x ist in der Weise zu montieren, dass die Anschlussstecker nach unten gerichtet sind. Sämtliche anderen Montagelagen sind nicht zulässig.*







1. Setzen Sie die Montagelasche (2) auf die Unterkante der Hutschiene auf. Durch eine leichte Drehung um die Horizontalachse des EC-CAN-00x rastet die Montagelasche (2) dann auf der Hutschiene ein (siehe Abbildung "Hutschienenmontage" auf Seite 44).

   Wichtiger Hinweis: *Es ist nicht zulässig die Montagelasche (2) zu entfernen oder die Schrauben, mittels derer die Montagelasche befestigt ist, zu lösen, da die Schrauben durch das Gehäuse ragen und bei der Produktion von innen entsprechend abgedichtet wurden. Beim Lösen der Schrauben wird diese Abdichtung beeinträchtigt und der Schutz gegen Eindringen von Feuchtigkeit in das Gerät ist nicht mehr gewährleistet.*

9.3 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 das EC-CAN-00x 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 19.

9.3.1 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.

9.4 Elektrische Installation

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

9.4.1 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.*
- *Um die Dichtheit des Gehäuses zu gewährleisten müssen sämtliche Stecker mit entsprechenden Gegenstücken verbunden oder mit den im Lieferumfang enthaltenen Schutzkappen versehen werden.*
- *Das Gerät muss mit einer T2A Vorsicherung (Pin2 des 5pol. M12-A Steckers für die Versorgung und den CAN-Bus) abgesichert werden.*

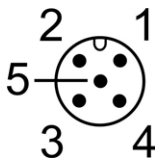


Anschluss der Sensoren, der Aktoren und der Versorgung

1	Anschluss für die GNSS-Antenne	**3**	Anschluss für die primäre Mobilfunkantenne (MAIN)
2	Anschluss für die Versorgung und den CAN-Bus (5pol. M12-A)	**4**	Anschluss für die sekundäre Mobilfunkantenne (AUX)[1)]

[1)] als Vorbereitung für Module der Kategorien LTE CAT-1 und höher

Belegung des Anschlusses für die Versorgung und den CAN-Bus

	5pol. M12-A Stecker	Anschlusskabel 1m	Signal
	1	braun	n.c.
	2	weiß	24VDC Versorgungsspannung
	3	blau	Masse
	4	schwarz	CAN High Leitung der CAN-Schnittstelle
	5	grau	CAN Low Leitung der CAN-Schnittstelle

1. Verbinden Sie Ihre Maschinen, Sensoren oder Aktoren sowie Ihre Energiequelle mit dem Anschluss für die Versorgung und den CAN-Bus. Sie können dazu das im Lieferumfang enthaltene Anschlusskabel mit offenen Enden verwenden. Achten Sie dabei auf Stromlosigkeit!

   Wichtiger Hinweis: *Bitte beachten Sie, dass es erforderlich ist, die 24VDC Versorgungsspannung (Pin2 des 5pol. M12-A Steckers) mittels T2A Sicherung (siehe "Blockschaltbild" auf Seite 23) abzusichern.*

2. Schließen Sie die Antennenkabel an (siehe "Anschluss der GSM-Antenne" auf Seite 47). Bei der im Lieferumfang enthaltenen Antenne handelt es sich um eine Kombiantenne für Mobilfunk/GNSS mit zwei Anschlusskabeln. Verbinden Sie das Kabel mit dem blauen Stecker (FAKRA Kodierung C) mit dem Anschluss für die GNSS-Antenne und das Kabel mit dem violetten Stecker (FAKRA Kodierung D) mit dem Anschluss für die primäre Mobilfunkantenne (MAIN).

3. Versehen Sie alle nicht benötigten Stecker mit entsprechenden Schutzkappen.

   Wichtiger Hinweis: *Alle nicht verwendeten Stecker des EC-CAN-00x müssen mit Hilfe der mitgelieferten Schutzkappen wasserdicht verschlossen werden. Andernfalls ist der Schutzgrad des gesamten Geräts nicht gegeben und die Gewährleistung des Herstellers erlischt.*

4. Schalten Sie die 24VDC Versorgungsspannung des EC-CAN-00x ein.

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 53).

9.4.2 Anschluss der GSM-Antenne

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

Die Kabel der Standardantenne werden direkt mit den Antennenanschlüssen (siehe "Übersicht" auf Seite 22) des EC-CAN-00x verbunden.

1. Stellen Sie sicher, dass das EC-CAN-00x spannungslos ist.

2. Verbinden Sie die Anschlusskabel der Antenne mit den entsprechenden Anschlüssen des EC-CAN-00x (siehe "Übersicht" auf Seite 22). Bei der im Lieferumfang enthaltenen Antenne handelt es sich um eine Kombiantenne für Mobilfunk/GNSS mit zwei Anschlusskabeln. Verbinden Sie das Kabel mit dem blauen Stecker (FAKRA Kodierung C) mit dem Anschluss für die GNSS-Antenne und das Kabel mit dem violetten Stecker (FAKRA Kodierung D) mit dem Anschluss für die primäre Mobilfunkantenne (MAIN).

3. Stellen Sie die Spannungsversorgung des EC-CAN-00x wieder her.

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 53).

9.4.3 Technische Details zur CAN-Schnittstelle

Hinweis: *Die CAN-Schnittstelle des EC-CAN-00x ist kompatibel zur Norm ISO-11898, einschließlich der Anforderungen für 24V.*

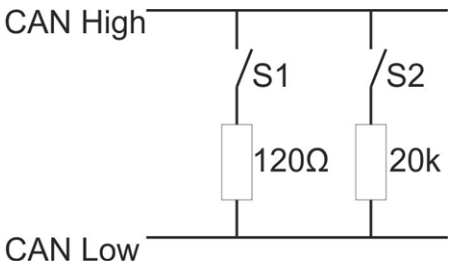
unterstützte CAN Spezifikation	V2.0B
Datenübertragungsrate	max. 1Mbit/s
zulässige kontinuierliche Buslast	40%
gleichzeitig anbindbare CAN Knoten	max. 112

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 EC-CAN-00x sowohl am Ende als auch mittels Stichleitungen an den CAN-Bus anzubinden. Wird das EC-CAN-00x am Ende des Busses anstelle des vorgeschriebenen 120Ω Abschlusswiderstandes angeschlossen, muss mittels Schalter 1 des Dip-Switches SW1 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 20k Widerstand mittels Schalter 2 des Dip-Switches SW1 zu aktivieren. Im Auslieferungszustand ist keiner der beiden Widerstände aktiv. Der Dip-Switche SW1 (siehe "Technische Details zur CAN-Schnittstelle" auf Seite 48) ist nur nach Öffnen des Gehäuses zugänglich.

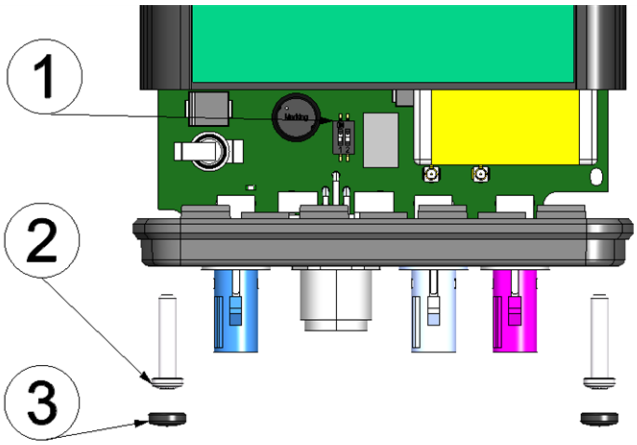
Wichtiger Hinweis: *Geräteschäden wie z.B. durch Feuchtigkeitseintritt, die auf das Öffnen und anschließendes unsachgemäßes Wiederverschließen des Gehäuses zurückzuführen sind entfallen aus der Haftung des Herstellers.*

Dip-Switches SW1

S1	120Ω Abschlusswiderstand zw. CAN High und CAN Low
S2	20k Abschlusswiderstand zw. CAN High und CAN Low



Prinzipschaltbild zu den zuschaltbaren Widerständen



Position des Dip-Switches SW1

1 Dip-Switches SW1	**3** Abdeckkappe
2 Schraube 30x10 TOAX 10	

9.4.4 Technische Details zur USB-Schnittstelle

> ***Wichtiger Hinweis:*** *Bei der USB-Schnittstelle handelt es sich um eine Serviceschnittstelle, die nur nach Öffnen des Gehäuses zugänglich ist. Geräteschäden wie z.B. durch Feuchtigkeitseintritt, die auf das Öffnen des Gehäuses zurückzuführen sind, entfallen aus der Haftung des Herstellers.*

Über die USB-Slave-Schnittstelle wird die Verbindung zu einem PC hergestellt. Dabei ist ausschließlich die Kommunikation mit der Entwicklungsumgebung rapidM2M Toolset vorgesehen. Es ist nicht möglich mittels PAWN-Script auf die USB-Schnittstelle zuzugreifen. Sowohl das rapidM2M Toolset , als auch die USB-Schnittstelle werden nur während der Entwicklungsphase eines neuen Scripts benötigt. Eine detaillierte Beschreibung der Entwicklungsumgebung rapidM2M Toolset finden Sie im Kapitel "rapidM2M Toolset " auf Seite 63. Das rapidM2M Toolset steht unter folgender Adresse gratis zum Download bereit:

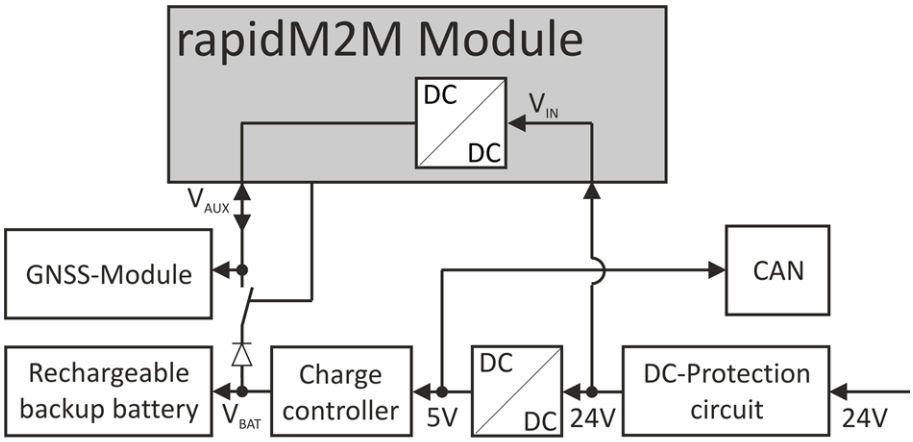
www.microtronics.com/Toolset

9.4.5 Technische Details zum Energiemanagement

Fällt die externe 24VDC Versorgungsspanng aus und wird durch das Script rechtzeitig auf den integrierten Pufferakku umgeschaltet, arbeitet das Gerät bis 3,4V normal. Ab dieser Schwelle wird das Modem deaktiviert und der Log-Eintrag "UV MODEMLOCKOUT" ins Gerätelog eingetragen. D.h. sollte sich das Gerät im "online"-Modus befinden oder ins GSM-Netzwerk eingebucht sein (Modus "Intervall & Wakeup"), wird die Verbindung getrennt. Wird nach Erreichen dieser Schwelle versucht die Verbindung mittels der Funktion "rM2M_TxStart()" aufzubauen, liefert die Funktion als Ergebnis "ERROR". Ebenso liefert die Funktion "rM2M_TxSetMode()" als Ergebnis "ERROR", wenn versucht wird den "Intervall & Wakeup"- oder "online"-Modus zu aktivieren. Das Script wird nach Erreichen dieser Schwelle weiterhin normal ausgeführt. Sinkt die interne Spannung jedoch unter 2,8V , ist eine zuverlässige Ausführung des Scripts nicht mehr gewährleistet. Daher empfiehlt es sich bei Erreichen dieser Schwelle den Pufferakku mittels der Script-Funktion tbd zu deaktivieren. Dadurch werden sowohl die Tiefentladung des Pufferakkus als auch unzulässige Programmzustände vermieden. Sobald die externe Versorgungsspannung erneut verfügbar ist, nimmt das EC-CAN-00x den Betrieb gemäß Konfiguration wieder auf und die Laderegelung sorgt für die Wiederaufladung des Pufferakkus. Die Laderegelung ist permanent aktiv und arbeitet unabhängig von Firmware oder Script. Sie umfasst Sicherheitsfunktionen wie die Überwachung der Akkutemperatur und der Ladezeit. Bei Verfügbarkeit der externen Versorgungsspannung versucht die Laderegelung den Pufferakku

auf 4,2V zu laden. Sobald der Pufferakku vollständig geladen ist, beginnt die Laderegelung mit der Überwachung der Spannung am Pufferakku und startet den Ladevorgang erneut, sobald diese unter 4,1V fällt.

9.4.6 Technische Details zur Energieversorgung



Prinzipschaltbild der Energieversorgung

Spannungsversorgung	24VDC (+5/-10%)
Leistungsaufnahme	typ. 0,5W max. 3,5W
Verpolungsschutz	Ja
V_{AUX}	3,82V
V_{BAT}	4,2V

Der Versorgungsspannungseingang ist mit einer Diode zum Schutz vor Verpolung ausgestattet. Die 24VDC Versorgungsspanng ist sowohl mit dem DC/DC-Wandler, der die 5V für die Speisung des CAN-Moduls und des Ladereglers für den Pufferakku erzeugt, als auch mit dem am rapidM2M Modul verbauten DC/DC-Wandler, der V_{AUX} für die Versorgung des rapidM2M Moduls und des GNSS-Moduls bereitstellt, verbunden. Mittels der Script-Funktion tbd kann bei Ausfall der 24VDC Versorgungsspanng V_{AUX} auf die Speisung über den integrierten Pufferakku umgeschaltet werden. In diesem Zustand ist die CAN-Schnittstelle nicht mehr verfügbar. Alle anderen Funktionen bleiben jedoch erhalten. Eine Diode verhindert, dass der Pufferakku über V_{AUX} geladen wird, wenn auf den Pufferakku umgeschaltet wird, aber die 24VDC Versorgungsspanng noch vorhanden ist.

> ***Wichtiger Hinweis:*** *Wird der Pufferakku über längere Zeit aktiviert, obwohl die 24VDC Versorgungsspanng verfügbar ist, kommt es durch die permanente Wiederaufladung (da V_{AUX} mit 3,82V unter der Schwelle von 4,1V liegt ab der der Ladevorgang gestartet wird) zu einem deutlich erhöhten Verschleiß des Pufferakkus.*

9.4.7 Technische Details zur Systemzeit

Der EC-CAN-00x 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 10 Inbetriebnahme

10.1 Hinweise an den Benutzer

Bevor Sie das EC-CAN-00x 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 EC-CAN-00x 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 213).

Für die Inbetriebnahme des Gesamtsystems sind eventuell die Handbücher der Zubehörteile ebenfalls hinzu zu ziehen. Diese sind im Lieferumfang der Zubehörteile enthalten.

10.2 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 EC-CAN-00x und den Eingabemasken des myDatanet-Servers vertraut, bevor Sie mit der Konfiguration beginnen.

10.3 Inbetriebnahme des Systems

Hinweis: *Es empfiehlt sich das EC-CAN-00x zuerst im Büro in Betrieb zu nehmen bevor Sie das Gerät am Einsatzort fix montieren. Dabei sollten Sie gleich eine Messstelle für den späteren Betrieb am myDatanet–Server anlegen (siehe "Anlegen der Messstelle" auf Seite 81) und eine Messstellenkonfiguration (inkl. Connector und Script) festlegen (siehe "Messstellenkonfiguration" auf Seite 56). Wenn Sie die Messstelle aus einer Applikations-Vorlage (siehe "Benutzerhandbuch für myDatanet-Server " 206.886) erstellen, werden der Connector und das Script aus der Vorlage 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 Maschinen, Sensoren oder Aktoren verwenden, um die Konfiguration des EC-CAN-00x 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 Messstelle / Applikation für den Betrieb am myDatanet-Server an (siehe "Anlegen der Messstelle" auf Seite 81).

   Hinweis: *Für den Betrieb des EC-CAN-00x muss eine Messstelle des Typs "rapidM2M" oder eine Messstelle aus einer zum Messstellentyp"rapidM2M" kompatiblen Applikations-Vorlage erstellt werden.*

3. Konfigurieren Sie die erstellte Messstelle / Applikation entsprechend Ihren Anforderungen (siehe "Messstellenkonfiguration" auf Seite 56). Wurde die Messstelle nicht aus einer Applikations-Vorlage erstellt, müssen Sie auch noch den Connector und das Script über den Konfigurationsabschnitt "Steuerung" festlegen (siehe "Steuerung" auf Seite 57).
4. Schließen Sie die Antennenkabel an (siehe "Anschluss der GSM-Antenne" auf Seite 47).
5. Lösen Sie einen Verbindungsaufbau aus. Falls noch kein Script 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 ein Script ins Gerät geladen, führen Sie bitte die im Script vorgesehenen Operationen zum Auslösen des Verbindungsaufbaus aus.
6. Verschieben Sie das EC-CAN-00x vom Pool in den gewünschten Kunden (siehe "Gerät dem Kunden zuweisen" auf Seite 83).
7. Verknüpfen Sie das EC-CAN-00x mit der erstellten Messstelle / Applikation (siehe "Site" auf Seite 56 bzw. "Anlegen der Messstelle" auf Seite 81).
8. Lösen Sie den Verbindungsaufbau erneut aus (z.B. durch Trennen und erneutes Herstellen der Spannungsversorgung), damit die Konfiguration der Messstelle zum EC-CAN-00x übertragen wird.

   Hinweis: *Diesen Schritt können Sie auch überspringen, da bei der Installation vor Ort ebenfalls eine Verbindung ausgelöst werden sollte, wodurch die Konfiguration dann zu diesem Zeitpunkt zum EC-CAN-00x übertragen wird.*

9. Trennen Sie anschließend die Verkabelung der Versorgungsspannung, möglichst im stromlosen Zustand, vom Gerät.
10. Lösen Sie die Anschlusskabel der Antenne wieder vom EC-CAN-00x .

Folgende Arbeiten werden direkt am Einsatzort des Geräts durchgeführt:

11. Führen Sie alle im Kapitel "Anschluss der Sensoren, der Aktoren und der Versorgung" auf Seite 46 beschriebenen Schritte durch.
12. Überprüfen Sie, ob die Verbindung zum myDatanet-Server korrekt funktioniert (siehe "Kommunikation mit dem Gerät testen" auf Seite 53).

10.4 Kommunikation mit dem Gerät testen

1. Legen Sie eine Messstelle für den Betrieb am myDatanet-Server an (siehe "Anlegen der Messstelle" auf Seite 81).

   Hinweis: *Für den Betrieb des EC-CAN-00x muss eine Messstelle des Typs "rapidM2M" oder eine Messstelle aus einer zum Messstellentyp"rapidM2M" kompatiblen Applikations-Vorlage erstellt werden.*

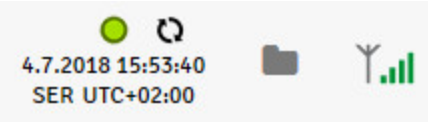
2. Konfigurieren Sie die erstellte Messstelle / Applikation entsprechend Ihren Anforderungen (siehe "Messstellenkonfiguration" auf Seite 56). Wurde die Messstelle nicht aus einer Applikations-Vorlage erstellt, müssen Sie auch noch den Connector und das Script über den Konfigurationsabschnitt "Steuerung" festlegen (siehe "Steuerung" auf Seite 57).

3. Verknüpfen Sie das EC-CAN-00x mit der erstellten Messstelle (siehe "Site" auf Seite 56 bzw. "Anlegen der Messstelle" auf Seite 81).

4. Schließen Sie die Antennenkabel an (siehe "Anschluss der GSM-Antenne" auf Seite 47).

5. Lösen Sie einen Verbindungsaufbau aus. Falls noch kein Script 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 ein Script ins Gerät geladen, führen Sie bitte die im Script vorgesehenen Operationen zum Auslösen des Verbindungsaufbaus aus.

6. 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 das EC-CAN-00x 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.

7. Führen Sie alle im Kapitel "Anschluss der Sensoren, der Aktoren und der Versorgung" auf Seite 46 beschriebenen Schritte durch.

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

8. Für die Überprüfung der Datenübertragung können Sie die "Auswertungen" des myDatanet-Servers verwenden (siehe"Benutzerhandbuch für myDatanet-Server " 206.886). Dazu ist allerdings die Konfiguration des myDatanet Connector (siehe "myDatanet Connector " auf Seite 185) erforderlich.

9. Nachdem Sie die erforderlichen Vorbereitungen getroffen haben, lösen Sie direkt am Gerät eine Übertragung aus. Falls Sie keine Möglichkeit zum Auslösen einer Übertragung in Ihrem Pawn Script vorgesehen haben, warten Sie bis zur nächsten planmäßigen Datenübertragung.

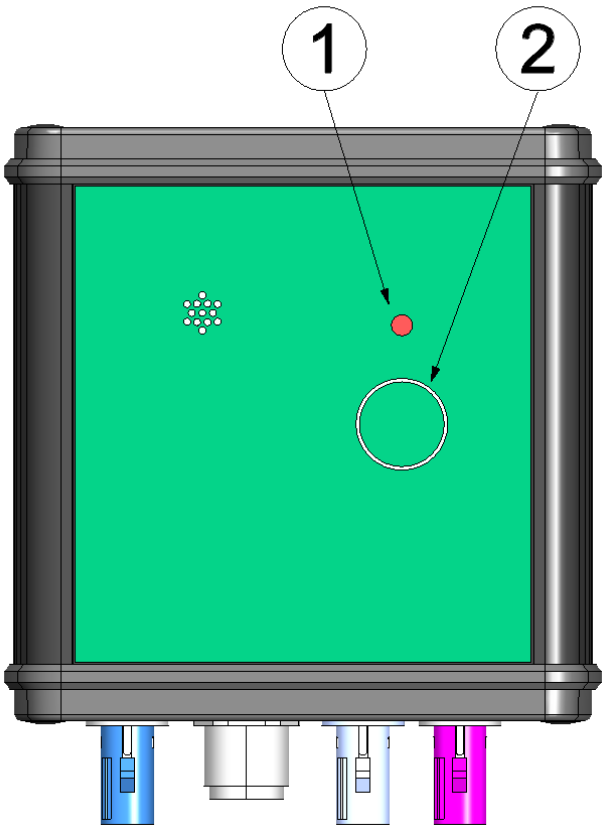
10. Werten Sie die ankommenden Daten aus.

Kapitel 11 Benutzerschnittstellen

Die Konfiguration des EC-CAN-00x erfolgt über das Web-Interface am myDatanet-Server (siehe "Benutzerschnittstelle am myDatanet-Server" auf Seite 56), dessen Web-Adresse Sie von Ihrem zuständigen Vertriebspartner erhalten.

11.1 Benutzerschnittstelle am EC-CAN-00x

11.1.1 Bedienelemente



Bedienelemente

1 LED	**2** Taste

11.1.1.1 Taste

Die Taste des EC-CAN-00x ist mit INT1 (X1-Pin31) des eingesetzten rapidM2M Moduls verbunden. Zur Verwendung der Taste muss sowohl der Interrupt mittels der Script-Funktion rM2M_IrqInit() aktiviert als auch die Callback Funktion erstellt werden, die im Falle eines Interrupts durch das System aufgerufen werden soll. Bei der Initialisierung des Interrupts kann angegeben werden, ob der Interrupt bei steigender oder fallender Flanke ausgelöst werden soll. Eine gleichzeitige Überwachung beider Flanken ist nicht möglich. D.h. bei der Entwicklung der Applikation gilt es die Entscheidung zu treffen, ob eine Aktion bereits beim Drücken der Taste ausgelöst werden soll oder erste wenn die Taste wieder losgelassen wird. Zur Unterdrückung ungewollter mehrfacher Impulse beim Tastendruck verfügt das EC-CAN-00x über eine Schaltung zur Entprellung.

11.1.1.2 LED

Die LED des EC-CAN-00x ist mit dem GPIO1 (X1-Pin23) des eingesetzten rapidM2M Moduls verbunden. Die Steuerung der LED erfolgt mittels der Script-Funktion "rM2M_GpioSet()". Zuvor muss jedoch die Signalrichtung des GPIO1 mittels der Script-Funktion "rM2M_GpioDir()" auf "Ausgang" gesetzt werden.

Signalrichtung	Level	LED
Eingang	---	leuchtet
Eingang	---	leuchtet
Ausgang	low	aus
Ausgang	high	leuchtet

11.2 Benutzerschnittstelle am myDatanet-Server

11.2.1 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).

11.2.1.1 Site

Kunde
gibt an, welchem Kunden die Messstelle zugeordnet ist

-Symbol
Messstelle einem anderen Kunden zuweisen

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

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

Applikations-Vorlage
Name der Applikations-Vorlage aus der die Messstelle erstellt wurde

Tags
Liste der Tags, die der Messstelle 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.

11.2.1.2 Kommentar

Kommentar
freies Kommentarfeld (wird auch unterhalb des Gerätetyps in der Messstellenliste angezeigt)

11.2.1.3 Steuerung

Hinweis: *Wurde die Messstelle aus einer Applikations-Vorlage (siehe "Benutzerhandbuch für myDatanet-Server " 206.886) erstellt, ist dieser Konfigurationsabschnitt nicht sichtbar.*

Script Typ	aus	Script deaktiviert	
	Pawn	Aktiviert die Scriptabarbeitung und teilt dem EC-CAN-00x mit, dass es sich bei dem Script um ein „Pawn Script“ handelt.	
Script Quelle	Script	Script	Eingabefenster zum Editieren des Scripts, das in das EC-CAN-00x geladen wird (siehe "Pawn Script" auf Seite 85)
	Hochladen eines kompilierten Scripts	Datei hochladen	Auswahl des Script Binary-Files (*.amx), das auf den myDatanet-Server hochgeladen und bei der nächsten Verbindung in das EC-CAN-00x geladen wird. Der Dateipfad wird nur solange angezeigt, solange die Eingabemaske zur Konfiguration der Messstelle nicht geschlossen wurde.
Datenstruktur	Eingabefenster zum Konfigurieren des myDatanet Connector (siehe "myDatanet Connector " auf Seite 185)		

11.2.1.4 Konfiguration 0 - Konfiguration 1

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 myDatanet Connector (siehe "myDatanet Connector " auf Seite 185) definiert werden.

11.2.1.5 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.
Bezeichnung Benutzeralarm 1	frei wählbare Bezeichnung für den benutzerdefinierten Alarm 1. Wird von einem mit der Messstelle 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	

11.2.1.6 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	Die Standardgrafik wird geladen.
	"Name einer Auswertung"	Die ausgewählte Auswertung wird geladen.
Auswertungs-Vorlage	Auswahl, ob beim Klicken auf das Symbol zur Anzeige der Messdaten, das sich in der Messstellen-/Applikationsliste befindet, die standardmäßige Grafik oder eine Auswertungs-Vorlage zur Darstellung der Daten verwendet wird. In der Dropdown-Liste werden nur jene Auswertungs-Vorlagen angezeigt, bei denen der Messstellen-/Applikationstyp der ersten Wildcard kompatibel zur Messstelle/Applikation ist, die aktuell bearbeitet wird.	
	(nicht zugeordnet)	Die standardmäßige Grafik wird für die Anzeige der Messdaten verwendet.
	"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.

11.2.1.7 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	

11.2.2 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 Messstellenliste (siehe "Benutzerhandbuch für myDatanet-Server " 206.886) oder durch Klicken auf den Gerätenamen in der Messgeräteliste (siehe "Benutzerhandbuch für myDatanet-Server " 206.886).

11.2.2.1 Kommentar

Kommentar
freies Kommentarfeld (wird auch unterhalb des Messstellennamens in der Messgeräteliste angezeigt)

11.2.2.2 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 Messstelle verbunden werden kann, müssen die Geräteklasse der Messstelle 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	
Modem Version	aktuell installierte Softwareversion des Modemcontrollers	
OS Version	OS Version des Modems	
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.	
Script Sync	Produktiv	Stimmen das im Gerät installierte und das am Server gespeicherte Applikationsscript nicht überein, wird das am Server gespeicherte Applikationsscript in das Gerät geladen.
	Entwicklung (sync)	Es erfolgt eine Synchronisation des Applikationsscripts 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 des Applikationsscripts 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)	
Hardware Version	Hardwareversion des EC-CAN-00x	

11.2.2.3 GPRS

SIM Tarif
ausgewählter SIM-Tarif

Kapitel 12 rapidM2M Toolset

Hinweis: *Alle Screenshots zeigen das rapidM2M Toolset in der Version 1.12. Bei neueren Versionen können geringfügige Änderungen am Erscheinungsbild des Programms vorgenommen worden sein.*

12.1 Allgemein

Das rapidM2M Toolset steht unter folgender Adresse gratis zum Download bereit:

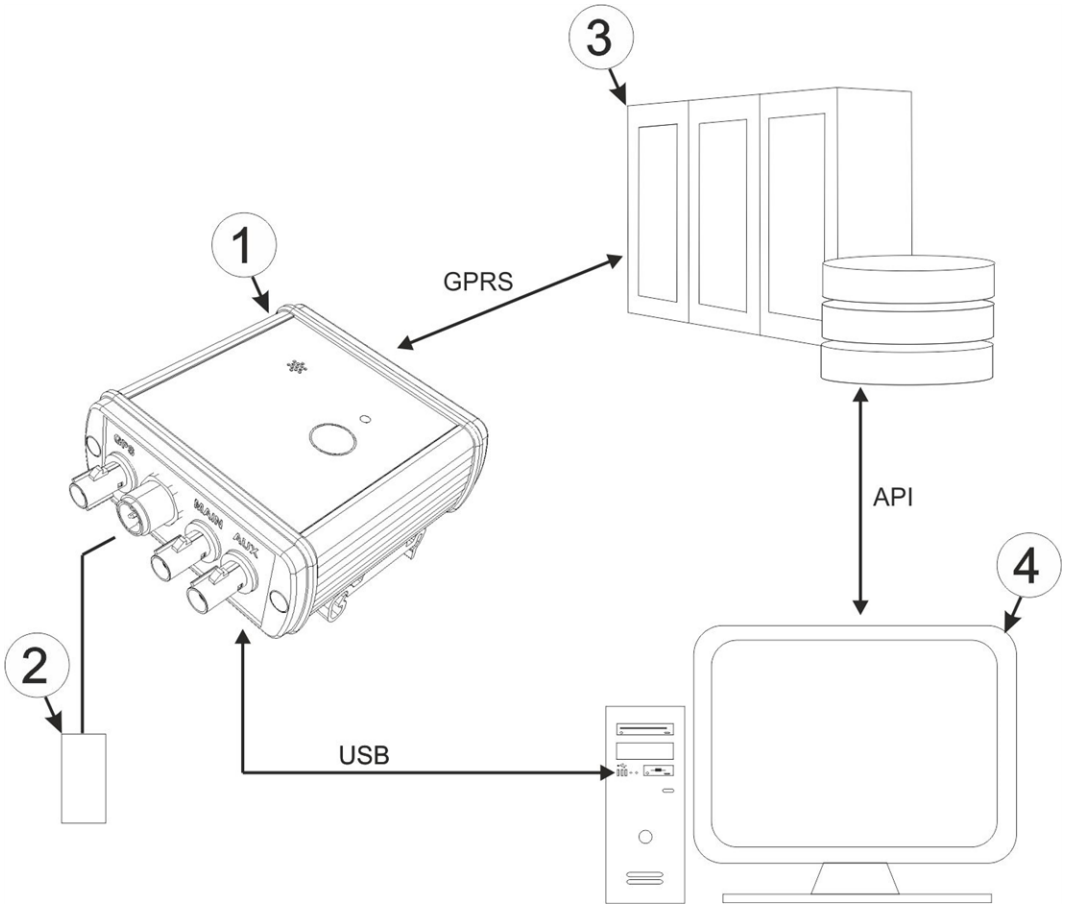
www.microtronics.com/Toolset

Es handelt sich um eine IDE, die den Kunden bei der Erstellung von Applikationen für das EC-CAN-00x unterstützen soll. Diese umfasst den kompletten Entwicklungsprozess - angefangen vom Editieren des PAWN-Scripts (siehe "Pawn Script" auf Seite 85), über das Kompilieren des Scripts in Byte-Code und Laden des Byte-Codes in das EC-CAN-00x bis hin zum Auslesen der vom EC-CAN-00x an den myDatanet-Server übertragenen Daten über die API des myDatanet-Servers. Das rapidM2M Toolset ermöglicht auch die Anzeige der Debug-Informationen, die innerhalb des PAWN-Scripts mittels der Funktionen "print" und "printf" auf die Standardausgabe gedruckt wurden und das Laden des Log-Files aus dem EC-CAN-00x . Die 10 Konfigurationsspeicherblöcke können ebenfalls mittels rapidM2M Toolset aus dem EC-CAN-00x gelesen bzw. in das EC-CAN-00x geschrieben werden. Das rapidM2M Toolset kann auch dazu verwendet werden, Firmwareupdates in das EC-CAN-00x einzuspielen und enthält eine Sammlung von Beispiel-Scripts.

12.2 Voraussetzungen

Schnittstellen	1 x USB
Betriebssystem	Win XP Windows Vista Windows 7
Internetverbindung	empfohlen
Benötigter Speicherplatz	ca. 75MB

12.3 Funktionsprinzip



Funktionsprinzip

1	EC-CAN-00x	**3**	myDatanet-Sever
2	Sensor, der mit dem EC-CAN-00x verbunden ist (optional)	**4**	PC, auf dem das rapidM2M Toolset installiert ist

> ***Wichtiger Hinweis:*** *Bei der USB-Schnittstelle handelt es sich um eine Serviceschnittstelle, die nur nach Öffnen des Gehäuses zugänglich ist. Geräteschäden wie z.B. durch Feuchtigkeitseintritt, die auf das Öffnen des Gehäuses zurückzuführen sind, entfallen aus der Haftung des Herstellers.*

Das rapidM2M Toolset kommuniziert über die USB-Schnittstelle direkt mit dem EC-CAN-00x . Dazu gehören:

- der Script-Download (siehe "Kateireiter "Script"" auf Seite 70)
- der Up/Download der Konfigurationsspeicherblöcke (siehe "Karteireiter "Config"" auf Seite 72)
- die Anzeige der Konsolenausgaben (siehe "Karteireiter "Console"" auf Seite 73)
- das Auslesen der Log-Einträge (siehe "Karteireiter "Log"" auf Seite 75)
- das Firmwareupdate (siehe "Karteireiter "FW"" auf Seite 76)

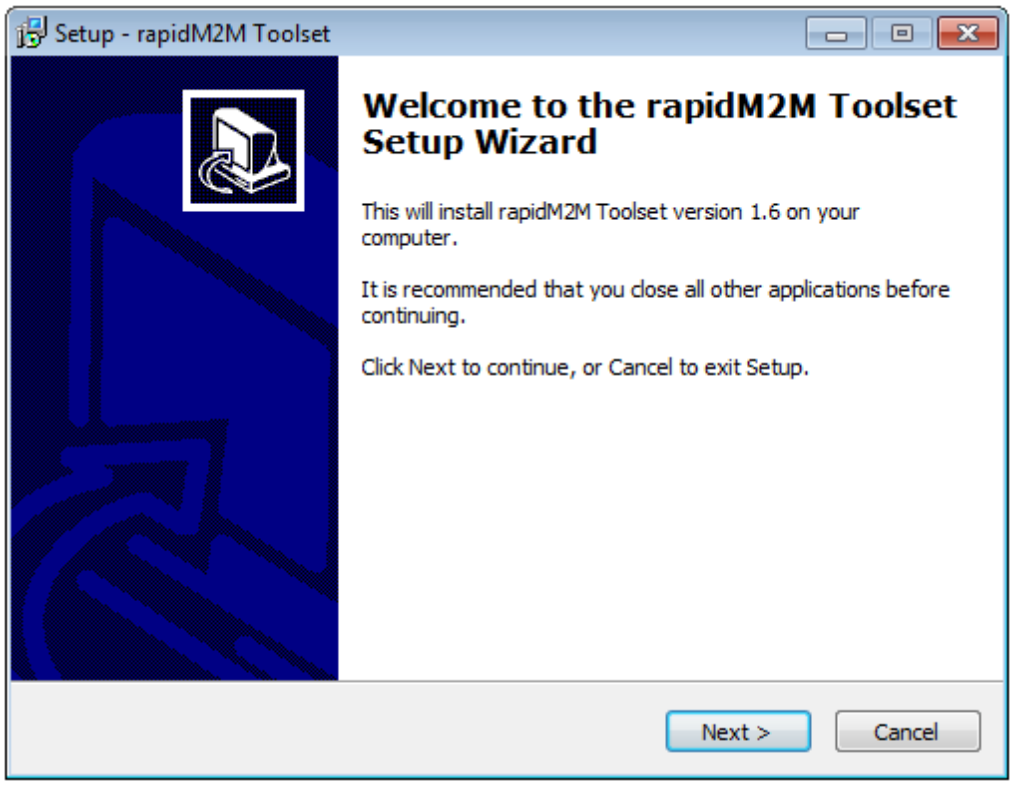
Das rapidM2M Toolset bietet auch die Möglichkeit die Daten über die API des myDatanet-Severs auszulesen (siehe "Karteireiter "Server"" auf Seite 74). Dadurch kann sehr einfach und schnell überprüft werden, ob die komplette Prozesskette - angefangen beim Speichern der Daten mittels Script über die Übertragung per GPRS bis hin zur Ablage der Daten am Server - korrekt funktioniert.

12.4 Installation

Das folgende Kapitel beschreibt den Installationsprozess unter Windows 7.

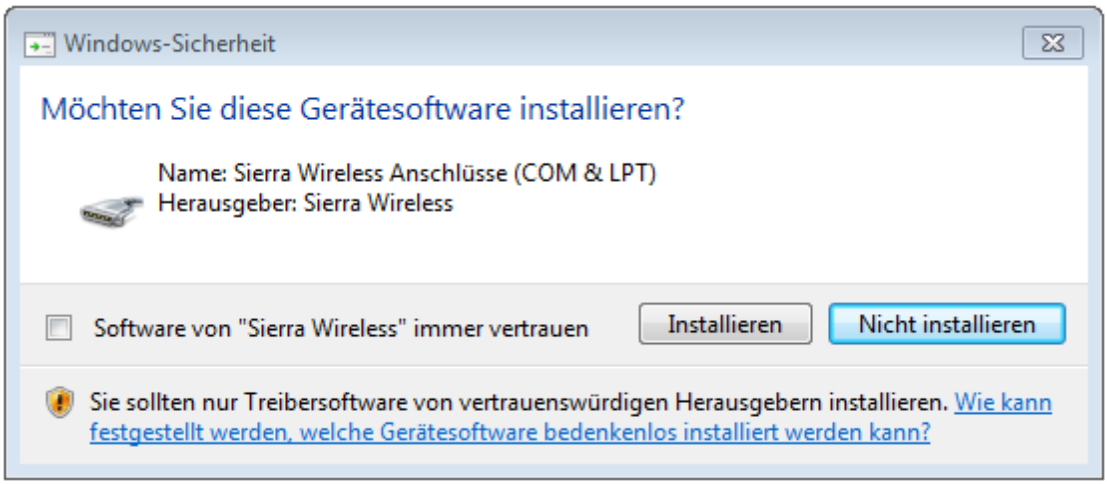
1. Führen Sie die Datei *"InstRapidM2MToolset.exe"* aus, um den Installationsprozess zu starten.

   Hinweis: *Verbinden Sie das EC-CAN-00x erst nach Abschluss des Installationsprozesses mit Ihrem PC, da die benötigten USB-Treiber erst während dieses Vorgangs installiert werden.*

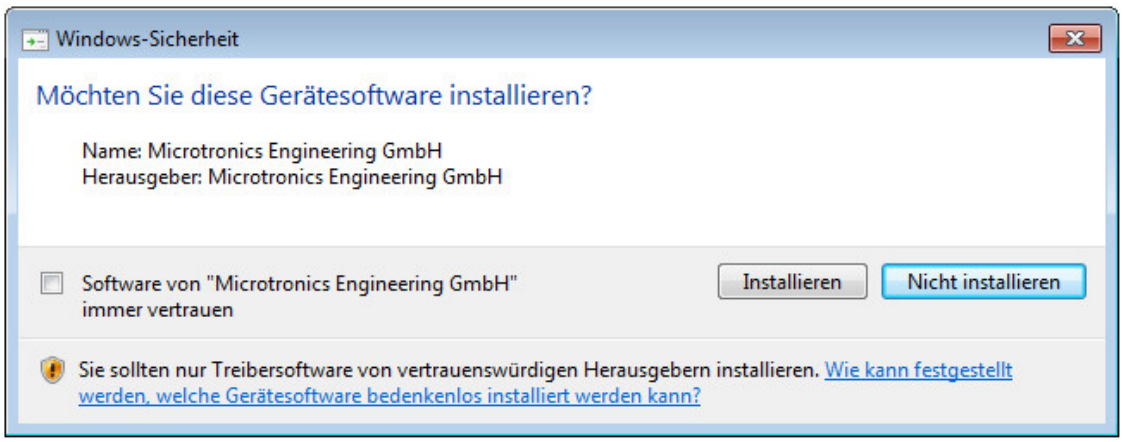


rapidM2M Toolset Setup Wizard

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

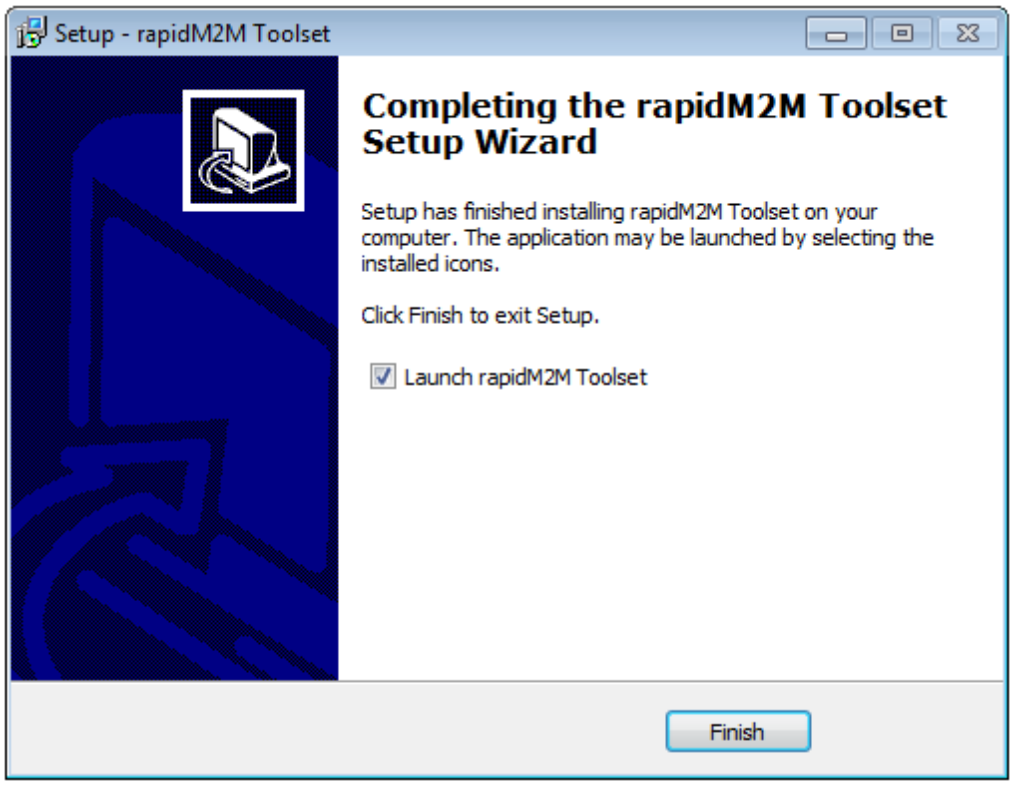


Installation der USB-Treiber für M12x Module



Installation der USB-Treiber für M22x Module und EC-CAN-00x

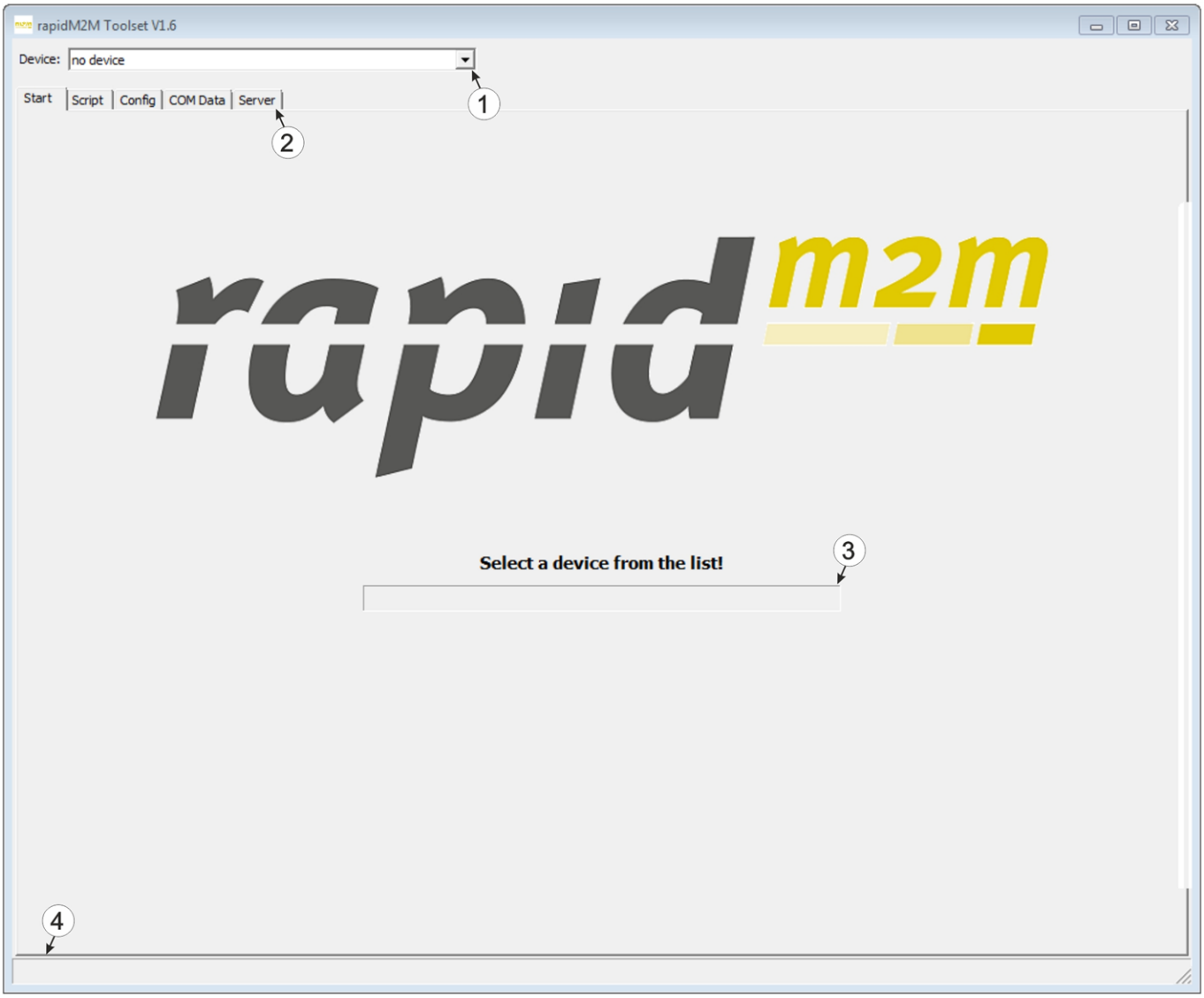
3. Wenn Sie schließlich zur folgenden Ansicht gelangen, schließen Sie den Installationsvorgang durch Klicken auf den Button *"Finish"* ab.



Setup abschließen

4. Nachdem Sie auf den Button *"Finish"* geklickt haben, startet die Installation der USB Treiber.

12.5 Übersicht



Karteireiter "Start" (kein EC-CAN-00x ausgewählt)

1	Auswahl des EC-CAN-00x	**3**	Fortschrittsbalken für den Aufbau der USB-Verbindung zum EC-CAN-00x
2	Auswahl der Karteireiter für die verschiedenen Funktionen des rapidM2M Toolset , die auch ohne verbundenem EC-CAN-00x verfügbar sind	**4**	Anzeige des Kommunikationsstatus auf der USB-Schnittstelle zum EC-CAN-00x

Sobald die USB-Verbindung zu einem EC-CAN-00x hergestellt wurde, werden die Karteireiter der Funktionen, die eine bestehende USB-Verbindung zum EC-CAN-00x erfordern, eingeblendet und der Karteireiter "Start" ausgeblendet. Zusätzlich werden im rechten oberen Bereich des rapidM2M Toolset weitere Informationen zum EC-CAN-00x eingeblendet.

Wichtiger Hinweis: *Verknüpfen Sie das EC-CAN-00x mit einer Messstelle (siehe "Anlegen der Messstelle" auf Seite 81), bevor Sie per USB darauf zugreifen. Bei einer nachträglichen Verknüpfung würden alle bisher getroffenen Einstellungen (inkl. Script) durch die Übernahme der Einstellungen von der Messstelle verloren gehen. Die Meldung "`Warning: No Context available! All settings and script will be lost when device is assigned to a site`" in der Anzeige für den Kommunikationsstatus auf der USB-Schnittstelle weist darauf hin, dass noch keine Messstelle mit dem EC-CAN-00x verknüpft ist.*



Verbindung zum EC-CAN-00x hergestellt

1	Auswahl des EC-CAN-00x	**4**	Anzeige der GSM-Signalstärke
2	Auswahl der Karteireiter für die verschiedenen Funktionen des rapidM2M Toolset	**5**	Seriennummer des verbundenen EC-CAN-00x
3	Anzeige des GSM-Verbindungsstatus. Durch Klicken auf dieses Symbol kann ein Verbindungsaufbau gestartet werden.		

GSM-Verbindungsstatus

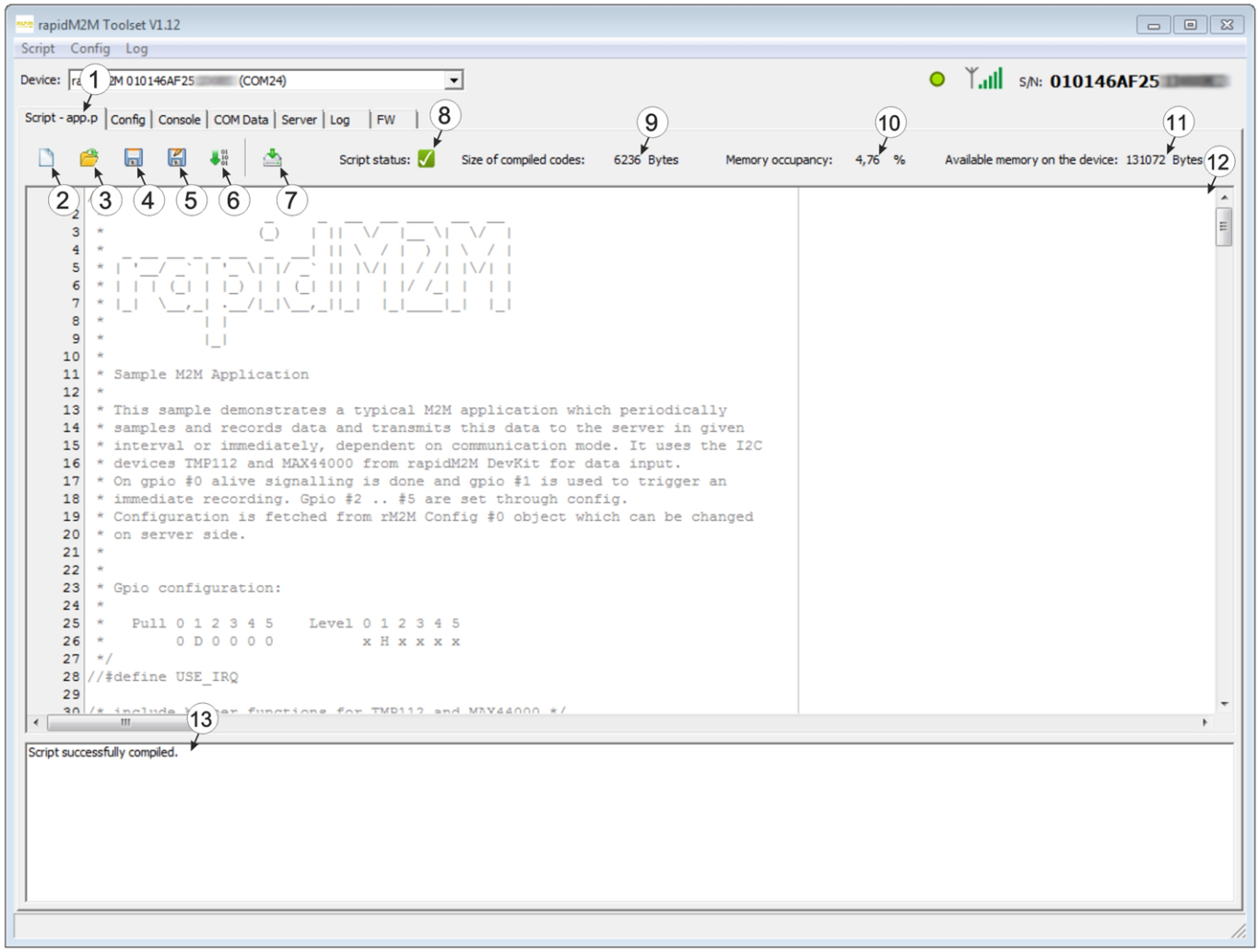
Statussymbol	
	offline
	offline, Wakeup über den myDatanet-Server möglich
, blinkend	Verbindung wird hergestellt
	verbunden
	letzter Verbindungsversuch ist fehlgeschlagen
	letzter Verbindungsversuch ist fehlgeschlagen, Wakeup über den myDatanet-Server möglich

GSM-Signalstärke

	> -64dBm
	-64...-73dBm
	-74...-83dBm
	-84...-93dBm
	-94...-107dBm
	<= -108dBm

12.6 Kateireiter "Script"

Dieser Bereich dient zum Erstellen des Scripts. Der Karteireiter ist auch verfügbar, wenn keine USB-Verbindung zu einem EC-CAN-00x besteht. Mit Ausnahme des Downloads des kompilierten Scripts und der Anzeige der Speichergrößen ("Size of compiled codes", "Memory occupancy" und "Available memory on the device"), stehen in diesem Fall alle Funktionen zur Verfügung. Das zuletzt geöffnete Script wird beim Programmstart automatisch geladen. Sollte das gerade geöffnete Script durch ein anderes Programm verändert worden sein, erfolgt eine Meldung mit der Möglichkeit das geänderte Script in den Script-Editor zu laden.



Karteireiter "Script"

1	Dateiname des geöffneten Scripts. Wurde das Script geändert jedoch noch nicht gespeichert, wird dies durch "*" nach dem Dateinamen signalisiert.
2	erstellt ein neues leeres Script
3	öffnet ein gespeichertes Script
4	speichert das aktuell geöffnete Script
5	öffnet den Dialog zum Speichern des Scripts unter einem anderen Dateinamen
6	kompiliert das geöffnete Script. Dabei wird es auch automatisch gespeichert.

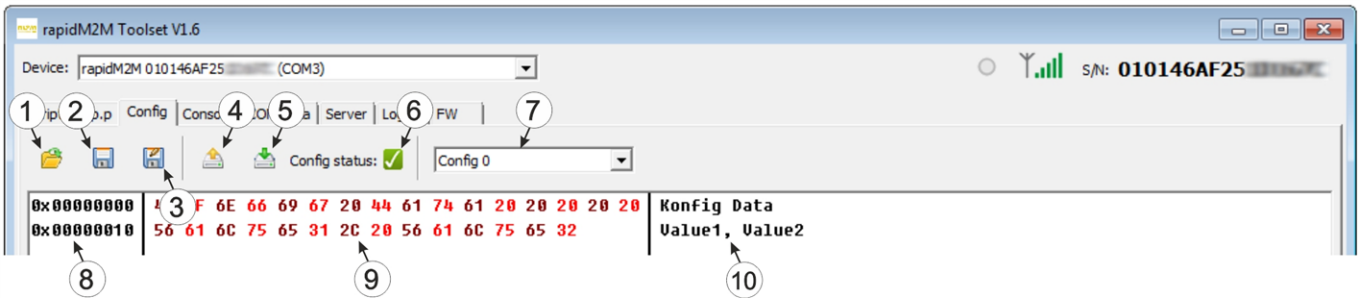
7	überträgt das geöffnete Script in das Modul/Gerät. Dabei wird das Script zuerst gespeichert und dann kompiliert. Das kompilierte PAWN-Binary (*.amx) wird im selben Ordner wie das Script gespeichert.
8	Status der Script-Übertragung (siehe "Script Status")
9	Größe des kompilierten Codes
10	gibt die Belegung des verfügbaren Speichers in % an
11	gibt die Größe des verfügbaren Speichers am EC-CAN-00x an (Angabe der komprimierten Größe)
12	Script-Editor
13	Debugausgabe des Script-Kompilers. Durch einen Doppelklick auf einen Eintrag in der Debugausgabe springt der Curser im Script-Editor zur betroffenen Zeile.

Script Status

Symbol	Beschreibung
	Script wurde erfolgreich in das EC-CAN-00x geladen.
	Script wird gerade zum EC-CAN-00x übertragen.
	Der Zeitstempel des im EC-CAN-00x gespeicherten kompilierten Scripts stimmt nicht mit dem des kompilierten Scripts des rapidM2M Toolset überein.

12.7 Karteireiter "Config"

Dieser Karteireiter ermöglicht den Zugriff auf die 10 Konfigurationsspeicherblöcke. Diese können editiert, gespeichert und vom und zum EC-CAN-00x übertragen werden. Der Karteireiter ist auch verfügbar, wenn keine USB-Verbindung zu einem EC-CAN-00x besteht. In diesem Fall ist es allerdings nicht möglich, die Konfigurationsblöcke vom und zum EC-CAN-00x zu übertragen.



Karteireiter "Config"

1	öffnet einen gespeicherten Konfigurationsspeicherblock	**6**	Status der Konfigurationsblockübertragung (siehe "Config Status")
2	speichert den aktuell geöffneten Konfigurationsspeicherblock	**7**	Auswahl des Konfigurationsspeicherblocks[1)]
3	öffnet den Dialog zum Speichern des aktuell geöffneten Konfigurationsspeicherblocks unter einem anderen Dateinamen	**8**	Adressspalte
4	lädt den ausgewählten Konfigurationsspeicherblock vom EC-CAN-00x	**9**	Hex-Ansicht des Konfigurationsspeicherblocks
5	überträgt den ausgewählten Konfigurationsspeicherblock ans EC-CAN-00x	**10**	ASCII-Ansicht des Konfigurationsspeicherblocks

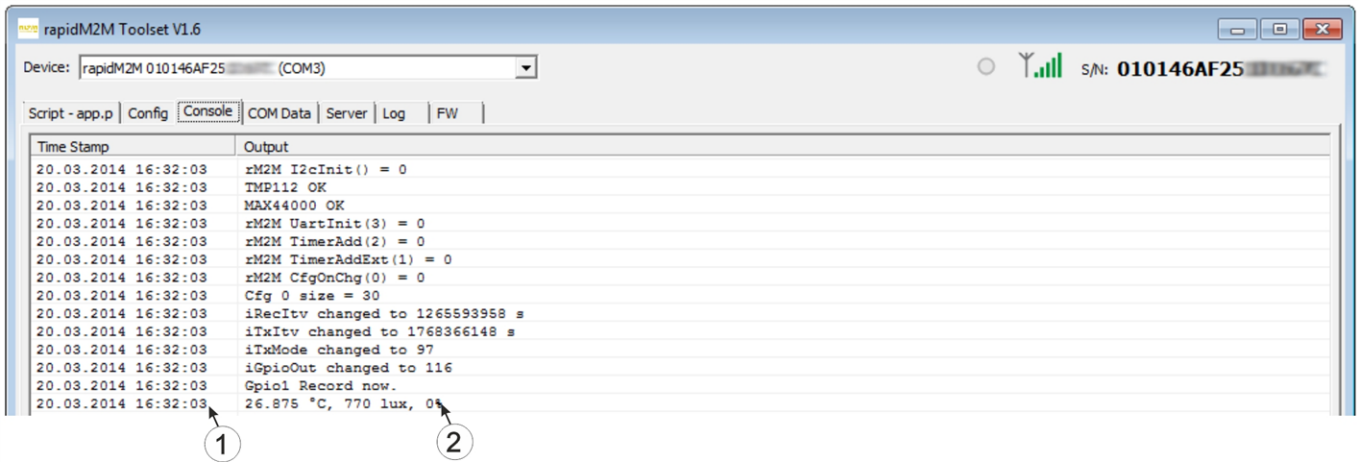
[1)] Die Operationen "öffnen", "speichern" und "übertragen" beziehen sich immer nur auf den ausgewählten Konfigurationsspeicherblock.

Config Status

Symbol	Beschreibung
	Der ausgewählte Konfigurationsspeicherblock wurde erfolgreich in das EC-CAN-00x geladen.
	Der ausgewählte Konfigurationsspeicherblock wird gerade übertragen.
	Der Zeitstempel des im EC-CAN-00x gespeicherten Konfigurationsspeicherblock stimmt nicht mit dem des rapidM2M Toolset überein. Die Anzeige gilt für den aktuell ausgewählten Konfigurationsspeicherblock.

12.8 Karteireiter "Console"

Dieser Bereich dient der Anzeige von Debug-Informationen, die innerhalb des PAWN-Scripts mittels der Funktionen "print" und "printf" (siehe "Console" auf Seite 153) auf die Standardausgabe gedruckt wurden. Der Karteireiter "Console" ist nur verfügbar, wenn eine USB-Verbindung zu einem EC-CAN-00x besteht.



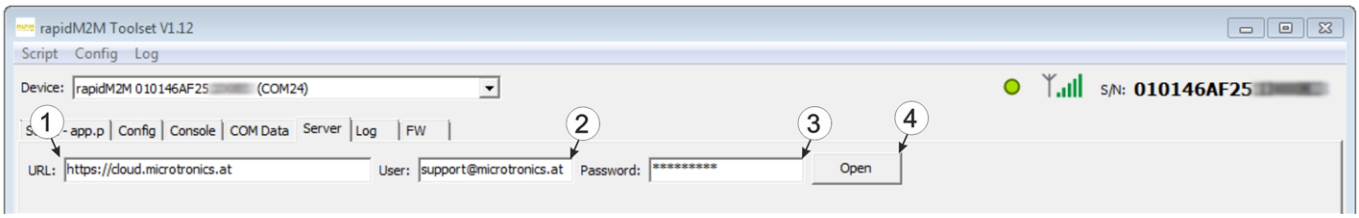
Karteireiter "Console"

1 Zeitpunkt, zu dem die Debug-Informationen über die USB-Schnittstelle empfangen wurden	**2** Anzeige der Debug-Informationen

Durch einen Klick mit der rechten Maustaste auf einen der Einträge der Debug-Anzeige wird ein Kontextmenü eingeblendet, dass das Löschen sowie das Speichern der Liste ermöglicht.

12.9 Karteireiter "Server"

Über diesen Karteireiter erreichen Sie die Testseite der REST-API des myDatanet-Servers. Mit ihrer Hilfe können Sie sehr einfach und schnell überprüften, ob die komplette Prozesskette - angefangen beim Speichern der Daten mittels Script, über die Übertragung per GPRS, bis hin zur Ablage der Daten am Server - korrekt funktioniert. Diese Funktion kann auch ohne über die USB-Schnittstelle verbundenes EC-CAN-00x verwendet werden. In diesem Fall muss zusätzlich zu Serveradresse, Benutzername und Passwort auch noch die Seriennummer des Geräts eingegeben werden (siehe "Karteireiter "Server" ohne verbundenem EC-CAN-00x " auf Seite 74).



Karteireiter "Server" mit verbundenem EC-CAN-00x

1	Adresse des Servers (myDatanet-Server)	**3**	Passwort für das Benutzerkonto
2	Benutzername eines am myDatanet-Server angelegten Benutzerkontos	**4**	öffnet die Testseite der REST-API des myDatanet-Servers im Browser

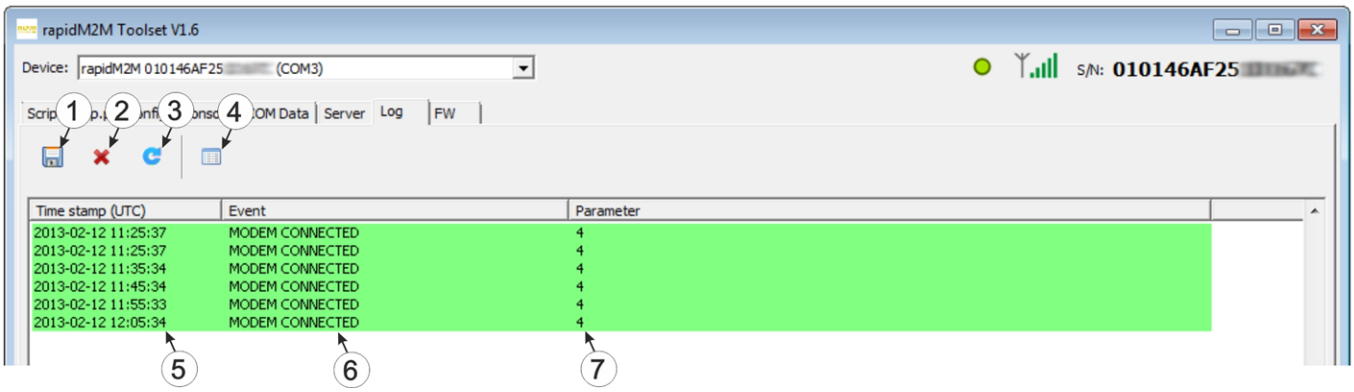


Karteireiter "Server" ohne verbundenem EC-CAN-00x

1	Seriennummer des EC-CAN-00x auf dessen Daten per REST-API zugegriffen werden sollen

12.10 Karteireiter "Log"

Dieser Bereich dient der Verwaltung der Log-Einträge. Er ermöglicht das Laden der Einträge vom EC-CAN-00x , das Speichern als *.tsv-Datei und das Löschen der Einträge aus dem Speicher des EC-CAN-00x .

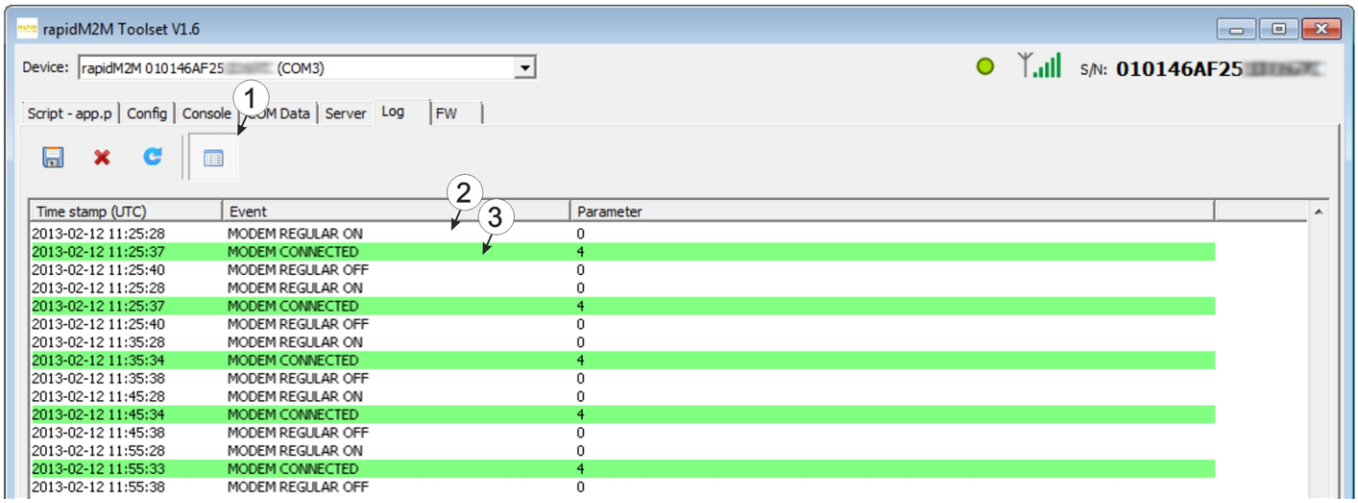


Karteireiter "Log"

1	speichert die angezeigten Log-Einträge als *.tsv-Datei	**5**	Zeitstempel des Log-Eintrags
2	löscht die Log-Einträge aus dem Speicher des EC-CAN-00x	**6**	Log-Eintrag
3	lädt die Log-Einträge vom EC-CAN-00x	**7**	Parameter des Log-Eintrags
4	aktiviert die detaillierte Darstellung der Log-Einträge		

Die farbliche Markierung gibt Aufschluss darüber, wie kritisch der Log-Eintrag zu bewerten ist. Die weiß gekennzeichneten informativen Log-Einträge werden nur angezeigt, wenn die detaillierte Darstellung der Log-Einträge aktiviert ist (siehe "Karteireiter "Log" mit aktivierter Detailansicht" auf Seite 76).

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

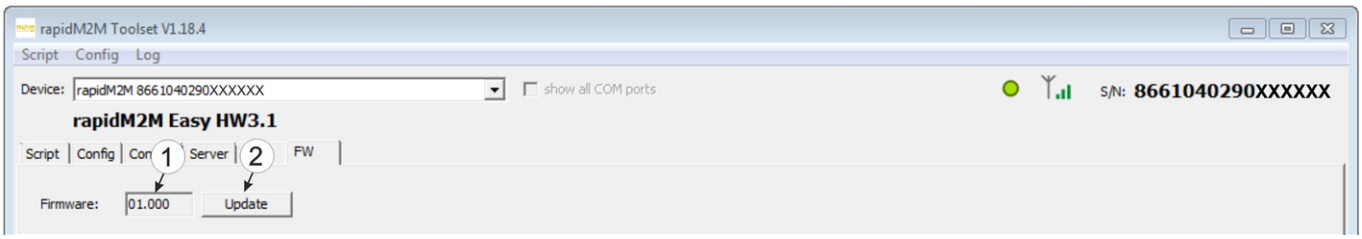


Karteireiter "Log" mit aktivierter Detailansicht

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

12.11 Karteireiter "FW"

Dieser Bereich ermöglicht das direkte Einspielen der Firmware über die USB-Schnittstelle.



Karteireiter "FW"

1	aktuell installierte Firmware	**2**	öffnet den Dialog zur Auswahl der Firmware-Datei

12.12 Hot-Keys

Befehl	Tastenkombination	Erklärung
New Script	Ctrl+N	erstellt ein neues leeres Script
Open Script	Ctrl+O	öffnet ein gespeichertes Script
Save Script	Ctrl+S	speichert das aktuell geöffnete Script
Compile Script	Shift+F9	kompiliert das geöffnete Script. Dabei wird es auch automatisch gespeichert.
Compile and Send to Device	F9	überträgt das geöffnete Script in das Modul/Gerät. Dabei wird das Script zuerst gespeichert und dann kompiliert. Das kompilierte PAWN-Binary (*.amx) wird im selben Ordner wie das Script gespeichert.

Kapitel 13 myDatanet-Server

Hinweis: *Alle Screenshots zeigen den myDatanet-Server in der Version 44.1 unter Verwendung des Standard-Farbschemas. Bei neueren Versionen können geringfügige Änderungen am Erscheinungsbild des Servers vorgenommen worden sein.*

13.1 Übersicht



Übersicht myDatanet-Server

1	frei wählbares Logo	**5**	öffnet die Maske zur Eingabe der globalen Einstellungen für den Server
2	Benutzergruppe, der der aktuell angemeldete Benutzer zugeordnet ist	**6**	wechselt in den Bereich "Datenexports" zur Konfiguration des Datenexports. Diese Schaltfläche ist nur sichtbar, wenn zumindest die Lizenz für eine Exportvariante vorhanden ist.
3	aktiven Benutzer ausloggen	**7**	öffnet die Eingabemaske zum Upload einer XML-Datei. Diese Schaltfläche ist nur sichtbar, wenn die Lizenz für den XML-Import vorhanden ist.
4	Schaltflächen zum Wechseln zwischen den einzelnen Serverbereichen	**8**	Aufruf des Hilfe-Menüs

13.1.1 Erklärung der Symbole



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

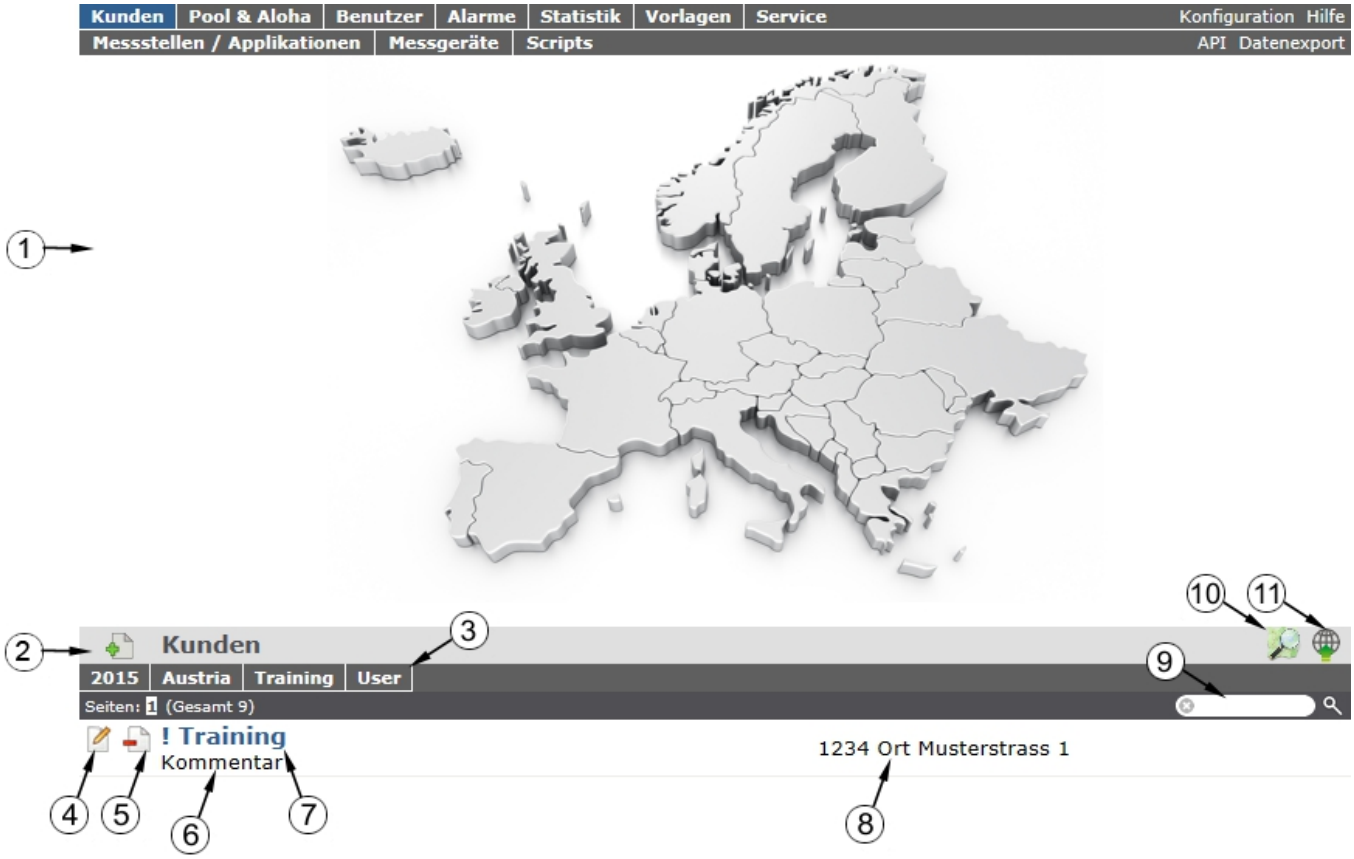


Löscht das nebenstehende Element (Auswertung, Messstelle, Benutzer, ...) aus der Liste



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

13.2 Bereich "Kunden"

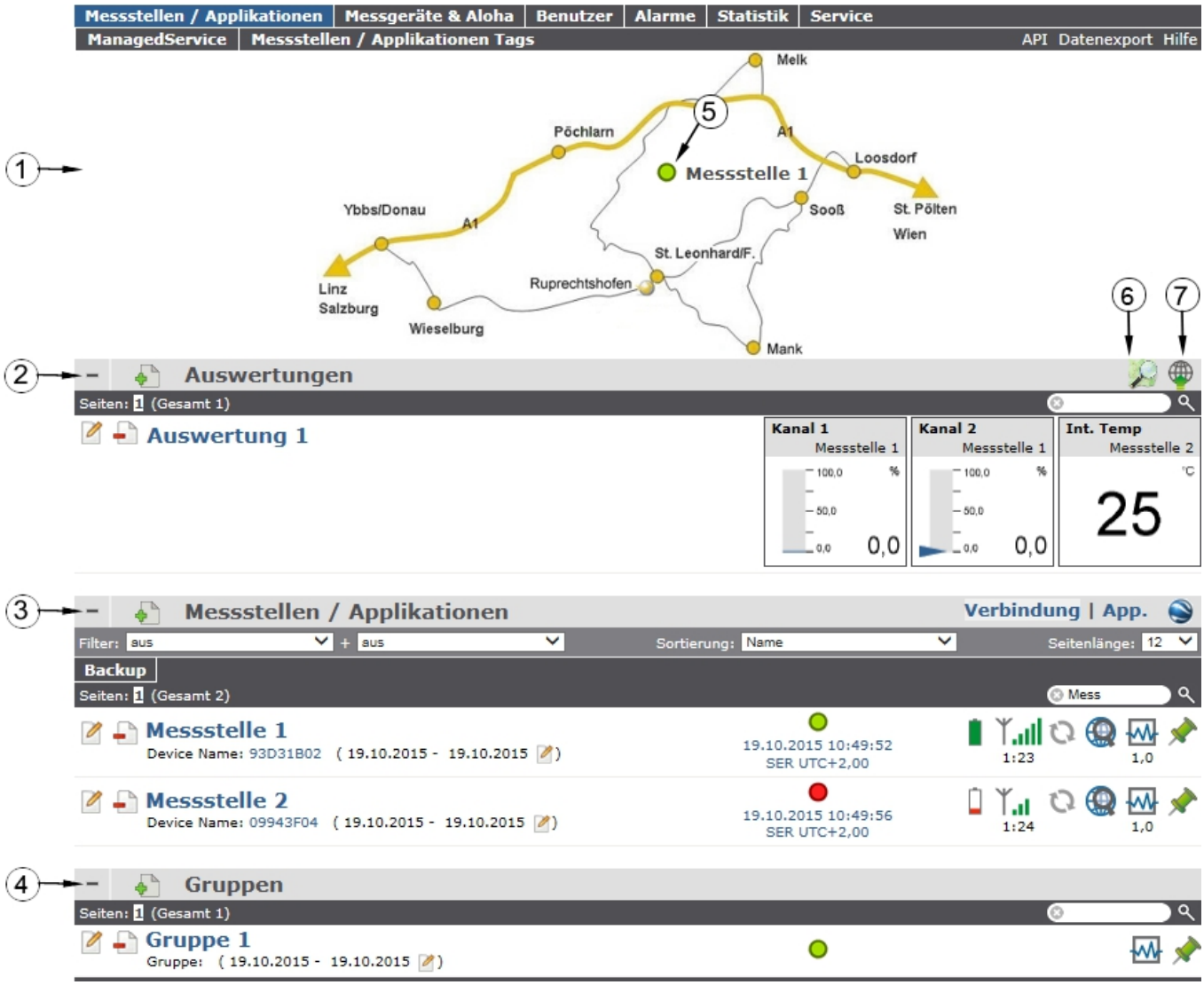


Übersicht des Bereichs "Kunden"

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 Messstellen manuell platzieren. In der OpenStreetMaps Karte werden die Messstellen erst angezeigt, wenn der Messstelle 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 "Messstellen" auf Kundenebene geöffnet (siehe "Bereich "Messstellen" auf Kundenebene" auf Seite 80 bzw. "Auswertungen" auf Seite 81).
8	Adresse des Kunden, die über die Eingabemaske zur Konfiguration des Kunden eingegeben werden kann
9	Suchfeld zum Filtern der Kundenliste
10	Symbol, über das sich eine OpenStreetMaps Karte laden lässt, auf der die Messstellen dargestellt werden. (siehe "Kartendarstellung" auf Seite 81)
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.

13.3 Bereich "Messstellen" auf Kundenebene



Übersicht des Bereichs "Messstellen" 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 Messstellen manuell platzieren. In der OpenStreetMaps Karte werden die Messstellen erst angezeigt, wenn der Messstelle GPS-Koordinaten zugewiesen wurden.
2	Liste der Auswertungen (siehe "Auswertungen" auf Seite 81)
3	Liste der Sites / Applikationen (siehe "Site" auf Seite 56)
4	Liste der Gruppen (siehe "Gruppen" auf Seite 81)
5	Symbol, das eine Messstelle auf der "Karte" repräsentiert

6	Symbol, über das sich eine OpenStreetMaps Karte laden lässt, auf der die Messstellen dargestellt werden. (siehe "Kartendarstellung" auf Seite 81)
7	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.

13.3.1 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).

13.3.2 Gruppen

Gruppen bieten eine Vielzahl an Möglichkeiten zur Verdichtung und Zusammenfassung der Messdaten unterschiedlicher Messstellen. Eine detaillierte Anleitung zum Erstellen und dem Umgang mit den Gruppen finden Sie im Benutzerhandbuch für myDatanet-Server (206.886).

13.3.3 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).

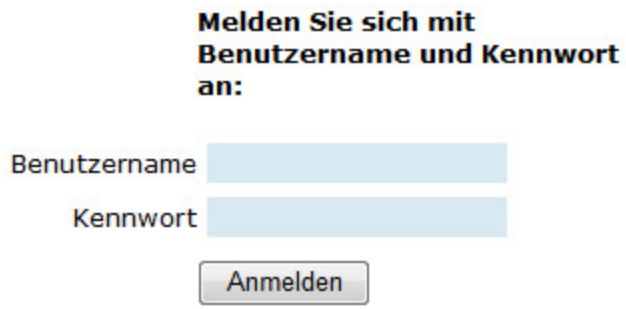
13.4 Empfohlene Vorgehensweise

13.4.1 Anlegen der Messstelle

> ***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 Messstelle 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

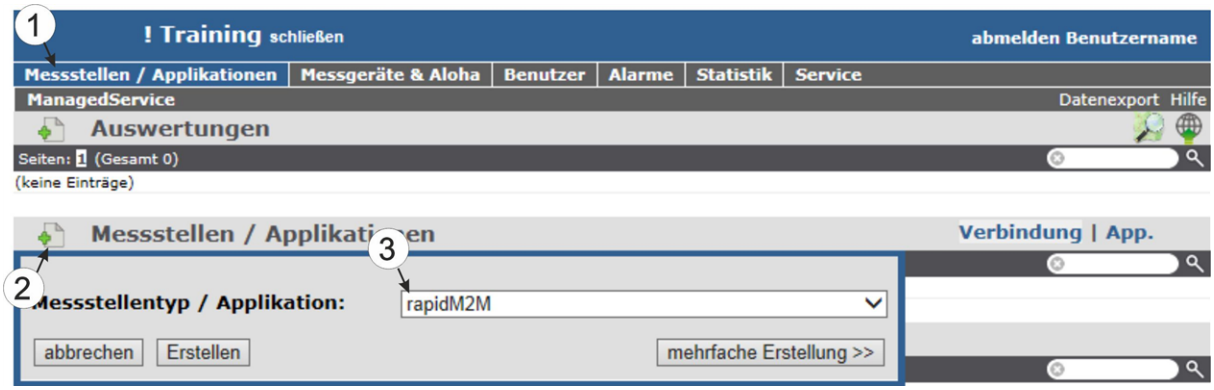
2. Klicken Sie auf den Menüpunkt "Kunde" des myDatanet–Servers um die Liste der verfügbaren Kunden aufzurufen. Wählen Sie einen bestehenden Kunden aus oder legen Sie einen neuen Kunden an.



Auswählen des Kunden

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

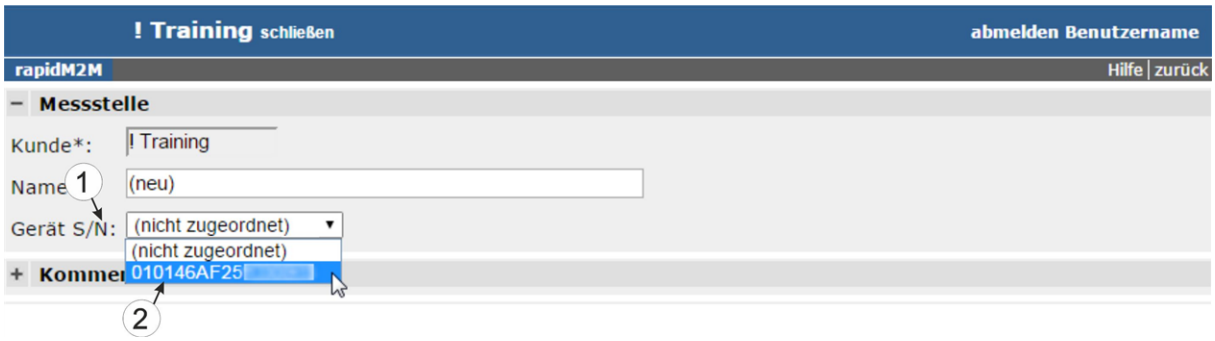
3. Klicken Sie auf den Menüpunkt "Messstellen / Applikationen" des myDatanet–Servers, um die Liste der verfügbaren Messstellen bzw. Applikations-Vorlagen aufzurufen. Legen Sie eine neue Messtelle vom Typ "rapidM2M" oder eine Messstelle aus einer zum Messstellentyp "rapidM2M" kompatiblen Applikations-Vorlage an.



Anlegen der Messstelle

1 Menüpunkt zum Aufrufen der Messstellenliste	**3** Auswahl des Messstellentyps
2 Anlegen einer neuen Messstelle	

4. Verknüpfen Sie die Messstelle / Applikation mit dem EC-CAN-00x über die Auswahl der Seriennummer. Wenn die Seriennummer Ihres Geräts nicht in der Liste enthalten ist, müssen Sie das Gerät zuerst dem Kunden zuweisen (siehe "Gerät dem Kunden zuweisen" auf Seite 83) oder das Gerät ist bereits einer anderen Messstelle / Applikation zugeordnet. Detaillierte Informationen dazu finden Sie im Benutzerhandbuch für myDatanet-Server (206.886).



Verknüpfung von Gerät und Messstelle

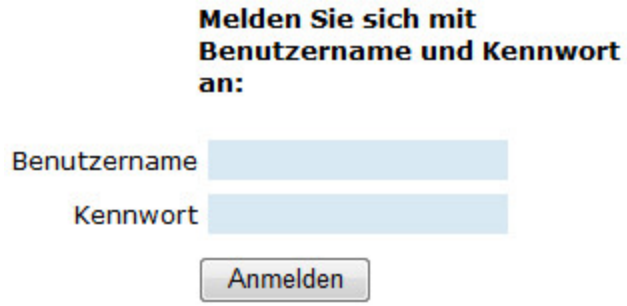
1 Gerätezuordnung	**2** Liste der noch nicht mit einer Messstelle verknüpften Geräte

13.4.2 Gerät dem Kunden zuweisen

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.*

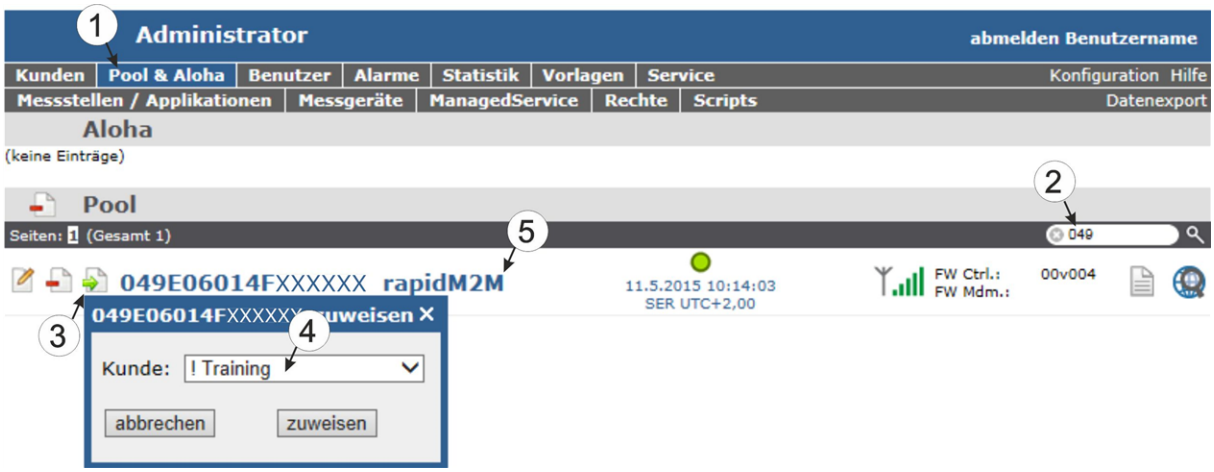
Eine detaillierte Anleitung zum Verschieben der Geräte finden Sie im Benutzerhandbuch für myDatanet-Server (206.886).

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 "Pool und Aloha" des myDatanet–Servers, um zur Liste der Geräte zu gelangen, die keinem Kunden zugewiesen sind und geben Sie die Seriennummer des EC-CAN-00x in das Suchfeld ein. Wenn Ihr Gerät nicht in der Liste enthalten ist, müssen Sie gegebenenfalls eine Übertragung auslösen. Falls noch kein Script 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 ein Script ins Gerät geladen, führen Sie bitte die im Script vorgesehenen Operationen zum Auslösen des Verbindungsaufbaus aus. Das Gerät wird nicht in der Liste angezeigt, wenn es bereits einem Kunden zugeordnet ist. Detaillierte Informationen dazu finden Sie im Benutzerhandbuch für myDatanet-Server (206.886).



Gerät dem Kunden zuweisen

1	Menüpunkt zum Aufrufen der Liste der Geräte, die keinem Kunden zugewiesen sind	**4**	Auswahl des Kunden, dem das Gerät zugewiesen werden soll
2	Filter für die Messgeräteliste	**5**	Messgeräteliste
3	Öffnet den Dialog zur Auswahl des Kunden, dem das Gerät zugewiesen werden soll		

3. Benutzen Sie den Filter, um das gewünschte Gerät in der Messgeräteliste zu finden. Klicken Sie auf das Symbol zum Öffnen des Dialogs zur Auswahl des Kunden (siehe "Gerät dem Kunden zuweisen" auf Seite 84) und weisen Sie das Gerät dem gewünschten Kunden zu.

Kapitel 14 Pawn Script

14.1 Allgemein

Das folgende Kapitel beschreibt die Funktionalität des Pawn Scripts. PAWN (vormals SMALL) ist eine C-ähnliche Skriptsprache, 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 Pawn Script für das EC-CAN-00x zu erstellen:

- Direkte Eingabe in das Eingabefenster „Script“ im Konfigurationsabschnitt „Steuerung“
- Hochladen eines zuvor erstellten Binary-Files (*.amx) auf den myDatanet–Server
- Benutzen des Script-Editors des rapidM2M Toolset

14.1.1 Direkte Eingabe eines Pawn Scripts

Die Eingabe des Pawn Scripts erfolgt über den Konfigurationsabschnitt „Steuerung“ (siehe "Steuerung" auf Seite 57) der Eingabemaske zur Konfiguration der Messstelle. Als „Script Type“ muss „Pawn“ ausgewählt werden, damit das EC-CAN-00x die unter „Script“ eingegebenen Befehle als Pawn Script interpretiert.

14.1.2 Hochladen eines Binary-Files

Wurde über die Listenauswahl "Script Quelle" im Konfigurationsabschnitt „Steuerung“ (siehe "Steuerung" auf Seite 57) der Eingabemaske zur Konfiguration der Messstelle der Eintrag "Hochladen eines kompilierten Scripts" ausgewählt, kann ein zuvor mittels z.B. rapidM2M Toolset (siehe "Kateireiter "Script"" auf Seite 70) erstelltes Binary-File auf den myDatanet–Server hochgeladen werden. Dieses wird dann bei der nächsten Verbindung in das EC-CAN-00x geladen. Als „Script Type“ muss auch bei dieser Methode „Pawn“ ausgewählt werden, damit das EC-CAN-00x die Befehle als Pawn Script interpretiert.

14.1.3 Script-Editor des rapidM2M Toolset

Das rapidM2M Toolset beinhaltet unter dem Kateireiter "Script" (siehe "Kateireiter "Script"" auf Seite 70) einen Editor zum Erstellen und Kompilieren der Pawn Scripts. Zum Funktionsumfang des rapidM2M Toolset gehört auch das Übertragen der kompilierten Scripts per USB-Verbindung in das EC-CAN-00x .

14.2 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>
```

14.3 rapidM2M API

14.3.1 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_FEATURE_LOCKED      = -103,// feature locked
ERROR_TXITF               = -104,/* tx interface (uplink) not available
                                    (e.g. not opened, currenly closing) */
```

14.3.2 Timer, Datum & Zeit

14.3.2.1 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
// day          Tag
// hour         Stunden
// minute       Minuten
// second       Sekunden
// 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                          ]
```

14.3.2.2 Konstanten

Zeitbasis-Flags

Steuerflags für die Funktion rM2M_SetDateTime()

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

14.3.2.3 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 87)*

	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 87).* *.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 kein gültiger Index übergeben wurde, bei einem internen Fehler oder wenn keine weiteren Timer mehr angelegt werden können (maximale Anzahl erreicht)* • *< OK, wenn ein anderer Fehler auftritt (siehe "Returncodes für allgemeine Zwecke" im Kapitel "Konstanten" auf Seite 86).*

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 86).*

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

erzeugt einen neuen ms Timer

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 86).*

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 86).*

14.3.3 Uplink

14.3.3.1 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
//            Verbindung verwendet wurde
//            0, wenn noch keine Verbindung stattgefunden hat
// mnc        MNC (Mobile Network Code) des Netzes, das für die letzte
//            Verbindung verwendet wurde
//            0, wenn noch keine Verbindung stattgefunden hat
// simstate   Aktueller SIM-Status (siehe "SIM-Status" im Kapitel
//            "Konstanten" auf Seite 92)

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

14.3.3.2 Konstanten

SIM-Status

```
//Verbindung kann per Script 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 Script nicht möglich
RM2M_SIM_STATE_COLD       = 3,   //Vertragsende oder Fair-Use Verletzung
RM2M_SIM_STATE_DISCARDED  = 4,   //Gerät wurde außer Dienst gestellt
```

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) */

// 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) */

// Netzwerkbezogen 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
```

Konfigurationsflags für die Funktion rM2M_CfgInit()

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

14.3.3.3 Callback Funktionen

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

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

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)*

public func(cfg);

vom Script-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*

14.3.3.4 Funktionen

native rM2M_TxStart(flags=0);

löst eine Verbindung zum Server aus

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*
	Bit4 (RM2M_TX_SUPPRESS_ POSUPDATE):	*wenn gesetzt, wird das alle 24h von der Firmware automatisch ausgeführte Update der GSM-Positionsdaten unterdrückt*

	Erklärung
Rückgabewert	• *OK, wenn erfolgreich* • *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 86).*

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_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 86).*

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 92.*

	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_GSMGetRSSI();

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.*

	Erklärung
Rückgabewert	*Signalstärke in [dBm] (maximaler Wertebereich: -128...127)* *GSM-Werte sind im Bereich -113 .. -51 dBm.*

native rM2M_GetRSSI();

liefert die Signalstärke am für den Uplink verwendeten Kommunikationsinterface

	Erklärung
Rückgabewert	*Signalstärke in [dBm] (maximaler Wertebereich: -128...127)* *GSM-Werte sind im Bereich -113 .. -51 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 92)*
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 86).*

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. Dies inkludiert auch den für die serverseitige Zuordnung der Rohmessdaten ("rm2mraw") zu den strukturierten Messdatenkanälen ("histdata0" - "histdata9") erforderlichen Schlüssel ("key") am Beginn des Datensatzes (siehe "Split-tag" auf Seite 186).*
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 86).*

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);*

	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 86)*

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. Bitte 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	• *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 86).*

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 86)*

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	• *Größe des verwendeten Speichers im Konfigurationsspeicherblock* • *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 86).*

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 86).*

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 86).*

14.3.4 Encoding

14.3.4.1 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
```

14.3.4.2 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 Unsigned-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 86)*

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 86)*

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 Unsigned-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 86)*

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 86)*

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: Wert, der aus dem Array gelesen werden soll*
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 105). 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 86)*

14.3.5 GPIO, IRQ

14.3.5.1 Konstanten

Einstellung der Signalrichtung der GPIOs

```
RM2M_GPIO_DISABLED = -1,          // GPIO deaktiviert
RM2M_GPIO_INPUT    = 0,           // Eingang
RM2M_GPIO_OUTPUT   = 1,           // Ausgang
```

Singnalpegel der GPIOs

```
RM2M_GPIO_LOW      = 0,           // Signalpegel "low"
RM2M_GPIO_HIGH     = 1,           // Signalpegel "high"
```

Auswahl der Flanke, die den Interrupt auslösen soll

Konfigurationsflags für die Funktion rM2M_IrqInit()

```
RM2M_IRQ_RISING    = 0,           // bei steigender Flanke
RM2M_IRQ_FALLING   = 1,           // bei fallender Flanke
```

14.3.5.2 Callback Funktionen

public func(pinstate);

vom Script-Entwickler bereitzustellende Funktion, die im Falle eines Interrupts an einem der interruptfähigen Pins aufgerufen wird.

Parameter	*Erklärung*
pinstate	*Signallevel am interruptfähigen Pin* • *RM2M_GPIO_LOW für "low" am interruptfähingen Pin* • *RM2M_GPIO_HIGH für "high" am interruptfähingen Pin*

14.3.5.3 Funktionen

native rM2M_GpioDir(gpio, dir);

setzt die Signalrichtung für einen GPIO

Parameter	*Erklärung*
gpio	*Nummer des GPIOs, beginnend mit 0 für GPIO1*
dir	*Einstellung der Signalrichtung:* *RM2M_GPIO_DISABLED : GPIO deaktiviert* *RM2M_GPIO_INPUT : Eingang* *RM2M_GPIO_OUTPUT : Ausgang*

	Erklärung
Rückgabewert	• *OK, wenn erfolgreich* • *< OK, wenn ein Fehler auftritt (siehe "Returncodes für allgemeine Zwecke" im Kapitel "Konstanten" auf Seite 86)*

native rM2M_GpioSet(gpio, level);

setzt das Ausgangslevel eines GPIOs, der als Ausgang konfiguriert wurde

Parameter	*Erklärung*
gpio	*Nummer des GPIOs, beginnend mit 0 für GPIO1*
level	*Angabe des auszugebenden Levels:* *RM2M_GPIO_LOW : Ausgangslevel auf "low"* *RM2M_GPIO_HIGH : Ausgangslevel auf "high"*

	Erklärung
Rückgabewert	• *OK, wenn erfolgreich* • *ERROR_BAD_STATE, wenn der betreffende GPIO nicht als Ausgang konfiguriert wurde* • *< OK, wenn ein anderer Fehler auftritt (siehe "Returncodes für allgemeine Zwecke" im Kapitel "Konstanten" auf Seite 86).*

native rM2M_GpioGet(gpio);

liest das Signallevel eines GPIOs, der als Eingang konfiguriert wurde

Parameter	*Erklärung*
gpio	*Nummer des GPIOs, beginnend mit 0 für GPIO1*

	Erklärung
Rückgabewert	• *RM2M_GPIO_LOW für "low" am Eingang* • *RM2M_GPIO_HIGH für "high" am Eingang* • *ERROR_BAD_STATE, wenn der betreffende GPIO nicht als Eingang konfiguriert wurde* • *< OK, wenn ein anderer Fehler auftritt (siehe "Returncodes für allgemeine Zwecke" im Kapitel "Konstanten" auf Seite 86).*

native rM2M_IrqInit(irq, config, funcidx);

aktiviert die Interrupt-Funktionalität eines interruptfähigen Pins

Parameter	*Erklärung*
irq	*Nummer des interruptfähigen Pins, beginnend mit 0 für INT0*
config	*Auswahl der Flanke, die den Interrupt auslösen soll* *RM2M_IRQ_RISING : bei steigender Flanke* *RM2M_IRQ_FALLING : bei fallender Flanke*
funcidx	*Index der öffentlichen Funktion, die im Falle eines Interrupts ausgeführt werden soll* *Typ der Funktion: public func(pinstate);*

	Erklärung

Rückgabewert	• *OK, wenn erfolgreich*
	• *ERROR_ALREADY_SUBSCRIBED, wenn die Interrupt-Funktionalität für den betreffenden Pin bereits initialisiert wurde*
	• *ERROR, wenn einer der folgenden Fehler auftritt*
	  ◦ *Interrupt-Funktionalität für diesen Pin nicht verfügbar*
	  ◦ *Die im Fall eines Interrupts auszuführende Funktion ist nicht öffentlich.*
	• *< OK, wenn ein anderer Fehler auftritt (siehe "Returncodes für allgemeine Zwecke" im Kapitel "Konstanten" auf Seite 86).*

native rM2M_IrqClose(irq);

deaktiviert die Interrupt-Funktionalität eines interruptfähigen Pins

Parameter	*Erklärung*
irq	*Nummer des interruptfähigen Pins, beginnend mit 0 für INT0*

	Erklärung
Rückgabewert	• *OK, wenn erfolgreich* • *< OK, wenn ein Fehler auftritt (siehe "Returncodes für allgemeine Zwecke" im Kapitel "Konstanten" auf Seite 86)*

14.3.6 SPI, I2C, UART

14.3.6.1 Konstanten

Konfiguration von Clock-Polarität und Clock-Phase des SPI-Interfaces

Konfigurationsflags für die Funktion rM2M_SpiInit()

```
RM2M_SPI_CLKPOL    = 0b00000001, // Clock-Polarität: Idle "high"
RM2M_SPI_CLKPHA    = 0b00000010, /* Clock-Phase: Daten werden bei der ersten
                                    Flanke übernommen. */
```

Konfiguration des UART-Interfaces

Konfigurationsflags für die Funktion rM2M_UartInit()

```
RM2M_UART_1_STOPBIT   = 0b000000001, // 1 Stopbit
RM2M_UART_2_STOPBIT   = 0b000000010, // 2 Stopbit
RM2M_UART_PARITY_NONE = 0b000000000, // keine Parität
RM2M_UART_PARITY_ODD  = 0b000000100, // ungerade Parität
RM2M_UART_PARITY_EVEN = 0b000001000, // gerade Parität
RM2M_UART_7_DATABIT   = 0b000000000, // 7 Datenbits
RM2M_UART_8_DATABIT   = 0b000010000, // 8 Datenbits
RM2M_UART_FLOW_NONE   = 0b000000000, // keine Flusskontrolle
RM2M_UART_FLOW_RTSCTS = 0b001000000, // RTS/CTS Handshake
RM2M_UART_FULL_DUPLEX = 0b000000000, // Full Duplex Modus
RM2M_UART_HALF_DUPLEX = 0b100000000, // Half Duplex Modus
```

14.3.6.2 Callback Funktionen

public func(const data{}, len);

vom Script-Entwickler bereitzustellende Funktion, die beim Empfang von Zeichen über die UART-Schnittstelle aufgerufen wird.

Parameter	*Erklärung*
data	*Array, das die empfangenen Daten enthält*
len	*Anzahl der empfangenen Bytes*

14.3.6.3 Funktionen

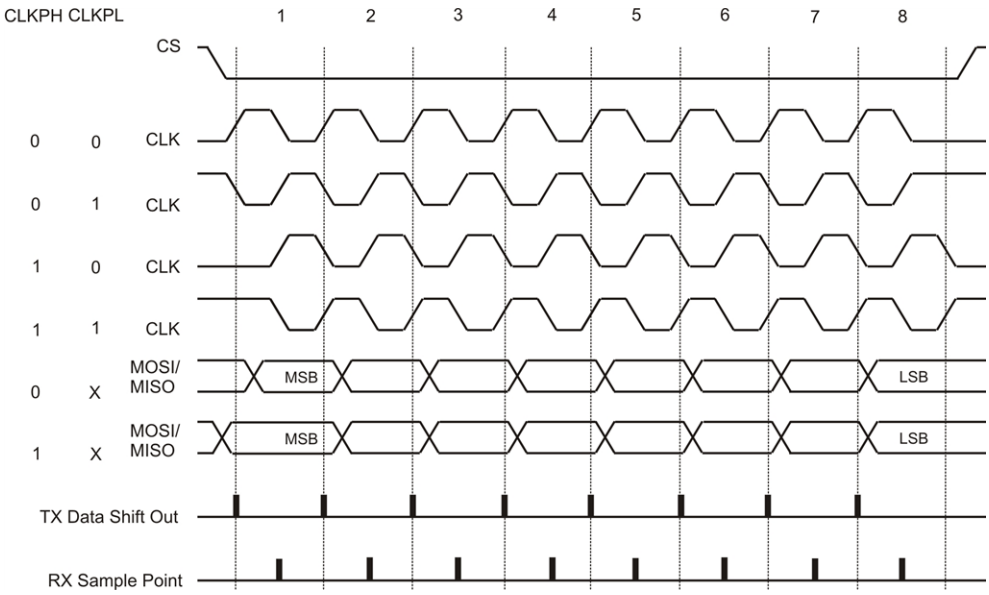
native rM2M_SpiInit(spi, clock, config);

initialisiert das SPI-Interface

Parameter	*Erklärung*
spi	*Nummer des SPI-Interfaces, beginnend mit 0 für SPI1*
clock	*zu verwendende Clockfrequenz in Hz. Bitte beachten Sie die für das verwendete Modul gültigen Grenzwerte (siehe).*
config	*Konfiguration von Clock-Polarität und Clock-Phase* *Bit0: Clock-Polarität* *0 = Idle "low"* *1 = Idle "high"* *Bit1: Clock-Phase* *0 = Daten werden bei der ersten Flanke ausgegeben.* *1 = Daten werden bei der ersten Flanke übernommen.* ***Hinweis:** Sie können für diesen Parameter auch die vordefinierten Konstanten verwenden (siehe "Konfiguration von Clock-Polarität und Clock-Phase des SPI-Interface" im Kapitel "Konstanten" auf Seite 114). Die Konstanten lassen sich auch durch "oder"-Verknüpfung kombinieren.*

	Erklärung
Rückgabewert	• *OK, wenn erfolgreich* • *ERROR, wenn die Nummer des SPI-Interfaces ungültig ist* • *ERROR_ALREADY_SUBSCRIBED, wenn das SPI-Interface bereits initialisiert wurde* • *< OK, wenn ein anderer Fehler auftritt (siehe "Returncodes für allgemeine Zwecke" im Kapitel "Konstanten" auf Seite 86).*

***Hinweis:** Ergänzende Erklärung zur Konfiguration von Clock-Polarität und Clock-Phase*



native rM2M_SpiClose(spi);

schließt das SPI-Interface

Parameter	*Erklärung*
spi	*Nummer des SPI-Interfaces, beginnend mit 0 für SPI1*

	Erklärung
Rückgabewert	• *OK, wenn erfolgreich* • *ERROR, wenn die Nummer des SPI-Interfaces ungültig ist* • *< OK, wenn ein anderer Fehler auftritt (siehe "Returncodes für allgemeine Zwecke" im Kapitel "Konstanten" auf Seite 86).*

native rM2M_SpiCom(spi, data{}, txlen, rxlen);

führt eine asynchrone SPI Kommunikation aus. Zuerst werden Daten versendet, danach Daten empfangen.

Parameter	*Erklärung*
spi	*Nummer des SPI-Interfaces, beginnend mit 0 für SPI1*
data	*Array, in dem zunächst die zu versendenden Daten gespeichert sein müssen. Wurden die Daten versendet, wird das Array als Speicher für die zu empfangenden Daten verwendet.*
txlen	*Anzahl der zu sendenden Bytes*
rxlen	*Anzahl der zu empfangenden Bytes*

	Erklärung

Rückgabewert	• *OK, wenn erfolgreich*
	• *ERROR, wenn einer der folgenden Fehler auftritt*
	  ◦ *Anzahl der zu sendenden Bytes > 255*
	  ◦ *Anzahl der zu empfangenden Bytes > 255*
	• *< OK, wenn ein anderer Fehler auftritt (siehe "Returncodes für allgemeine Zwecke" im Kapitel "Konstanten" auf Seite 86).*

native rM2M_I2cInit(i2c, clock, config);

initialisiert das I²C-Interface

Parameter	*Erklärung*
i2c	*Nummer des I²C-Interfaces, beginnend mit 0 für I2C1*
clock	*zu verwendende Clockfrequenz in Hz. Bitte beachten Sie die für das verwendete Modul gültigen Grenzwerte (siehe).*
config	*reserviert für Erweiterungen*

	Erklärung
Rückgabewert	• *OK, wenn erfolgreich* • *ERROR, wenn die Nummer des I²C-Interfaces ungültig ist* • *ERROR_ALREADY_SUBSCRIBED, wenn das I²C-Interface bereits initialisiert wurde* • *< OK, wenn ein anderer Fehler auftritt (siehe "Returncodes für allgemeine Zwecke" im Kapitel "Konstanten" auf Seite 86).*

native rM2M_I2cClose(i2c);

schließt das I²C-Interface

Parameter	*Erklärung*
i2c	*Nummer des I²C-Interfaces, beginnend mit 0 für I2C1*

	Erklärung
Rückgabewert	• *OK, wenn erfolgreich* • *ERROR, wenn die Nummer des I²C-Interfaces ungültig ist* • *< OK, wenn ein anderer Fehler auftritt (siehe "Returncodes für allgemeine Zwecke" im Kapitel "Konstanten" auf Seite 86).*

native rM2M_I2cCom(i2c, adr, data{}, txlen, rxlen);

führt eine I²C Kommunikation aus. Zuerst werden Daten versendet, danach Daten empfangen.

Parameter	*Erklärung*
i2c	*Nummer des I²C-Interfaces, beginnend mit 0 für I2C1*
adr	*Adresse des I²C Slaves (Bit7-Bit1, Bit0 unused)*
data	*Array, in dem zunächst die zu versendenden Daten gespeichert sein müssen. Wurden die Daten versendet, wird das Array als Speicher für die zu empfangenden Daten verwendet.*
txlen	*Anzahl der zu sendenden Bytes*
rxlen	*Anzahl der zu empfangenden Bytes*

	Erklärung

Rückgabewert	• *OK, wenn erfolgreich*
	• *ERROR, wenn einer der folgenden Fehler auftritt*
	  ◦ *Anzahl der zu sendenden Bytes > 255*
	  ◦ *Anzahl der zu empfangenden Bytes > 255*
	• *< OK, wenn ein anderer Fehler auftritt (siehe "Returncodes für allgemeine Zwecke" im Kapitel "Konstanten" auf Seite 86).*

native rM2M_UartInit(uart, baudrate, mode, funcidx);

initialisiert das UART-Interface

Parameter	*Erklärung*
uart	*Nummer des UART-Interfaces, beginnend mit 0 für UART1*
baudrate	*zu verwendende Baudrate. Bitte beachten Sie die für das verwendete Modul gültigen Grenzwerte (siehe).*
mode	*Bit 0...1* *1 = 1 Stopbit* *2 = 2 Stopbit* *Bit 2...3* *0 = keine Parität* *1 = ungerade Parität* *2 = gerade Parität* *Bit 4...5* *0 = 7 Datenbits* *1 = 8 Datenbits* *Bit 6...7* *0 = keine Flusskontrolle* *1 = RTS/CTS Handshake* *Bit 8* *0 = Full Duplex Modus* *1 = Half Duplex Modus* ***Hinweis:*** *Sie können für diesen Parameter auch die vordefinierten Konstanten verwenden (siehe "Konfiguration des UART-Interface" im Kapitel "Konstanten" auf Seite 114). Die Konstanten lassen sich auch durch "oder"-Verknüpfung kombinieren.*
funcidx	*Index der öffentlichen Funktion für den UART-Zeichenempfang* *Typ der Funktion: public func(const data{}, len);*

	Erklärung
Rückgabewert	• *OK, wenn erfolgreich* • *< OK, wenn ein Fehler auftritt (siehe "Returncodes für allgemeine Zwecke" im Kapitel "Konstanten" auf Seite 86)*

native rM2M_UartSetbuf(uart, rxbuf{}, rxlen, txbuf{}, txlen);

stellt der Firmware für das gewählte UART-Interface einen Sende- und/oder Empfangspuffer aus dem für das Script reservierten RAM-Bereich zur Verfügung. Beim Aufruf dieser Funktion wird von den beiden in die Firmware integrierten Puffern auf die übergebenen Puffer umgeschaltet.

> ***Wichtiger Hinweis:*** *Diese Funktion muss vor dem Initialisieren des UART-Interface durch die Funktion "rM2M_UartInit()" aufgerufen werden.*

Parameter	***Erklärung***
uart	*Nummer des UART-Interfaces, beginnend mit 0 für UART1*
rxbuf	*statisches Byte-Array, das als Empfangspuffer verwendet werden soll*
rxlen	*Größe des Empfangspuffers in Byte* ***Hinweis:*** *Wird die Funktion erneut aufgerufen und dabei die Größe auf "0" gesetzt, wird wieder auf den integrierten Empfangspuffer zurückgeschaltet. Das übergebene statische Byte-Array kann anschließend wieder durch das Script verwendet werden.*
txbuf	*statisches Byte-Array, das als Sendepuffer verwendet werden soll*
txlen	*Größe des Sendepuffers in Byte* ***Hinweis:*** *Wird die Funktion erneut aufgerufen und dabei die Größe auf "0" gesetzt, wird wieder auf den integrierten Sendepuffer zurückgeschaltet. Das übergebene statische Byte-Array kann anschließend wieder durch das Script verwendet werden.*

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

native rM2M_UartClose(uart);

schließt das UART-Interface

Parameter	***Erklärung***
uart	*Nummer des UART-Interfaces, beginnend mit 0 für UART1*

	Erklärung
Rückgabewert	• *OK, wenn erfolgreich* • *< OK, wenn ein Fehler auftritt (siehe "Returncodes für allgemeine Zwecke" im Kapitel "Konstanten" auf Seite 86)*

native rM2M_UartWrite(uart, const data{}, len);

versendet Daten über das angegebene UART-Interface

Parameter	*Erklärung*
uart	*Nummer des UART-Interfaces, beginnend mit 0 für UART1*
data	*Array, das die zu sendenden Daten enthält*
len	*Anzahl der zu sendenden Bytes*

	Erklärung
Rückgabewert	• *Anzahl der gesendeten Bytes, wenn erfolgreich* • *ERROR, wenn die Anzahl der zu sendenden Bytes > 256* • *< OK, wenn ein anderer Fehler auftritt (siehe "Returncodes für allgemeine Zwecke" im Kapitel "Konstanten" auf Seite 86).*

14.3.7 Registry

14.3.7.1 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 der Produktion (Empfehlung)
RM2M_REG_SYS_FLASH = 1, // im Betrieb veränderbar (nicht durch Script)

//Applikationsspezifische Daten
RM2M_REG_APP_OTP   = 2, // einmalig im Zuge der Produktion (Empfehlung)
RM2M_REG_APP_FLASH = 3, // im Betrieb veränderbar (auch durch Script)

//Applikationsspezifische, flüchtige Daten
RM2M_REG_APP_STATE = 4, // im Betrieb veränderbar (nur durch Script)

//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)
```

14.3.7.2 Callback Funktionen

public func(reg);

vom Script-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 121), der geändert wurde*

14.3.7.3 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 121)*

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 121)*
name	*Name des Eintrags*
string	*Array zur Aufnahme des zu lesenden Strings*
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 121)*

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 121)*
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 121)*

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.

Parameter	***Erklärung***
reg	*Index des Registrierungsspeicherblocks (siehe "Indizes der Registrierungsspeicherblöcke" im Kapitel "Konstanten" auf Seite 121)*
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 121)*

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 121)*
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 121)*

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 121)*
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 121)*

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 121)*
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 121)*

native rM2M_RegOnChg(funcidx);

legt die Funktion fest, die aufgerufen werden soll, wenn sich einer der Registrierungsspeicherblöcke geändert hat. 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 86)*

14.3.8 Position

14.3.8.1 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 129)
// SatUsed     Anzahl der zur Positionsbestimmung verwendeten Satelliten
// HDOP        relative Genauigkeit der horizontalen Position [0,01]

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

14.3.8.2 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)
```

14.3.8.3 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 193) 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 129)* *[1] : enthält den Typ des decodierten NMEA-Datensatzes (siehe "Liste der unterstützten NMEA-Datensätze" im Kapitel "Konstanten" auf Seite 129)* *[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 129)*

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 193) 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 129)*

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 86)*

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 129)*
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 127)*
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 127)*

	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*

14.3.9 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);
Wurzel 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

14.3.10 Char & String

Hinweis: *Um die Funktionen dieses Kapitels verwenden zu können, benötigen Sie folgendes Include-File:* `#include <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

Parameter	***Erklärung***
dest	*Array zur Aufnahme des formatierten Ergebnisses*
maxlength	*maximale Zeichenanzahl, die das dest Array aufnehmen kann*
format	*die zu verwendende Format-Zeichenkette*

	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')

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

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

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

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

	Erklärung
Rückgabewert	• *-1, wenn das gesuchte Zeichen nicht in der Zeichenkette enthalten ist* • *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* • *Array-Index an der die zu suchende Zeichenkette string2 im string1 beginnt*

native strtol(const string[], base);

wandelt eine Zeichenkette in einen Wert um

Parameter	*Erklärung*
string	*umzuwandelnde Zeichenkette*
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*

	Erklärung
Rückgabewert	*Wert, der der Zeichenkette entspricht*

native Float: atof(const string[]);

wandelt eine Zeichenkette in einen Float um

Parameter	*Erklärung*
string	*umzuwandelnde Zeichenkette*

	Erklärung
Rückgabewert	*Float, dessen Zahlenwert der Zeichenkette entspricht*

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.*

14.3.11 CRC & Hash

14.3.11.1 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]]
```

14.3.11.2 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*

	Erklärung
Rückgabewert	*---*

14.3.12 Verschiedene Funktionen

14.3.12.1 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 M2
          HW1.3")
// module     rapidM2M Modultyp (z.B. "M2")
// 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 Pawn-Instructions
// tmp           für interne Verwendung, kein Schreibzugriff gestattet

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

14.3.12.2 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
};
```

14.3.12.3 Funktionen

native getapilevel();

gibt das implementierte API-Level der Skript-Engine aus

	Erklärung
Rückgabewert	*implementiertes API-Level der Skript-Engine*

native exists(const name[]);

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

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 Pawn-Engine um eigene native Funktionen erweitert werden. Die Implementierung von Operationen als native Funktion ermöglicht eine deutliche Steigerung der Abarbeitungsgeschwindigkeit im Vergleich zur Umsetzung in Pawn. 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 Pawn-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

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 Pawn-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);

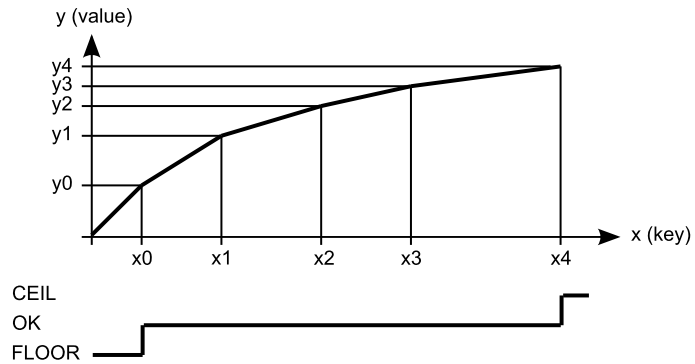
*sucht einen bestimmten Wert in der "key"-Spalte der übergebenen Stützpunktabelle und liefert den entsprechenden Wert der "value"-Spalte der Tabelle. Liegt der gesuchte Wert zwischen zwei Stützpunkten, wird der Rückgabewert zwischen den zwei angrenzenden "value"-Spaltenwerten linear interpoliert (Geradengleichung: 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 86).*

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 146)*
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 86).* ***Hinweis:** Die Firmware des Moduls/Geräts erkennt, wenn ein Script verwendet wird, bei dem die Funktion nur einen Übergabeparameter (Verwendung eines älteren Include-Files) besitzt und liefert aus Kompatibilitätsgründen "OK" anstelle der Gösse 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

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

	Erklärung
Rückgabewert	*Die Anzahl der Argumente, die an eine Funktion übergeben wurden. numargs ist nützlich innerhalb von Funktionen mit einer variablen Argumentenliste.*

native getarg(arg, index=0);

liefert 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*

	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*

14.3.13 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.

Parameter	*Erklärung*
format[]	*die zu verwendende Format-Zeichenkette*

	Erklärung
Rückgabewert	• *Anzahl der gedruckten Zeichen* • *ERROR, falls nicht erfolgreich*

native setbuf(buf{}, size);

stellt der Firmware einen Puffer aus dem für das Script 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.

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 das Script verwendet werden.*

	Erklärung
Rückgabewert	• *OK, wenn erfolgreich* • *ERROR, falls nicht erfolgreich*

14.3.14 SMS

Wichtiger Hinweis: *Wenn sich das Gerät im "online"-Modus befindet, können keine SMS verarbeitet werden.*

14.3.14.1 Callback Funktionen

public func(const SmsTel[], const SmsText[]);

vom Script-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*

14.3.14.2 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 86)*

native rM2M_SmsClose();

deaktiviert den SMS-Empfang

	Erklärung
Rückgabewert	*OK*

14.3.15 File Transfer

14.3.15.1 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 155)
#define TFT_Info[  .name{256}, .stamp, .stamp256, .size, .crc, .flags ]
```

14.3.15.2 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 vom Script 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.
```

14.3.15.3 Callback Funktionen

public func(id, cmd, const data{}, len, ofs);

vom Script-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 155) 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.</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
FT_CMD_STORE: Größe des vom myDatanet-Server erhaltenen Dateieigenschaftenblocks
FT_CMD_WRITE: Anzahl der vom myDatanet-Server erhaltenen Bytes</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</td></tr>
</table>

14.3.15.4 Funktionen

native FT_Register(const name{}, id, funcidx);

registriert eine Datei, die durch das PAWN-Script 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 155) 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 86)*

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 86)*

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 155)*

	Erklärung
Rückgabewert	• *OK, wenn erfolgreich* • *< OK, wenn ein Fehler auftritt (siehe "Returncodes für allgemeine Zwecke" im Kapitel "Konstanten" auf Seite 86)*

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 155)*
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 86)*

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	*Arry, 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 86)*

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 156) 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 86)*

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 156) 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 86)*

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 86)*

14.4 Pawn Script Fehlercodes

Sollte beim Durchlaufen des Pawn Scripts ein Fehler auftreten, wird der entsprechende Fehlercode ins Gerätelog eingetragen und das Script neu gestartet. Kommt es innerhalb von 24h mehr als 3 Mal zu einem derartiger Fehler, wird das Script deaktiviert und das Errorhandling aktiviert (siehe "Errorhandling" auf Seite 35). Bei allen Log-Einträgen bis auf "SCRIPT_ERR" enthält der Parameter den 32-Bit Instruction Pointer der PAWN 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 "Kateireiter "Log""(siehe "Karteireiter "Log"" auf Seite 75).

Log-Eintrag		Parameter		Beschreibung
Code	**Klartext**	**Code**	**Klartext**	
3000	SCRIPT_ERR	0	NO SCRIPT	kein gültiges Script vorhanden
		1	SCRIPT UPDATE	neues Script erhalten
		2	SCRIPT EXCEPT LOOP	Exception Loop erkannt (4 Systemstarts nach Exeption innerhalb von 10min.) Das Script wird deaktiviert und das Errorhandling aktiviert (siehe "Errorhandling" auf Seite 35). 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. Script wurde neu gestartet.
		4	SCRIPT SOFT ERROR 2	Laufzeitfehler zum zweiten Mal innerhalb von 24h aufgetreten. Script wurde neu gestartet.
		5	SCRIPT SOFT ERROR 3	Laufzeitfehler zum dritten Mal innerhalb von 24h aufgetreten. Script wurde neu gestartet. Sollte innerhalb von 24h ein weiterer Laufzeitfehler auftreten, wird das Script deaktiviert und das Errorhandling aktiviert (siehe "Errorhandling" auf Seite 35).
		6	SCRIPT UPDATE ERROR	Verbindungsabbruch während des Scriptdownloads Das bestehende Script kann dadurch, dass es 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.
3001	AMX_ERR_EXIT	##	---	Abbruch z.B. Max. Anzahl der PAWN-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

Log-Eintrag		Parameter		Beschreibung
Code	**Klartext**	**Code**	**Klartext**	
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
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.

Log-Eintrag		Parameter		Beschreibung
Code	Klartext	Code	Klartext	
3027	AMX_ERR_ GENERAL	##	---	allgemeiner Fehler (unbekannter oder nicht spezifizierter Fehler)
3028	AMX_ERR_ OVERLAY	##	---	Overlays werden nicht unterstützt (JIT) oder sind nicht initialisiert.

14.5 Syntax

14.5.1 Allgemeine Syntax

14.5.1.1 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.

14.5.1.2 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.

14.5.1.3 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.

14.5.1.4 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. Pawn unterscheidet zwischen Groß- und Kleinschreibung. Der Parser schneidet Bezeichner ab einer bestimmten Länge ab. Es werden standardmäßig nur die ersten 16 Zeichen für die Unterscheidung herangezogen.

14.5.1.5 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			

14.5.1.6 Numerische Konstanten

14.5.1.6.1 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

14.5.1.6.2 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.

14.5.2 Variablen

14.5.2.1 Deklaration

Das Schlüsselwort "new" deklariert eine neue Variable. Für spezielle Deklarationen wird das Schlüsselwort "new" durch "static" ersetzt (siehe "Statische lokale Deklaration" auf Seite 165). 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 176) - lokale Variable

Beispiel:

```
new a;     // ohne Initialisierung (Wert ist 0)
new b = 3; // mit Initialisierung (Wert ist 3)
```

14.5.2.2 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.

14.5.2.3 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.

14.5.2.4 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.

14.5.2.5 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".

14.5.2.6 Gleitkommawerte

Pawn unterstützt Gleitkommawerte. 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)
```

14.5.3 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 179.

14.5.4 Array Variablen

14.5.4.1 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.

14.5.4.2 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. */
```

14.5.4.3 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
```

14.5.4.4 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 167).

14.5.4.5 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.

14.5.5 Operatoren und Ausdrücke

14.5.5.1 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*

14.5.5.2 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
```

14.5.5.3 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 in Pawn nicht definiert. Die Operatoren ++ und -- ändern den Operanden. Der Operand muss ein "lvalue" sein.*

14.5.5.4 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.

14.5.5.5 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

14.5.5.6 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.

14.5.5.7 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.

14.5.5.8 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.

14.5.5.9 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

14.5.6 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.

14.5.6.1 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.

14.5.6.2 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.

14.5.6.3 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.

14.5.6.4 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.

14.5.6.5 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
}
```

14.5.6.6 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
}
```

14.5.6.7 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
}
```

14.5.6.8 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
}
```

14.5.6.9 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.

14.5.6.10 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)".

14.5.6.11 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.

14.5.6.12 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
}
```

14.5.6.13 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.

14.5.6.14 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
}
```

14.5.6.15 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
}
```

14.5.7 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 168 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.

14.5.7.1 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 */
```

14.5.7.2 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.

14.5.7.3 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]
}
```

14.6 Beispiele

t.b.d.

14.7 Unterschiede zu C

- Pawn fehlt der Eingabe-Mechanismus von C. Pawn ist 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. Werte wie ".5" und "6." sind in C akzeptabel, aber im Pawn muss man "0.5" und "6.0" schreiben. In C ist der Dezimalpunkt optional, wenn ein Exponent enthalten ist, so kann man in C "2E8" schreiben; Pawn akzeptiert den Großbuchstaben "E" nicht. Verwenden Sie den Kleinbuchstaben "e". Es erfordert das Komma: z.B. "2.0e8" (siehe "Numerische Konstanten" auf Seite 164).
- Pawn unterstützt keine "Zeiger". Für die Übergabe von Funktionsparametern als Referenz bietet Pawn ein "Referenz"-Argument (siehe "Funktionsargumente ("call-by-value" versus "call-by-reference")" auf Seite 179). Das "Platzhalter"-Argument ersetzt einige Verwendungen des NULL-Zeigers (siehe "Standardwerte von Funktionsargumenten" auf Seite 181).
- Zahlen können mit Hexadezimal-, Dezimal-, oder Binärbasis angegeben werden. Die Oktale Basis wird nicht unterstützt (siehe "Numerische Konstanten" auf Seite 164). 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 177). In C/C++ ist die "switch"-Anweisung ein "bedingtes goto". In Pawn ist die "switch"-Anweisung ein strukturiertes "if".
- Eine "break"-Anweisung beendet nur Schleifen. In C/C++ beendet die "break"-Anweisung auch ein "case" in einer "switch"-Anweisung.
- Pawn unterstützt "array Zuweisungen", mit der Limitation, 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, unterstützt Pawn auch literale Arrays und somit Ausdrücke wie "a = {0,1,2,3,4,5}", wobei "a" eine Array Variable mit 6 Elementen ist.
- "defined" ist ein Operator und keine Präprozessor-Direktive. Der "defined" Operator in Pawn 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 171.
- Eine leere Anweisung ist ein leerer Block (mit {}), nicht ein Semikolon (siehe "Zusammengesetzte Anweisungen" auf Seite 173). 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 168).
- 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" in Pawn implementiert die "enum" Funktionalität von C.
- In den meisten Fällen sind Vorwärts-Deklarationen von Funktionen (d.h. Prototypen) nicht notwendig. Pawn ist ein 2-Pass-Compiler. 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. Pawn kümmert sich um Parameter-Namen im Prototyp auf Grund der "benannte Parameter"-Funktion. Pawn verwendet Prototypen, 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.

Kapitel 15 myDatanet Connector

Das Grundprinzip des EC-CAN-00x ist eine sogenannte Storage-2-Storage-Datenübertragung. Für diese Art der Datenübermittlung müssen weder das EC-CAN-00x 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 EC-CAN-00x an den Server übermittelten Daten sind somit frei gestaltbar. Es stehen 1024 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 EC-CAN-00x 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 myDatanet Connector beinhaltet sowohl das Werkzeug für die Beschreibung der Daten, als auch die korrekte Bereitstellung der Daten zur Verwendung mit der Oberfläche.

15.1 Voraussetzungen

> ***Wichtiger Hinweis:*** *Bitte lesen Sie dieses Kapitel sorgfältig durch bevor Sie den myDatanet Connector verwenden! Es ist wichtig, diese Vorgaben einzuhalten, da die korrekte Funktion des myDatanet Connector ansonsten nicht gewährleistet werden kann.*

Um den myDatanet Connector für rapidM2M verwenden zu können, müssen einige Voraussetzungen erfüllt werden, die die grundsätzliche Dynamik von rapidM2M zwar einschränken, jedoch nötig sind, um die Daten auch in myDatanet verfügbar machen zu können.

- Verwendung einer Daten-Präambel (gilt nur für Daten)

  zu Beginn des Datensatzes, um die verschiedenen Nutzdatenblöcke (z.B. Messdaten, Alarme, Log-Einträge) voneinander zu separieren. Es wird empfohlen, die ersten ein oder zwei Bytes hierfür zu verwenden. Die Daten-Präambel wird in den <split>-tags der Datenstruktur-Beschreibung verwendet.

  Beispiel für die Verwendung eines Bytes:

  0x01 ... Messdaten
  0x02 ... Alarme
  0x03 Log-Einträge

- Vollständigkeit des Nutzdatenblocks (gilt nur für Daten)

  Nutzdatenblöcke sind immer mit allen Datenfeldern zu befüllen, auch wenn sich nur ein oder zwei Werte seit der letzten Übertragung verändert haben. Die Daten werden vom myDatanet Connector mit Byte-Offset und Länge aus dem Nutzdatenblock extrahiert, daher müssen sich die Felder immer an der gleichen Position befinden.

- Homogenität des Nutzdatenblocks (gilt nur für Daten)

  Ein Nutzdatenblock der mittels Split-tag (siehe "Split-tag" auf Seite 186) in einen der strukturierten Messdatenkanäle ("histdata0" - "histdata9") kopiert wird, muss immer gleich aufgebaut sein (d.h. die selben Datenfelder enthalten). Werden unterschiedliche Nutzdatenblockstrukturen verwendet, muss jeder Variante mittels Split-tag ein eigener strukturierter Messdatenkanal zugeordnet werden.

- Verwendung von Big Endian (gilt sowohl für Daten als auch für Konfigurationen)

  Im folgenden Beispiel wird die Ganzzahl 439.041.101 als dword (u32) ab Speicheradresse 10000 gespeichert.

	Big Endian			Little Endian		
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

15.2 Split-tag

Ein Split-tag wird verwendet, um die vom Gerät erhaltenen Daten serverseitig richtig zuzuordnen.

Ein EC-CAN-00x übermittelt seine Daten im sogenannten "rm2mraw"-Bereich. Dieser Bereich ist für die reine Storage-2-Storage-Verwendung des Servers vorgesehen. Sollen alle oder bestimmte Daten für Auswertungen, Ansichten, Grafiken, etc. in myDatanet verfügbar gemacht werden, muss ein Split-tag konfiguriert werden.

> ***Wichtiger Hinweis:*** *Die Split-tags werden nicht auf bereits im System vorhandene Daten angewandt. Beim Empfang jedes rapidM2M -Rohdatensatzes ("rm2mraw") werden die erste Bytes (je nach Länge des "key" im entsprechenden Split-tag) ausgewertet. Bei Übereinstimmung der ersten Bytes mit dem spezifizierten "key" wird der Datensatz in den mittels "target" angegebenen strukturierten Messdatenkanal kopiert. Eine nachträgliche Verarbeitung ist nicht vorgesehen.*
>
> ***Wichtiger Hinweis:*** *Wollen Sie nur den Rohdatensatzes ("rm2mraw") verwenden, ist die Konfiguration eines Split-tags nicht erforderlich.*

Beispiel für einen Split-tag:

Code	Erklärung
`<split>`	Beginn des Split-tags
`source =rm2mraw`	Quelle der Daten, die kopiert werden sollen
`target =histdata0`	Ziel, in das die Daten kopiert werden sollen
`key =*`	Schlüssel für die Auftrennung der Daten
`</split>`	Ende des Split-tags

source

Tabellenname der Quelle
Bei rapidM2M ist die Quelle immer `rm2mraw`*.*

target

Tabellenname des Ziels
Die strukturierten Messdatenkanäle sind bei rapidM2M mit "histdata0" - "histdata9" bezeichnet.

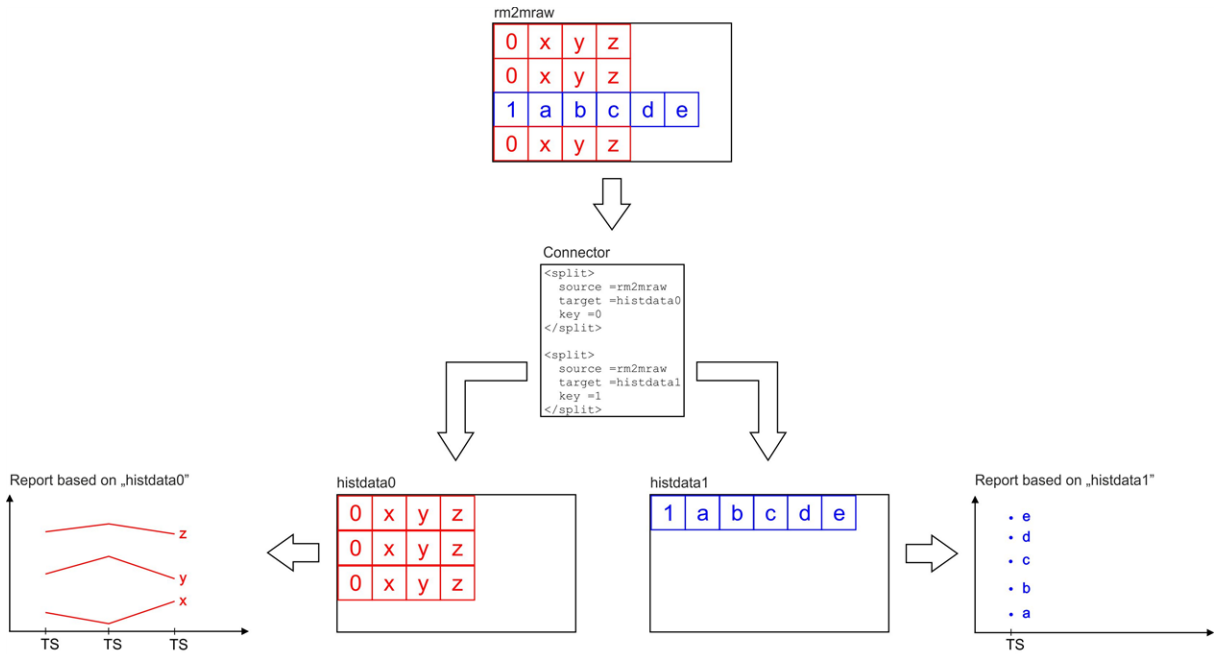
key

Dieses Feld bietet die Möglichkeit zu bestimmen, ob der aktuelle Datensatz kopiert werden soll:

- *Bei Angabe eines Sterns (*) werden alle Datensätze von source nach target kopiert.*
- *Bei Angabe eines Wertes im HEX-Format wird überprüft, ob der Beginn des empfangenen Datensatzes (auch mehr als das erste Byte bei entsprechender Angabe) diesem Wert entspricht. Bei Übereinstimmung wird dieser Datensatz kopiert.*

Es können mehrere Split-tags angegeben werden. Der Datensatz wird beim ersten passenden key-Eintrag kopiert. Die darauf folgenden Split-tags werden nicht mehr berücksichtigt.

> ***Hinweis:*** *Ergänzende Erklärung zur Funktionsweise der Split-tags*
>
> *Im folgenden Beispiel existieren zwei Nutzdatenblockstrukturen, die mittels myDatanet Connector vom unstrukturierten Bereich ("rm2mraw") in unterschiedliche strukturierte Messdatenkanäle ("histdata0" und "histdata1") kopiert werden sollen, um sie in Verbindung mit der myDatanet Oberfläche nutzen zu können (Auswertungen, Visualisierungen, Grafiken, etc.). Das erste Byte zu Beginn des Datenbereich eines Nutzdatenblocks wird dazu verwendet, den Typ der Nutzdatenblockstruktur anzugeben. Der Wert 0 gibt an, dass der Nutzdatenblock in den strukturierten Messdatenkanal "histdata0" kopiert werden soll. Ist das erste Byte 1, soll der Nutzdatenblock in den strukturierten Messdatenkanal "histdata1" kopiert werden. Jeder Nutzdatenblock verfügt auch über Header-Informationen (Zeitstempel, CRC,...), die in der folgenden Grafik zugunsten der Übersichtlichkeit weggelassen wurden.*



Graphische Darstellung der Funktionsweise der Split-tags

15.3 Datenstruktur

Dieser Abschnitt beschreibt wie strukturierte Messdatenkanäle ("histdata0" - "histdata9") sowie Konfigurationsspeicherblöcke ("config0" - "config9") für die Verwendung am myDatanet-Server in einzelne Datenfelder aufgeteilt werden.

> ***Wichtiger Hinweis:*** *Soll ein strukturierter Messdatenkanal oder Konfigurationsblock am myDatanet-Server oder über die REST-API verfügbar sein, müssen alle Datenfelder mittels myDatanet Connector definiert werden.*

Ein erweitertes Beispiel, in dem die meisten der verfügbaren Attribute verwendet werden, finden Sie im Kapitel "Beispiel" auf Seite 191.

Beispiel für die Aufteilung eines strukturierten Messdatenkanals ("histdata0") in einzelne Datenfelder

Code	Erklärung

`<table>`
`  name  =histdata0`
`  <field>`
`  name  =ch0`
`  title  =Verzögerung`
`  units  =sec`
`  type  =u16`
`  byteofs =0`
`  vofs  =10`
`  min  =10`
`  max  =2000`
`  chmode  =3`
`  </field>`
`</table>`
	Beginn des Table-tags
	Name der zu beschreibenden Datentabelle
	Beginn einer Feld-Definition (ch0) für ein Feld des Typs u16, Beschreibung der Attribute siehe unten
	
	Ende einer Feld-Definition (ch0)

Beispiel für die Aufteilung eines Konfigurationsspeicherblocks ("config0") in einzelne Datenfelder

Code	Erklärung

`<table>`
`  name  =config0`
`  <field>`
`  name  =field0`
`  title  =Aufzeichnungsitv`
`  units  =s`
`  type  =u32`
`  byteofs=0`
`  </field>`
`</table>`
	Beginn des Table-tags
	Name der zu beschreibenden Konfigurationstabelle
	Beginn einer Feld-Definition (field0) für ein Feld des Typs u32, Beschreibung der Attribute siehe unten
	
	Ende einer Feld-Definition (field0)

15.3.1 Attribute der Feld-Definition

name

Alphanumerisch.

- *Bei strukturierten Messdatenkanälen ("histdata0" - "histdata9"):*

  Interner Name des Kanals. Bei rapidM2M sind 1000 Kanäle verfügbar. Es können also die Bezeichnungen ch0 bis ch999 verwendet werden.

- *Bei Konfigurationsspeicherblöcken ("config0" - "config9"):*

  Interner Name des Konfigurationsparameters. Bei rapidM2M sind 1000 Konfigurationsparameter verfügbar. Es können also die Bezeichnungen field0 bis field999 verwendet werden.

alias

Alphanumerisch. Alias für den internen Namen des Kanals der von der REST-API verwendet wird.

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.

type

Datentyp des Feldes. Der Datentyp bestimmt gleichzeitig die Anzahl der aus dem Datenstream herauszuholenden Bytes. Die einzige Ausnahme ist der Typ string (bzw. char), dessen Länge durch das Attribut max bestimmt wird.
Verfügbare Datentypen: s8, s16, s32, s64, u8, u16, u32, f32, f64, string
Alternative Bezeichnungen (standardisierte Werte in Klammer): bint(s8), wint(s16), dint(s32), qint(s64), byte(u8), word(u16), dword(u32), float32(f32), float(f32), float64(f64), char(string)

param1

Alphanumerisch. Momentan einzig erlaubter Wert: nocodepage
Dieses Attribut muss für Felder gesetzt werden, die einen ANSI-kodierten String enthalten.

byteofs

Numerisch, ganzzahlig positiv. Offset des ersten für das Feld zu verwendende Byte, 0-basierend.

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

Formatanweisung	***Erklärung***
0=off;1=on	*Es wird eine ComboBox erstellt in der für jeden Eintrag anstelle des Wertes der Text nach dem "=" angezeigt wird. Die Einträge sind durch ";" zu trennen.*

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.

15.4 Beispiel

`<table>`	Beginn des Table-Tags
`  name  =histdata0`	Name der zu beschreibenden Datentabelle
`  <field>`	Beginn einer Feld-Definition (ch0)
`    name    =ch0`	interner Name des Kanals
`    title   =Verzögerung`	Titel des Kanals
`    units   =sec`	Einheiten
`    type    =u16`	Datentyp, mit u16 werden also 2 Bytes verwendet
`    byteofs =0`	Die ersten beiden Bytes des Datenstreams werden genommen.
`    vofs    =10`	Zum extrahierten Wert wird 10 addiert.
`    min     =10`	Der Minimalwert für die weitere Verwendung ist 10.
`    max     =2000`	Der Maximalwert für die weitere Verwendung ist 2000.
`    chmode  =3`	Es handelt sich um einen Intervallzähler.
`    index   =1`	Sortier-Nummer
`  </field>`	Ende des Field-Tags
`  <field>`	
`    name    =ch1`	interner Name des Kanals
`    title   =Höhe`	Titel des Kanals
`    units   =cm`	Einheiten
`    type    =f32`	Datentyp, Fließkomma 32 bit
`    byteofs =2`	Es werden die Bytes 2 bis 5 genommen.
`    decpl   =2`	2 Nachkommastellen
`    vofs    =0`	Kein Offset
`    vscale  =0.01`	Der extrahierte Wert wir mit 0.01 multipliziert.
`    min     =0`	Der Minimalwert für die weitere Verwendung ist 0.
`    max     =2000`	Der Maximalwert für die weitere Verwendung ist 2000.
`    chmode  =6`	Es handlet sich um einen Analogkanal.
`    index   =0`	Sortier-Nummer
`  </field>`	
`  <field>`	
`    name    =ch2`	interner Name des Kanals
`    title   =Pumpe`	Titel des Kanals
`    type    =u8`	Datentyp, ganzzahlig 8 bit
`    byteofs =6`	Es wird das Byte 6 genommen.
`    bitmask =01`	Es wird das niedrigstwertige Bit (LSB) aus dem Byte heraus maskiert.
`    min     =0`	Der Minimalwert für die weitere Verwendung ist 0.
`    max     =1`	Der Maximalwert für die weitere Verwendung ist 1.
`    chmode  =1`	Es handelt sich um einen Digitalkanal.
`  </field>`	
`  <field>`	
`    name    =ch3`	interner Name des Kanals
`    title   =Info`	Titel des Kanals
`    type    =string`	Datentyp, String, Länge wird im Feld max angegeben.
`    byteofs =10`	Der Text startet beim Byte 10.
`    max     =50`	Der Text ist maximal 50 Zeichen lang.
`    index   =10`	Sortier-Nummer
`  </field>`	
`</table>`	

Bei ch2 fehlt die Angabe von index, daher erhält dieser Kanal automatsich den Index 2. Die Sortierung der Kanäle ist also ch1, ch0, ch2, ch3.

15.5 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 16 API

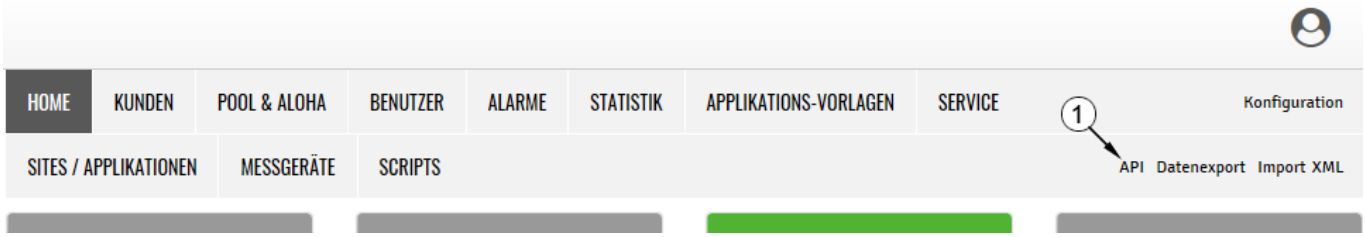
Wichtiger Hinweis: *Für die Verwendung der API (Application Programming Interface) sind die entsprechenden Lizenzen am myDatanet-Server erforderlich. Für nähere Informationen wenden Sie sich an Ihren zuständigen Vertriebspartner.*

16.1 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.

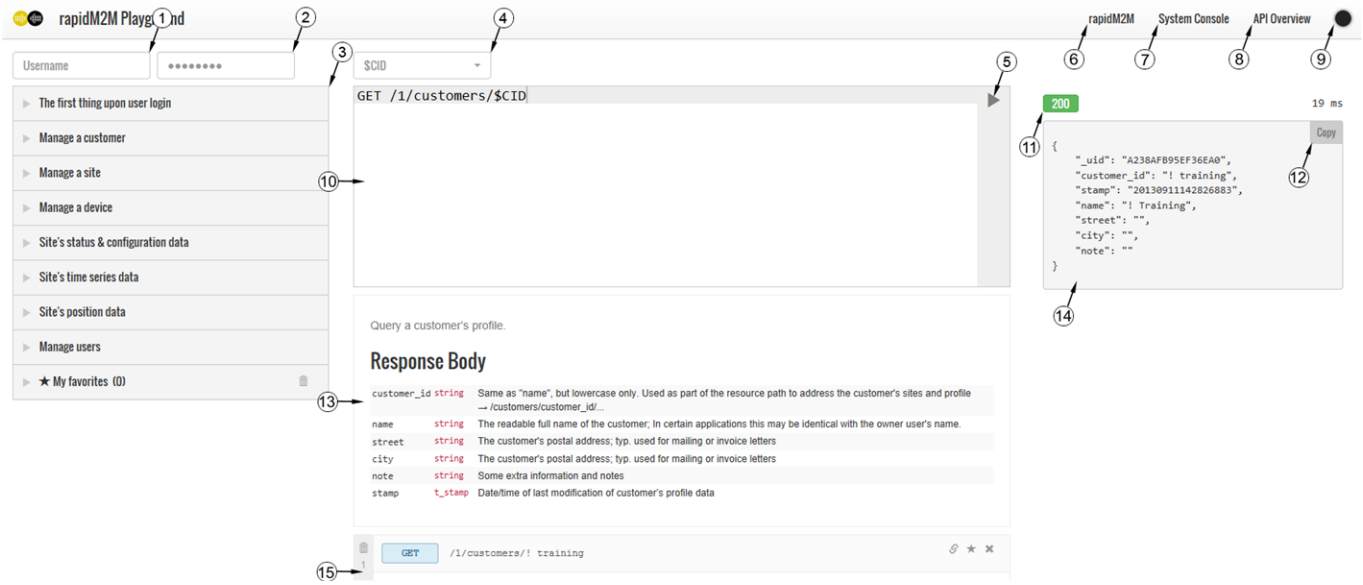
16.2 rapidM2M Playground

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

16.2.1 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 Messstelle eingeblendet, die die entsprechende Wildcard ("$CID"...Kunde , "$UID"...Benutzer, "$SID"...Messstelle) 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 den login-Dialog des mit dem rapidM2M Playground verknüpften myDatanet-Servers
8	öffnet die Kurzanleitung für die API
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 17 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.

17.1 Allgemeine Wartung

- Überprüfen Sie das EC-CAN-00x regelmäßig auf mechanische Beschädigungen.
- Überprüfen Sie regelmäßig alle Anschlüsse auf Dichtheit und Korrosion.
- Überprüfen Sie regelmäßig alle Kabel auf mechanische Beschädigungen.
- Reinigen Sie das EC-CAN-00x mit einem weichen, feuchten Tuch. Verwenden Sie ein mildes Reinigungsmittel, falls nötig.

17.2 Sicherungswechsel

GEFAHR:
Feuergefahr. Eine falsche Sicherung kann Verletzungen, Schäden oder Immissionen verursachen.

Im Inneren des EC-CAN-00x befindet sich keine Sicherung. Die Montagevorschriften (siehe "Anschluss der Sensoren, der Aktoren und der Versorgung" auf Seite 46) schreiben eine T2A Vorsicherung in der Versorgungsleitung vor. Unterbrechen Sie vor dem Sicherungswechsel auf jeden Fall die Stromzufuhr.

Kapitel 18 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.
3. Entfernen Sie das Akku- bzw. Batteriepack und entsorgen Sie dieses separat.



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. Das Gerät enthält eine Batterie bzw. einen Akku (Lithium), welcher separat zu entsorgen ist.

Kapitel 19 Fehlersuche und Behebung

19.1 Allgemeine Probleme

Problem	Ursache/Lösung
Gerät zeigt keine Reaktion.	• Kabelverbindungen überprüfen (siehe "Anschluss der Sensoren, der Aktoren und der Versorgung" auf Seite 46) • Überprüfen Sie die T2A Vorsicherung (siehe "Blockschaltbild" auf Seite 23).
Kommunikationsprobleme	• Laden Sie das Gerätelog mit Hilfe des rapidM2M Toolset vom EC-CAN-00x (siehe "Karteireiter "Log"" auf Seite 75). Eine Auflistung aller möglichen Fehlercodes finden Sie im Kapitel "Log-Einträge und Fehlercodes" auf Seite 200.
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 myDatanet Connector wurde nicht korrekt konfiguriert (siehe "myDatanet Connector " auf Seite 185). • Die Zuweisung von Gerät und Messstelle ist nicht korrekt (siehe "Site" auf Seite 56).
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 48). • Prüfen Sie, ob die kontinuierliche Buslast unter der zulässigen Grenze von 40% liegt. • Der myDatanet Connector wurde nicht korrekt konfiguriert (siehe "myDatanet Connector " auf Seite 185).
Das Pawn Script wird nicht korrekt ausgeführt.	• Prüfen Sie, ob bei der Konfiguration der Steuerung der korrekte Script-Typ ausgewählt wurde (siehe "Steuerung" auf Seite 57). • Laden Sie das Gerätelog mit Hilfe des rapidM2M Toolset vom EC-CAN-00x (siehe "Karteireiter "Log"" auf Seite 75). Eine Auflistung aller möglichen Fehlercodes des Pawn Scripts finden Sie im Kapitel "Pawn Script Fehlercodes" auf Seite 160. • Durch einen Kontextwechsel (Zuweisung einer anderen Messstelle / Applikation) wurde das bisherige Script durch jenes der neu zugewiesenen Messstelle / Applikation ersetzt. • Durch die Zuweisung einer neuen oder anderen Messstelle / Applikation wurde das per USB eingespielte Script durch jenes der neu zugewiesenen Messstelle / Applikation ersetzt.

19.2 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 213).
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. Dies kann entweder durch einen Akku-/Batteriewechsel oder dadurch, dass die Laderegelung den Akku ausreichend wieder aufgeladen hat, erfolgen. 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 213).

Log-Eintrag		Parameter		Beschreibung
Code	Klartext	Code	Klartext	
1038	UV MODEM LOCKOUT	---	---	Das Gerät deaktiviert aufgrund einer zu niedrigen Akku-/Batteriespannung das Modem. Das Herstellen einer Verbindung ist nicht mehr möglich.
1039	UV MODEM RECOVER	---	---	Die Akku-/Batteriespannung reicht wieder aus, um eine stabile Verbindung herzustellen. Dies kann entweder durch einen Akku-/Batteriewechsel oder dadurch, dass die Laderegelung den Akku ausreichend wieder aufgeladen hat, erfolgen.
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 213).
1192	FUTURE TIMESTAMP	##	---	interner Fehler Sollte dieser Fehler mehrmals im Gerätelog enthalten sein, kontaktieren Sie den Hersteller (siehe "Kontaktinformationen" auf Seite 213).
1200	MODEM ERROR			Modemfehler (siehe "Modemfehler" auf Seite 203)
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. Dieser Eintrag ist immer doppelt im Gerätelog enthalten. Beim ersten Eintrag gibt der Parameter den MCC (Mobile Country Code) und beim zweiten Eintrag den MNC (Mobile Network Code) des gewählten GSM-Netzwerks an.
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.
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 213).

Log-Eintrag		Parameter		Beschreibung
Code	**Klartext**	**Code**	**Klartext**	
1281	ZLIB STREAMPROCESS ERR	##	---	interner Fehler Sollte dieser Fehler mehrmals im Gerätelog enthalten sein, kontaktieren Sie den Hersteller (siehe "Kontaktinformationen" auf Seite 213).
1282	ZLIB STREAMFINISH ERR	##	---	interner Fehler Sollte dieser Fehler mehrmals im Gerätelog enthalten sein, kontaktieren Sie den Hersteller (siehe "Kontaktinformationen" auf Seite 213).
1300	USB CONNECTED	---	---	USB-Verbindung zu einem PC hergestellt
1310	USB DISCONNECTED	---	---	USB-Verbindung wurde getrennt
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)
2000 - 2199	MODULE ERR	##	---	Bereich für die kundenspezifischen kritischen Fehlercodes, die mittels der Pawn Script Funktion "rM2M_WriteLog()" in das Gerätelog geschrieben werden können
2200 - 2399	MODULE WARNING	##	---	Bereich für die kundenspezifischen unkritischen Fehlercodes, die mittels der Pawn Script Funktion "rM2M_WriteLog()" in das Gerätelog geschrieben werden können
2400 - 2599	MODULE INFO	##	---	Bereich für die kundenspezifischen Informationen über den aktuellen Betriebszustand, die mittels der Pawn Script Funktion "rM2M_WriteLog()" in das Gerätelog geschrieben werden können
2600 - 2799	MODULE DEBUG	##	---	Bereich für die kundenspezifischen Debug-Informationen, die mittels der Pawn Script Funktion "rM2M_WriteLog()" in das Gerätelog geschrieben werden können
3000 - 3099	SCRIPT ERROR	##	--	Fehlercodes der Scriptausführung (siehe "Pawn Script Fehlercodes" auf Seite 160).

19.2.1 Modemfehler

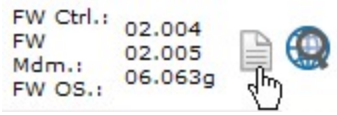
Log-Eintrag		Parameter		Beschreibung
Code	**Klartext**	**Code**	**Klartext**	
GPRS-Fehler				
1200	BEARER GPRS FAILED	-988	---	GPRS Setup-Fehler • Versuchen Sie die Antennenposition zu verbessern. • Überprüfen Sie, ob sich das Gerät im Versorgungsbereich befindet (www.microtronics.com/footprint).
1200	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

19.3 Auswerten des Gerätelogs

19.3.1 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).

19.3.2 Auswerten des Gerätelogs mittels rapidM2M Toolset

> ***Wichtiger Hinweis:*** *Bei der USB-Schnittstelle handelt es sich um eine Serviceschnittstelle, die nur nach Öffnen des Gehäuses zugänglich ist. Geräteschäden wie z.B. durch Feuchtigkeitseintritt, die auf das Öffnen des Gehäuses zurückzuführen sind, entfallen aus der Haftung des Herstellers.*

Mit Hilfe des rapidM2M Toolset können die Log-Einträge direkt über die USB-Schnittstelle aus dem EC-CAN-00x ausgelesen werden. Es können alle im Gerät gespeicherten Log-Einträge ausgelesen werden, auch jene, die noch nicht zum myDatanet-Server übertragen wurden.

Eine genauere Beschreibung zur Auswertung des Gerätelogs mittels rapidM2M Toolset finden Sie im Handbuch zum rapidM2M Toolset ("Benutzerhandbuch für rapidM2M Toolset" 300714).

Kapitel 20 Dokumentenhistorie

Rev.	Datum	Änderungen
01	15.09.2017	**Erste Version**
02	16.10.2017	**Kapitel "Blockschaltbild" auf Seite 23** *Die durch den Anwender zu implementierende Vorsicherung wurde in die Darstellung übernommen.* **Kapitel "Anschluss der Sensoren, der Aktoren und der Versorgung" auf Seite 46** *Hinweis hinzugefügt, dass die Versorgungsleitung durch den Anwender mittels Vorsicherung abgesichert werden muss* **Kapitel "Sicherungswechsel" auf Seite 195** *Kapitel hinzugefügt* **Kapitel "Fehlersuche und Behebung" auf Seite 199** *Hinweis hinzugefügt, dass wenn das Gerät keine Reaktion zeigt, auch die Vorsicherung überprüft werden sollte*
03	10.11.2017	**Kapitel "Technische Daten" auf Seite 17** *Bei der Variante EC-CAN-001 werden die Frequenzbänder B4 und B6 nicht mehr unterstützt.* *Angabe der Frequenzbänder für die Variante EC-CAN-004 hinzugefügt* **Kapitel "Blockschaltbild" auf Seite 23** *Bei der durch den Anwender zu implementierende Vorsicherung muss es sich um eine träge Sicherung (T2A) handel.* **Kapitel "Gerätekennzeichnung" auf Seite 24** *Typenschild der Variante EC-CAN-001 aktualisiert* *Typenschild der Variante EC-CAN-004 hinzugefügt* **Kapitel "Lieferumfang" auf Seite 41** *Lieferumfang für die Variante EC-CAN-004 hinzugefügt*
04	14.11.2017	**Kapitel "Technische Daten" auf Seite 17** *Angabe der Frequenzbänder für die Varianten EC-CAN-002 und EC-CAN-003 hinzugefügt* **Kapitel "Lieferumfang" auf Seite 41** *Lieferumfänge für die Varianten EC-CAN-002 und EC-CAN-003 hinzugefügt*
05	07.12.2017	**Kapitel "FCC Statement" auf Seite 11** *Kapitel hinzugefügt* **Kapitel "Technical Compliance Statement FCC" auf Seite 13** *Kapitel hinzugefügt* **Kapitel "Konformitätserklärung" auf Seite 15** *Konformitätserklärung aktualisiert* **Kapitel "Gerätekennzeichnung" auf Seite 24** *Typenschild der Varianten EC-CAN-001 und EC-CAN-003 aktualisiert* *Typenschild der Varianten EC-CAN-004 und EC-CAN-002 hinzugefügt*

Rev.	Datum	Änderungen
06 (1/3)	14.06.2019 (1/3)	**Kapitel "Bestimmungsgemäße Verwendung" auf Seite 23** *Hinweis zum Haftungsausschluss des Herstellers beim Verlust von Daten aktualisiert* **Kapitel "Gerätekennzeichnung" auf Seite 24** *Typenschild der Varianten EC-CAN-001 aktualisiert* *Verpackungsetikett der Variante EC-CAN-001 hinzugefügt* **Kapitel "Pflichten des Betreibers" auf Seite 26** *Verweis auf die Richtlinie für Mindestvorschriften für Sicherheit und Gesundheitsschutz bei Benutzung von Arbeitsmitteln durch Arbeitnehmer bei der Arbeit aktualisiert* **Kapitel "Anforderungen an das Personal" auf Seite 27** *Kapitel hinzugefügt* **Kapitel "Funktionsweise des internen Datenspeichers " auf Seite 31** *Erklärung hinzugefügt, dass der Timestamp bereits in den 10Byte Overhead pro Datensatz inkludiert ist* **Kapitel "Speicherorganisation" auf Seite 33** *Erklärung des optional im RAM initialisierbaren Registrierungsspeicherblocks "RM2M_REG_ APP_STATE" hinzugefügt* **Kapitel "File Transfer" auf Seite 38** *Angabe der maximale Anzahl an Dateien, die für den File Transfer registriert werden können, hinzugefügt* **Kapitel "Montage des EC-CAN-00x " auf Seite 43** *Hinweis hinzugefügt, dass der Druckausgleich vor Verschmutzung geschützt werden muss* **Kapitel "Technische Details zur Systemzeit" auf Seite 50** *Angaben zur Lebensdauer der RTC-Pufferbatterie hinzugefügt* **Kapitel "Alarmierung" auf Seite 58** *Erklärung der Eingabefelder für die Bezeichnung der benutzerdefinierten Alarme hinzugefügt* **Kapitel "Grundeinstellungen" auf Seite 59** *Erklärung des Parameters zur Auswahl, welche Änderungen an den Konfigurationen protokolliert werden sollen, hinzugefügt* **Kapitel "Messgerät" auf Seite 60** *Erklärung des Parameters "Wakeup SMS Anzahl" hinzugefügt* *Erklärung des Feldes, das die im Gerät verbaute Hardwareplattform angibt, hinzugefügt* **Kapitel "Compiler-Optionen" auf Seite 85** *Kapitel hinzugefügt* **Kapitel "Timer, Datum & Zeit" auf Seite 87** *Erklärung des Rückgabewertes und des Parameters "time" der Funktion "rM2M_ TimerAddExt()" in Bezug auf zyklische 0ms-Timer ergänzt*

Rev.	Datum	Änderungen
06 (2/3)	14.06.2019 (2/3)	**Kapitel "Uplink" auf Seite 92** *Erklärung dahingehend ergänzt, dass die Verbindung im online-Modus alle 7 Tage zwecks Überprüfung der Serverzuordung unterbrochen wird* *Erklärung des Parameters "errorcode" der Funktion "rM2M_TxGetStatus() hinzugefügt* *Erklärung der Konstanten für die Verbindungsfehlercodes hinzugefügt* *Erklärung des Flags "RM2M_TX_SUPPRESS_POSUPDATE" und "RM2M_TX_POSUPDATE_ONLY" der Funktion "rM2M_ TxStart()" hinzugefügt* *Erklärung des Flags "RM2M_TX_POSUPDATE_ACTIVE" der Funktion "rM2M_TxGetStatus()" hinzugefügt* *Erklärung der Konstanten für die Verbindungsfehlercodes um jene für Ethernet Fehler ergänzt.* *Hinweis hinzugefügt, dass die Funktion "rM2M_GetRSSI()" anstelle der Funktion "rM2M_GSMGetRSSI()" verwendet werden sollte* *Erklärung der Funktion "rM2M_GetRSSI()" hinzugefügt* **Kapitel "Encoding" auf Seite 105** *Hinweise hinzugefügt, dass die Funktion "rM2M_Pack()" anstelle der Funktionen "rM2M_SetPacked()" und "rM2M_GetPacked()" verwendet werden sollte* **Kapitel "GPIO, IRQ" auf Seite 111** *Durch Setzen des Parameters "dir" der Funktion "rM2M_GpioDir()" auf "RM2M_GPIO_DISABLED" kann ein GPIO wieder deaktiviert werden* **Kapitel "SPI, I2C, UART" auf Seite 114** *Erklärung der Funktion "rM2M_UartSetbuf()" hinzugefügt* **Kapitel "Registry" auf Seite 121** *Erklärung des Rückgabewertes der Funktion "rM2M_RegInit()" ergänzt* **Kapitel "Position" auf Seite 127** *Erklärung des Arrays mit symbolischen Indizes "TNMEA_GGA" hinzugefügt* *Erklärung der Konstanten für die Liste der unterstützten GNSS Geräte-ID's hinzugefügt* *Erklärung der Konstanten für die Liste der unterstützten NMEA-Datensätze hinzugefügt* *Erklärung der Konstanten für den NMEA Qualitätsindikator hinzugefügt* *Erklärung der Funktion "rM2M_DecodeNMEA()" hinzugefügt* *Datentyp des Parameters "Sentence" der Funktion "rM2M_SetPosNMEA()" von Cell-Array auf Byte-Array geändert* *Parameter "Qual" und "SatUsed" zur Funktion "rM2M_GetPos()" hinzugefügt* *Erklärung der Arrays mit symbolischen Indizes "TrM2M_PosUpdateGSM", "TrM2M_PosUpdateUMTS" und "TrM2M_PosUpdateWiFi" hinzugefügt* *Erklärung der Konstanten für Liste der unterstützten Typen von Zell/Netzwerk-Informationseinträgen hinzugefügt* *Erklärung der Funktion "rM2M_EnumPosUpdate()" hinzugefügt.* *Hinweis hinzugefügt, dass mittels der Funktion "rM2M_EnumPosUpdate()" spezifischere Informationen zu UMTS- bzw. LTE-Zellen erhalten werden können als mit der Funktion "rM2M_GetGSMPos()".* **Kapitel "CRC & Hash" auf Seite 145** *Kapitel hinzugefügt*

Rev.	Datum	Änderungen
06 (3/3)	14.06.2019 (3/3)	**Kapitel "Verschiedene Funktionen" auf Seite 146** *Initialwert für die Berechnung bei der Funktion "CRC32()" von "0xFFFFFFFF" auf "0" geändert* *Erklärung des Arrays mit symbolischen Indizes "TRTM_Data" hinzugefügt* *Erklärung der Funktionen "loadmodule()", "rtm_start()", "rtm_stop()" und "rand()" hinzugefügt* **Kapitel "Console" auf Seite 153** *Erklärung der Funktion "setbuf()" hinzugefügt* **Kapitel "SMS" auf Seite 154** *Hinweis hinzugefügt, dass keine SMS verarbeitet werden können, wenn sich das Gerät im "online"-Modus befindet* **Kapitel "File Transfer" auf Seite 155** *Struktur, welche die Eigenschaften eines Dateieintrags enthält um das Feld "stamp256" erweitert* *Hinweis hinzugefügt, dass die Funktion "FT_SetPropsExt()" anstelle der Funktion "FT_SetProps()" verwendet werden sollte* *Erklärung der Funktion "FT_SetPropsExt()" hinzugefügt* **Kapitel "Spezialwerte der Datentypen" auf Seite 192** *Erklärung der NaN-Codierung für float32 und float64 aktualisiert* **Kapitel "Log-Einträge und Fehlercodes" auf Seite 200** *Erklärung des Fehlercode "FUTURE TIMESTAMP" hinzugefügt*
07	16.08.2021	**Kapitel "Technical Compliance Statement FCC" auf Seite 13** *Technische Konformitätserklärung FCC aktualisiert* **Kapitel "Konformitätserklärung" auf Seite 15** *Konformitätserklärung aktualisiert* **Kapitel "Technische Daten" auf Seite 17** *Angabe der Frequenzbänder für die Variante EC-CAN-002 aktualisiert* *Varianten EC-CAN-003 und EC-CAN-004 entfernt* **Kapitel "Gerätekennzeichnung" auf Seite 24** *Typenschild der Variante EC-CAN-002 aktualisiert* *Varianten EC-CAN-003 und EC-CAN-004 entfernt* **Kapitel "Lieferumfang" auf Seite 41** *Varianten EC-CAN-003 und EC-CAN-004 entfernt*

Kapitel 21 Glossar

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).

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.

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, EN ISO 50001:2011 für myDatanet | TÜV SÜD: ATEX Richtlinie 2014/34/EU

© Microtronics Engineering GmbH. All rights reserved. Photos: Microtronics

Microtronics Engineering GmbH | www.microtronics.com
Hauptstrasse 7 | 3244 Ruprechtshofen | Austria | +43 2756 77180 | office@microtronics.com