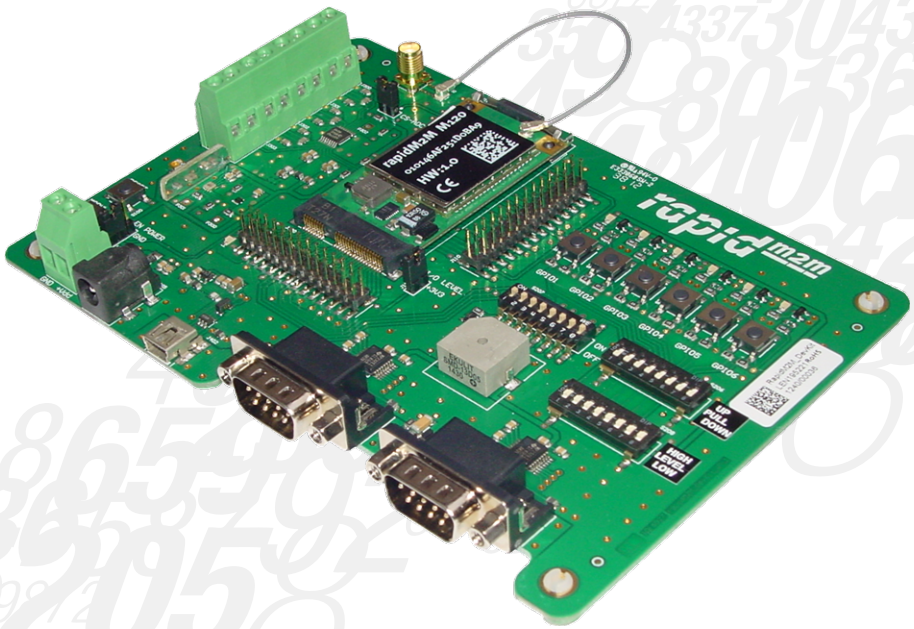
# Betriebsanleitung für das rapidM2M Developer Paket M12x




Rev. 03 gültig ab:

- Firmware Version: ---
- Modem Version: 00v014
- Server Version: 41.1
- Hardware Version: 1.2

CE

Microtronics Engineering GmbH | www.microtronics.at
Hauptstrasse 7 | 3244 Ruprechtshofen | Austria
Tel +43 2756 77180 23 | Fax +43 2756 77180 33 | support@microtronics.at

6872
6872
6872

# Kapitel 1 Inhaltsverzeichnis

**Betriebsanleitung für das rapidM2M Developer Paket M12x** ... 1

**Kapitel 1 Inhaltsverzeichnis** ... 3

**Kapitel 2 Technische Daten** ... 15

**Kapitel 3 Allgemeine Angaben** ... 19

3.1 Allgemeine Produktinformationen ... 19

3.2 Lieferumfang ... 20

3.3 Abmessungen ... 20

3.4 Gerätekennzeichnung ... 21

3.5 Aufbewahrung des Produkts ... 21

**Kapitel 4 Tutorial** ... 23

4.1 Beschreibung der Funktion des Scripts "app.p" ... 23

4.2 Voraussetzungen ... 24

4.3 Schritt-für-Schritt-Anleitung ... 24

4.4 Erklärung der rapidM2M spezifischen Codeblöcke des Scripts "app.p" ... 45

4.4.1 Initialisieren der Sensoren ... 45

4.4.2 Initialisieren der Timer ... 46

4.4.3 Callback für die Konfigurationsbehandlung festlegen ... 46

4.4.4 Verbindungsart setzen ... 46

4.4.5 Messdaten speichern ... 47

4.4.6 Überwachung der Taste S201 zum Auslösen einer sofortigen Aufzeichnung ... 47

4.4.7 Lesen der Sensorwerte ... 48

4.4.8 Lesen der Konfiguration ... 49

4.5 Erklärung der in der Datei "app_connector.txt" enthaltenen Connector-Definition ... 50

4.5.1 Split-tag ... 50

4.5.2 Messdatenkanal 0 ... 51

4.5.3 Konfigurationsspeicherblock 0 ... 52

**Kapitel 5 Funktionsprinzip** ... 55

5.1 Empfohlene Vorgehensweise ... 58

5.1.1 Entwicklung einer M2M / IoT Anwendung ... 58
5.2 Automatische Auswahl des GSM-Netzes ... 59
5.2.1 Banlist ... 60
5.3 Vorgehensweise bei Verbindungsabbrüchen ... 60
5.3.1 Verbindungsabbruch im "online"-Modus ... 61
5.3.2 Verbindungsabbruch während eines Scriptdownloads ... 62
5.4 Errorhandling ... 62
5.5 Registrierungsspeicherblöcke ... 63
5.5.1 REG_SYS_FLASH ... 63
5.5.2 REG_APP_OTP ... 64
5.6 File Transfer ... 64
5.7 Bedeutung des SIM-Status ... 65
**Kapitel 6 rapidM2M Developer Board ... 67**
6.1 Übersicht ... 67
6.1.1 Blockschaltbild ... 70
6.1.2 RS232-Schnittstellen ... 71
6.1.3 Simulationsschaltungen ... 72
6.1.3.1 Ausgabe von digitalen Signalen ... 73
6.1.3.2 Eingabe von digitalen Signalen ... 74
6.1.4 Jumper zur Versorgung der Level Shifter (J111) ... 75
6.1.5 Jumper zur Auswahl der USB-Versorgung für das Modul (J113) ... 75
6.1.6 Stiftleiste auf der die Pins mit den ungeraden Nummern des M120 Moduls aufgelegt sind (J106 und J109) ... 76
6.1.7 Stiftleiste auf der die Pins mit den geraden Nummern des M120 Moduls aufgelegt sind (J107 und J110) ... 78
6.1.8 USB-Schnittstelle ... 80
6.1.9 Status-LED (LED100) ... 80
6.1.10 Versorgung (J105 bzw. J108) ... 80
6.1.10.1 Klemmleiste (J108) ... 80
6.1.10.2 DC-Hohlstecker (J105) ... 81

6.1.11 Jumper zum Aktivieren der Energieversorgung des M120 Moduls (J104) ... 81
6.1.12 Taster zum Einschalten des Wireless GSM-Controllers am rapidM2M Modul M120 (S101) ... 81
6.1.13 Jumper zum Einschalten des Wireless GSM-Controllers am rapidM2M Modul M120 (J37) ... 81
6.1.14 Taste zum Reseten des Wireless GSM-Controllers am M120 Modul (S100) ... 82
6.1.15 Sensoren ... 82
6.1.15.1 ADC AD7792 (U302) ... 83
6.1.15.2 Temperatursensor TMP112 (U300) ... 84
6.1.15.3 Umgebungslicht und IR-Näherungssensor MAX44000 (U301) ... 85
6.1.16 Aufnahme des M120 Moduls ... 85
6.2 Schaltplan ... 86
6.2.1 Übersicht ... 86
6.2.2 Connector ... 87
6.2.3 USB RS232 ... 88
6.2.4 Sensoren ... 89
6.2.5 GPIO ... 90
6.3 Layout ... 91
6.4 Elektrische Installation ... 92
6.4.1 Hinweise zur Vermeidung elektrostatischer Entladungen (ESE) ... 92
6.4.2 Einsetzen des M120 Moduls ... 93
6.4.3 Anschluss der Sensoren, der Aktoren und der Versorgung ... 94
6.4.4 Anschluss der GSM-Antenne ... 96
**Kapitel 7 rapidM2M Modul M120 ... 99**
7.1 Technische Daten ... 99
7.2 Übersicht ... 101
7.2.1 Blockschaltbild ... 102
7.2.2 52pin Steckerleiste (X2) ... 102
7.2.3 Speicherorganisation ... 105
7.2.4 Gerätekennzeichnung ... 106

7.3 Abmessungen ... 106
7.3.1 Abmessungen der 52pin Steckerleiste (X2) ... 107
7.4 Technische Details ... 107
7.4.1 Technische Details zur Energieversorgung ... 107
7.4.1.1 Versorgung über VIN ... 108
7.4.1.2 Versorgung über VAUX ... 108
7.4.2 ENPower-Signal ... 108
7.4.3 ENCPU-Signal ... 108
7.4.4 Reset-Signal ... 109
7.4.5 Status-LED ... 109
7.4.6 Technische Details zu den Level Shiftern ... 110
7.4.7 Serielle Schnittstellen ... 110
7.4.7.1 UART ... 110
7.4.7.1.1 Pin-Beschreibung ... 110
7.4.7.2 SPI ... 111
7.4.7.2.1 Pin-Beschreibung ... 111
7.4.7.2.2 SPI Waveforms ... 112
7.4.7.2.3 Applikation ... 112
7.4.7.3 I2C ... 112
7.4.7.3.1 Pin-Beschreibung ... 112
7.4.7.3.2 I2C Waveforms ... 113
7.4.7.3.3 Applikation ... 113
7.4.8 GPIO ... 114
7.4.9 Interrupt Eingänge ... 114
7.5 Typische Anwendung ... 114
7.5.1 Versorgung über VAUX ... 114
7.5.2 Versorgung über VIN ... 115
7.5.3 Gültig für beide Anwendungsfälle ... 116
**Kapitel 8 Cloud-Server ... 117**
8.1 Neu registrieren und anmelden ... 117

8.1.1 Neuen Account anlegen ... 118
8.1.2 Anmelden ... 118
8.2 Übersicht ... 119
8.2.1 Erklärung der Symbole ... 119
8.3 Bereich "Messstellen" ... 120
8.3.1 Auswertungen ... 121
8.3.1.1 Grundkonfiguration ... 122
8.3.2 Konten ... 122
8.3.3 Messstellen ... 123
8.3.3.1 Anlegen einer neuen Messstelle ... 125
8.3.3.2 Messstelle Position ... 126
8.3.3.3 Dateiliste ... 127
8.3.3.3.1 Bearbeiten der Datei ... 128
8.3.3.4 Messstelle auf Karte positionieren ... 129
8.4 Bereich "Benutzer" ... 130
8.4.1 Benutzer anlegen ... 131
8.5 Bereich "Alarme" ... 131
8.5.1 Unterbereich "Alarme" ... 132
8.5.2 Unterbereich "Quittierte Alarme" ... 133
8.5.3 Unterbereich "Alarm Archiv" ... 134
8.5.4 Unterbereich "Alarmrufpläne" ... 135
8.5.4.1 Alarmrufplan und Kommentar ... 135
8.5.4.2 Einrichten einer telefonischen Alarmierung ... 135
8.5.5 Benachrichtigung ... 136
8.6 Bereich "Statistik" ... 136
8.6.1 Unterbereich "SMS/Anrufe" ... 137
8.6.2 Unterbereich "SIM" ... 138
8.7 Bereich "Meine Daten" ... 139
8.8 Applikations-Vorlagen des Cloud-Servers ... 139
**Kapitel 9 rapidM2M Toolset ... 141**

9.1 Allgemein ... 141
9.2 Voraussetzungen ... 141
9.3 Funktionsprinzip ... 142
9.4 Installation ... 143
9.5 Übersicht ... 146
9.6 Kateireiter "Script" ... 148
9.7 Karteireiter "Config" ... 150
9.8 Karteireiter "Console" ... 151
9.9 Karteireiter "COM Data" ... 151
9.10 Karteireiter "Server" ... 153
9.11 Karteireiter "Log" ... 154
9.12 Karteireiter "FW" ... 155
9.13 Hot-Keys ... 156
**Kapitel 10 XML-Schnittstelle ... 157**
10.1 Allgemein ... 157
10.2 Aufbau der Datenbank ... 157
10.2.1 Allgemein gültige Tabellenbezeichner ... 160
10.2.1.1 rm2mid ... 160
10.2.1.2 geraetid ... 164
10.2.1.3 bookingline (Read Only) ... 165
10.2.2 Geräteabhängige Tabellenbezeichner ... 166
10.2.2.1 histdata0 - histdata9 ... 166
10.2.2.2 rm2mlog ... 166
10.2.2.3 rm2mstate ... 166
10.2.2.4 rm2mlocals ... 167
10.2.2.5 rm2mpos ... 167
10.2.2.6 rm2m_ctrlscript ... 168
10.2.2.7 reg_sys_otp ... 169
10.2.2.8 reg_sys_flash ... 169
10.2.2.9 reg_app_otp ... 169

10.2.2.10 reg_app_flash ... 170
10.2.2.11 configo - config9 ... 170
10.2.2.12 rm2mraw ... 170
10.3 XML-Export ... 170
10.3.1 Aufbau des HTTP-GET-Kommandos ... 170
10.3.1.1 Syntax der xPath Query ... 171
10.3.1.1.1 Auflistung der Felder einer Tabelle ... 171
10.3.1.1.2 Liste der Messstellen-Typen ... 172
10.3.1.1.3 Filterbedingungen ... 172
10.3.1.1.3.1 Beispiel ... 173
10.3.1.1.4 Felder ... 174
10.3.1.1.4.1 Beispiel ... 174
10.3.1.1.5 Datenabfrage mit Index-Angaben ... 174
10.3.1.1.5.1 Beispiele ... 175
10.3.1.2 Datensatzverdichtung ... 176
10.3.1.3 Update von Datenfeldern ... 178
10.3.1.3.1 Beispiele ... 178
10.3.1.4 Datenabfrage mittels Streamingtechnologie ... 179
10.3.1.4.1 Events ... 180
10.3.1.5 Ausgabeformate des HTTP-GET-Kommandos ... 181
10.3.1.5.1 Format des XML-Ergebnisses ... 181
10.3.1.5.1.1 Beispiel ... 182
10.3.1.5.2 Format des JSON-Ergebnisses ... 182
10.3.1.5.2.1 Beispiel ... 183
10.4 XML-Import ... 183
10.4.1 Übertragung der Daten an den Cloud-Server ... 183
10.4.1.1 Aufbau des HTTP-POST-Kommandos ... 185
10.4.1.2 Aufbau der XML-Datei ... 186
10.4.1.2.1 Ändern von vorhandenen Daten ... 186
10.4.1.2.2 Import von Daten ... 186

**Kapitel 11 Inbetriebnahme** ..... **189**
11.1 Inbetriebnahme des Systems ..... 189
11.2 Kommunikation mit dem Gerät testen ..... 190
**Kapitel 12 Benutzerschnittstelle am Cloud-Server** ..... **193**
12.1 Messstellenkonfiguration ..... 193
12.1.1 Messstelle ..... 193
12.1.2 Kommentar ..... 193
12.1.3 Steuerung ..... 193
12.1.4 Konfiguration 0 - Konfiguration 1 ..... 194
12.1.5 Alarmierung ..... 194
12.1.6 Grundeinstellung ..... 194
12.1.7 FTP-Export Einstellungen ..... 195
12.2 Gerätekonfiguration ..... 195
12.2.1 Kommentar ..... 195
12.2.2 Messgerät ..... 195
**Kapitel 13 Pawn Script** ..... **197**
13.1 Allgemein ..... 197
13.1.1 Direkte Eingabe eines Pawn Scripts ..... 197
13.1.2 Hochladen eines Binary-Files ..... 197
13.1.3 Script-Editor des rapidM2M Toolset ..... 197
13.2 rapidM2M API ..... 198
13.2.1 Core Funktionen ..... 198
13.2.2 Consolen Funktionen ..... 201
13.2.3 rapidM2M Funktionen ..... 202
13.2.3.1 Arrays mit symbolischen Indizes ..... 202
13.2.3.2 Konstanten ..... 204
13.2.3.3 Callback Funktionen ..... 207
13.2.3.4 Funktionen ..... 208
13.2.4 File Transfer Funktionen ..... 241
13.2.4.1 Arrays mit symbolischen Indizes ..... 241

13.2.4.2 Konstanten ... 241
13.2.4.3 Callback Funktionen ... 242
13.2.4.4 Funktionen ... 243
13.2.5 Mathematische Funktionen ... 246
13.2.6 String Funktionen ... 248
13.2.7 Hilfsfunktionen ... 252
13.2.7.1 Arrays mit symbolischen Indizes ... 252
13.2.7.2 Konstanten ... 253
13.2.7.3 Funktionen ... 253
13.3 Pawn Script Fehlercodes ... 255
13.4 Syntax ... 258
13.4.1 Allgemeine Syntax ... 258
13.4.1.1 Format ... 258
13.4.1.2 Optionale Semikolons ... 258
13.4.1.3 Kommentare ... 258
13.4.1.4 Bezeichner ... 259
13.4.1.5 Reservierte Schlüsselworte ... 259
13.4.1.6 Numerische Konstanten ... 259
13.4.1.6.1 Numerische Integer-Konstanten ... 259
13.4.1.6.2 Numerische Gleitkomma-Konstanten ... 260
13.4.2 Variablen ... 260
13.4.2.1 Deklaration ... 260
13.4.2.2 Lokale Deklaration ... 260
13.4.2.3 Globale Deklaration ... 260
13.4.2.4 Statische lokale Deklaration ... 260
13.4.2.5 Statische globale Deklaration ... 261
13.4.2.6 Gleitkommawerte ... 261
13.4.3 Konstante Variablen ... 261
13.4.4 Array Variablen ... 262
13.4.4.1 Eindimensionales Array ... 262

13.4.4.2 Initialisierung ... 262
13.4.4.3 Progressive Initialisierung für Arrays ... 262
13.4.4.4 Mehrdimensionale Arrays ... 263
13.4.4.5 Arrays und der "sizeof"-Operator ... 263
13.4.5 Operatoren und Ausdrücke ... 264
13.4.5.1 Zeichenerklärung ... 264
13.4.5.2 Ausdrücke ... 264
13.4.5.3 Arithmetik ... 265
13.4.5.4 Bit-Manipulation ... 265
13.4.5.5 Zuweisung ... 266
13.4.5.6 Vergleichsoperatoren ... 266
13.4.5.7 Boolean ... 267
13.4.5.8 Sonstiges ... 268
13.4.5.9 Priorität der Operatoren ... 268
13.4.6 Anweisungen ... 270
13.4.6.1 Statement-Etikett ... 270
13.4.6.2 Zusammengesetzte Anweisungen ... 270
13.4.6.3 Ausdrucksanweisung ... 270
13.4.6.4 Leeres Statement ... 270
13.4.6.5 assert Ausdruck ... 270
13.4.6.6 break ... 271
13.4.6.7 continue ... 271
13.4.6.8 do Statement while ( Ausdruck ) ... 272
13.4.6.9 exit Ausdruck ... 272
13.4.6.10 for ( Ausdruck 1 ; Ausdruck 2 ; Ausdruck 3 ) Statement ... 272
13.4.6.11 goto Etikett ... 273
13.4.6.12 if ( Ausdruck ) Statement 1 else Statement 2 ... 273
13.4.6.13 return Ausdruck ... 274
13.4.6.14 switch ( Ausdruck ) { case Liste } ... 274
13.4.6.15 while ( Ausdruck ) Statement ... 275

13.4.7 Funktionen ... 275
13.4.7.1 Funktionsargumente ("call-by-value" versus "call-by-reference") ... 277
13.4.7.2 Benannte Parameter versus positionsgebundene Parameter ... 278
13.4.7.3 Standardwerte von Funktionsargumenten ... 279
13.5 Beispiele ... 280
13.6 Unterschiede zu C ... 280
**Kapitel 14 myDatanet Connector ... 283**
14.1 Voraussetzungen ... 283
14.2 Split-tag ... 284
14.3 Datenstruktur ... 286
14.3.1 Attribute der Feld-Definition ... 287
14.4 Beispiel ... 291
14.5 Spezialwerte der Datentypen ... 292
**Kapitel 15 Fehlersuche und Behebung ... 293**
15.1 Allgemeine Probleme ... 293
15.2 Log-Einträge und Fehlercodes ... 295
15.2.1 Modemfehler ... 298
**Kapitel 16 Ersatzteile und Zubehör ... 303**
16.1 rapid M2M-Module ... 303
16.2 Antennen ... 303
16.3 Versorgung ... 303
16.4 Adapter ... 303
16.5 Sonstiges Zubehör ... 303
**Kapitel 17 Dokumentenhistorie ... 305**
**Kapitel 18 Glossar ... 311**
**Kapitel 19 Kontaktinformationen ... 313**

6872
6872
6872

# Kapitel 2 Technische Daten

***Hinweis:*** *Die folgenden technischen Spezifikationen gelten für das rapidM2M Developer Board mit eingesetztem rapidM2M Modul M120 . Eine detaillierte technische Beschreibung des M120 Moduls finden Sie im Kapitel "Technische Daten" auf Seite 99.*

| | |
|---|---|
| Spannungsversorgung | 5...32VDC , max. t.b.d. mA@12V |
| Gehäuse | Gewicht: 134g (inkl. rapidM2M Modul M120 )<br>Abmessungen (BHT): 160 x 25 x 110mm (ohne Antenne und Anschlusskabel) |
| Betriebstemperatur | -5...+40°C, 15...90%rH nicht kondensierend |
| Lagertemperatur | -20...+70°C |
| Antennenanschluss | SMA |
| Serielle Schnittstellen | 2 x RS232[1)] für die Datenkommunikation<br>• Baudrate: 300-115200<br>• Stopbits: 1, 2<br>• Parität: N, E, O<br>• Datenbits: 7, 8<br>1 x SPI<br>• Modus: SPI-Master<br>• Clock Polarität/Phase einstellbar<br>• Clockfrequenz: 102kHz...13MHz<br>• Interface: 4-wire<br>1 x $I^2C$<br>• Modus: Master<br>• Clockfrequenz: 100kHz...400kHz<br>Zusätzliche Informationen finden Sie unter:<br>• "RS232-Schnittstellen" auf Seite 71<br>• "SPI" auf Seite 111<br>• "I2C" auf Seite 112 |
| USB-Schnittstelle | 1 x Mini-B USB 2.0 Slave für die Verbindung mit einem PC. Für die Kommunikation mit dem rapidM2M Developer Paket M12x muss am PC das rapidM2M Toolset installiert sein.<br>Zusätzliche Informationen finden Sie unter "USB-Schnittstelle" auf Seite 80. |

| | |
|---|---|
| Temperatursensor[2)] | Messbereich: -40...+125°C<br>Auflösung: 0,0625°C<br><br>Zusätzliche Informationen finden Sie unter "Temperatursensor TMP112 (U300)" auf Seite 84. |
| Umgebungslicht und IR-Näherungssensor[2)] | Messbereich: 0,03...65535lux<br><br>Zusätzliche Informationen finden Sie unter "Umgebungslicht und IR-Näherungssensor MAX44000 (U301)" auf Seite 85. |
| Analogeingänge[3)] | 3 x Differenzeingang<br>• Auflösung: 16 Bit<br>• Eingangsspannungsbereich: $V_{ref}$ / Gain<br>• Interne Referenzspannung: 1,17V<br>• Externe Referenzspannung: nominal 2,5V, max 3,3V<br>• Gain: 1-128<br><br>Zusätzliche Informationen finden Sie unter "ADC AD7792 (U302)" auf Seite 83. |
| Ausgänge[3)] | 2 x Konstantstromausgang<br>• Ausgangsstrom: 10/210/1000µA<br><br>Zusätzliche Informationen finden Sie unter "ADC AD7792 (U302)" auf Seite 83. |
| I/O-Leitungen | 6 x GPIO[4)]<br>2 x Interrupt-Eingänge |
| Simulationsschaltungen | 6 x Schaltung zur Eingabe von digitalen Signalen (Taster) bzw. Anzeige von digitalen Signalen (LED)<br>1 x Buzzer zur Ausgabe eines digitalen Signals |
| Datenspeicher | 3 x 349.525 Byte interner Flash-Speicher.<br>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. |
| Konfigurationsspeicher | 10 unabhängige Blöcke mit je 10kB |
| Registrierungsspeicher | 4 Blöcke mit je 1kB und vordefinierten Verwendungszwecken zur Ablage gerätespezifischer Daten<br><br>Zusätzliche Informationen finden Sie unter "Registrierungsspeicherblöcke" auf Seite 63. |

| | |
|---|---|
| Speicher für das PAWN-Binary | 256kB (unkomprimierte Größe)<br><br>Zusätzliche Informationen finden Sie unter "Speicherorganisation" auf Seite 105. |
| Datenübertragung | mittels GSM/GPRS Quad-Band Modem an den jeweiligen Cloud-Server |
| SIM | Das rapidM2M Modul M120 ist mit einem integrierten SIM-Chip versehen. |

[1)] Die RS232-Pegelwandler befinden sich am rapidM2M Developer Board . Bei der späteren Integration des M120 Moduls in eine eigene Hardware muss beachtet werden, dass die Ausgangspegel der UARTs des M120 Moduls von der Versorgung der Level Shifter des Moduls abhängig sind (siehe "Technische Details zu den Level Shiftern" auf Seite 110).

[2)] Realisiert über Sensoren (TMP112 und MAX44000), die sich am rapidM2M Developer Board befinden und mittels $I^2C$ Bus mit dem rapidM2M Modul M120 verbunden sind.

[3)] Realisiert über einen ADC (AD7792), der sich am rapidM2M Developer Board befindet und mittels SPI Bus mit dem rapidM2M Modul M120 verbunden ist.

[4)] Die 6 GPIOs können mittels Dip-Switch S206, S207 und S208 mit den Simulationsschaltungen verbunden werden.

6872
6872
6872

# Kapitel 3 Allgemeine Angaben

## 3.1 Allgemeine Produktinformationen

Es handelt sich um ein Developer Paket, das den Kunden bei der Integration des M120 Moduls in eigene Geräte unterstützen soll. Das rapidM2M Developer Board ist mit standardisierten Steckern, Stiftleisten, Klemmleisten, Tastern und LEDs (siehe "Übersicht" auf Seite 67) sowie einigen Sensoren ausgestattet, um einen einfachen und schnellen Aufbau von Testschaltungen zu gewährleisten. Bei dem im Lieferumfang des Developer Pakets M120 enthaltenen rapidM2M Modul M120 handelt es sich um ein Modul zur Aufzeichnung und Übertragung von Messdaten, die über standardisierte Mikrocontroller-Schnittstellen (2 x UART, 1 x SPI, 1 x $I^2C$) gelesen werden. Der Kunde kann mittels PAWN-Script (siehe "Pawn Script" auf Seite 197) seine eigene Applikation erstellen und diese per USB-Schnittstelle oder GPRS-Verbindung ins rapidM2M Modul M120 laden. Dies erfolgt entweder über das rapidM2M Toolset (siehe "rapidM2M Toolset " auf Seite 141) oder über die XML-Schnittstelle des Servers (siehe "XML-Schnittstelle" auf Seite 157). Das PAWN-Script ermöglicht den Zugriff auf die Mikrocontroller-Schnittstellen, wodurch der Kunde die Möglichkeit erhält nahezu alle mit diesen Schnittstellen kompatiblen Bausteine anzubinden. Zudem verfügt das rapidM2M Modul M120 noch über 6 GPIOs über die Steuersignale zwischen M120 und der umgebenden Hardware ausgetauscht werden können. Für zeitkritische Rückmeldungen der umgebenden Hardware an das M120 stehen 2 Interruptleitungen zur Verfügung.

Das rapidM2M Modul M120 bietet dem Kunden einen Speicherbereich für seine Daten (3 x 349.525 Byte) sowie 10 voneinander unabhängige Speicherblöcke für Konfigurationsdaten mit je 10kB . Zudem verfügt das rapidM2M Modul M120 noch über 4 Registrierungsspeicherblöcke mit je 1kB und vordefinierten Verwendungszwecken zur Ablage gerätespezifischer Daten (siehe "Registrierungsspeicherblöcke" auf Seite 63). 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 M120 Modul mit einem integrierten SIM-Chip versehen.

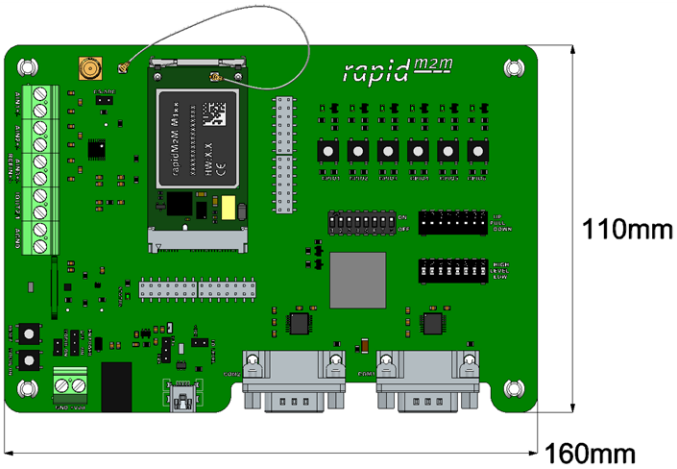
## 3.2 Lieferumfang

Zum Standardlieferumfang des Developer Pakets (300144) gehören:

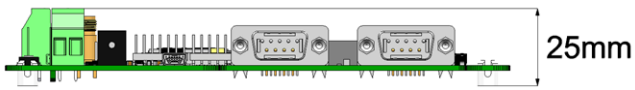
- rapidM2M Developer Board
- rapidM2M Modul M120 (300087 )
- Steckernetzgerät 6W 12V
- Magnetfußantenne SMA-M 2,5m (300085)
- 10 x Steckbrücke
- U.FL Antennenkabel
- myDatanet Tool Pen (206.646)
- USB-Kabel
- Nullmodemkabel
- Quick Start (300414)
- Kurzanleitung (300384)

Weiteres Zubehör wie M2M-Module, Antennen, Netzteile, Sensoren usw. je nach Bestellung. Diese bitte anhand des Lieferscheins prüfen.

## 3.3 Abmessungen



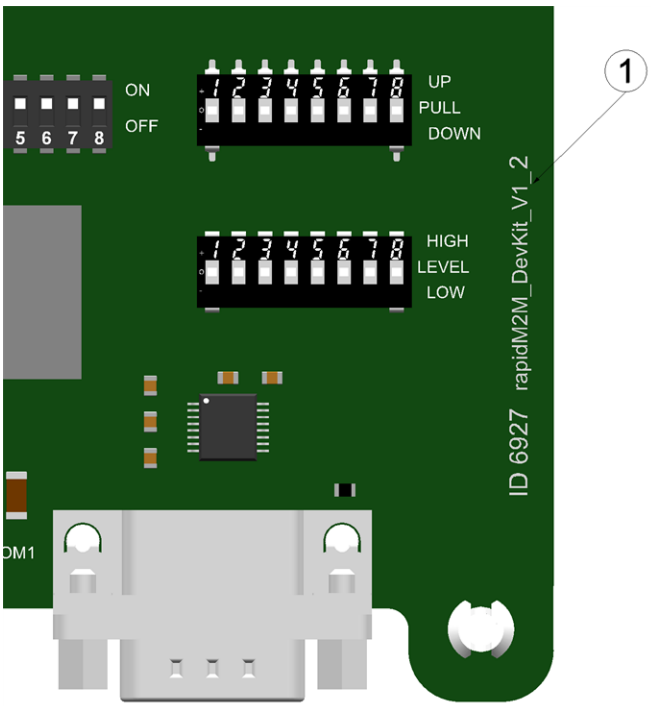

Abmessungen: Breite und Tiefe




Abmessungen: Höhe

## 3.4 Gerätekennzeichnung

Die Angabe, um welche Hardwareversion des Developer Boards es sich handelt, finden Sie auf der rechten Seite der Platine in der Nähe der Dip-Switches S208 und S206.




Hardwarekennzeichnung des Developer Boards

| | |
|---|---|
| 1 Hardwareversion | |

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

## 3.5 Aufbewahrung des Produkts

Zur Aufbewahrung des Developer Pakets stellen Sie sicher, dass alle relevanten Daten zum Cloud-Server übertragen wurden. Gegebenenfalls lösen Sie dazu direkt am Gerät, falls Sie dies in Ihrem PAWN-Script vorgesehen haben, 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 Cloud–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 Cloud–Server auslösen. Kontrollieren Sie anschließend erneut, ob nun alle relevanten Daten übertragen wurden. Bei der Verbindungsart „online“ werden die ermittelten Messdaten sofort zum Cloud–Server übertragen. Die Daten am Server sind immer aktuell und das Gerät kann jederzeit abgeschaltet werden. Als nächstes trennen Sie das rapidM2M Developer Board von der Versorgungsspannung. Wenn möglich, sollte auch die Versorgung aller angeschlossenen

Sensoren abgeschaltet werden bevor Sie die Verbindungen zw. dem rapidM2M Developer Board und den Sensoren trennen. Anschließend kann die Antenne entfernt werden. Bewahren Sie das rapidM2M Developer Paket M12x  inkl. Zubehör und Anschlusskabel in der Originalverpackung auf.

Die Konfiguration und die zuletzt ermittelten Daten bleiben erhalten. Die Systemzeit läuft allerdings ohne Energieversorgung nicht weiter. Das bedeutet, dass nach der Wiederinbetriebnahme eine Zeitsynchronisation erforderlich ist, damit die Zeitstempel der Mess- und Log-Daten wieder korrekt sind. Diese erfolgt automatisch bei der ersten Verbindung zum Cloud–Server.

# Kapitel 4 Tutorial

Das folgende Tutorial demonstriert Ihnen anhand des mit dem rapidM2M Toolset gelieferten Scripts "app.p" die Erstellung einer M2M Applikation. Sie erfahren auch wie Sie unter Verwendung der Auswertungen Daten am Cloud-Server darstellen und wie Sie die Eingabe von Konfigurationseinstellungen über die Oberfläche des Cloud-Servers ermöglichen.

## 4.1 Beschreibung der Funktion des Scripts "app.p"

Das Script erfasst periodisch Messdaten und überträgt diese an den Cloud-Server. Sowohl das Intervall für die Erfassung als auch die Übertragung der Messdaten lässt sich konfigurieren. Es ist auch möglich, die Verbindungsart zu konfigurieren. Wird die Verbindungsart "online" gewählt, werden die Messdaten unverzüglich sobald sie erstellt wurden zum Server übertragen. Neben einigen berechneten Messwerten werden die Werte des Temperatursensors TMP112 (U300) sowie des Umgebungslicht- und IR-Näherungssensors MAX44000 (U301) verwendet. Diese beiden Sensoren befinden sich am rapidM2M Developer Board und sind per $I^2C$ an das rapidM2M Modul M120 angebunden. Über LED200 (GPIO1) wird ein "alive" Signal ausgegeben, d.h. die LED blinkt im Sekundentakt. Mittels der Taste S201 (GPIO2) kann eine sofortige Aufzeichnung der Messwerte ausgelöst werden. LED202 - LED205 (GPIO2 - GPIO5) können per Konfigurationsparameter über die Oberfläche des Cloud-Servers ein- bzw. ausgeschaltet werden.

Das Script erzeugt und überträgt folgende Messwerte:

| Messwert | Einheit | Erklärung |
|---|---|---|
| Count | --- | Endloszähler, der die Anzahl der erzeugten Messdatensätze seit dem letzten PowerOn enthält |
| Timestamp | sec. | Sekunden seit 31.12.1999 (Es handelt sich hierbei nicht um den Wert, der vom System für die Verwaltung der Messdaten verwendet wird.) |
| Temperatur | °C | Messwert, der per $I^2C$ vom Temperatursensor TMP112 (U300) des rapidM2M Developer Board gelesen wird |
| Ambient Light | lux | Messwerte, die per $I^2C$ vom Umgebungslicht- und IR-Näherungssensor MAX44000 des rapidM2M Developer Board gelesen werden |
| Proximity | % | |

Das Script biete folgende Konfigurationsmöglichkeiten:

| Parameter | Einheit | Erklärung |
|---|---|---|
| Version | --- | Anzeige der aktuellen Version der Konfiguration (Read Only). Wird durch das Script vorgegeben und sollte geändert werden, wenn sich die Struktur der Konfiguration ändert (z.B. durch einen weiteren Parameter). |
| Aufzeichnungsintervall | sec. | zeitlicher Abstand der Messdatenaufzeichnungen |
| Übertragungsintervall | sec. | zeitlicher Abstand der Übertragungen |
| Modus | --- | 0: Das Gerät meldet sich im Übertragungsintervall.<br>1: Das Gerät meldet sich im Übertragungsintervall und kann über den Server in den ALOHA-Übertragungsmodus versetzt werden.<br>2: Das Gerät trennt die Verbindung nicht und übermittelt kontinuierlich die Messdaten. |
| GPIO ON/OFF | --- | Ermöglicht das gezielte Ein- bzw. Ausschalten der LEDs.<br>Bit0: LED202 (GPIO3); Bit gesetzt -> LED ein<br>Bit1: LED203 (GPIO4); Bit gesetzt -> LED ein<br>Bit2: LED204 (GPIO5); Bit gesetzt -> LED ein<br>Bit3: LED205 (GPIO6); Bit gesetzt -> LED ein<br>Wird dieser Parameter auf "8" gesetzt, leuchtet das LED205 (GPIO6). |

## 4.2 Voraussetzungen

Für dieses Tutorial benötigen Sie folgende Komponenten:

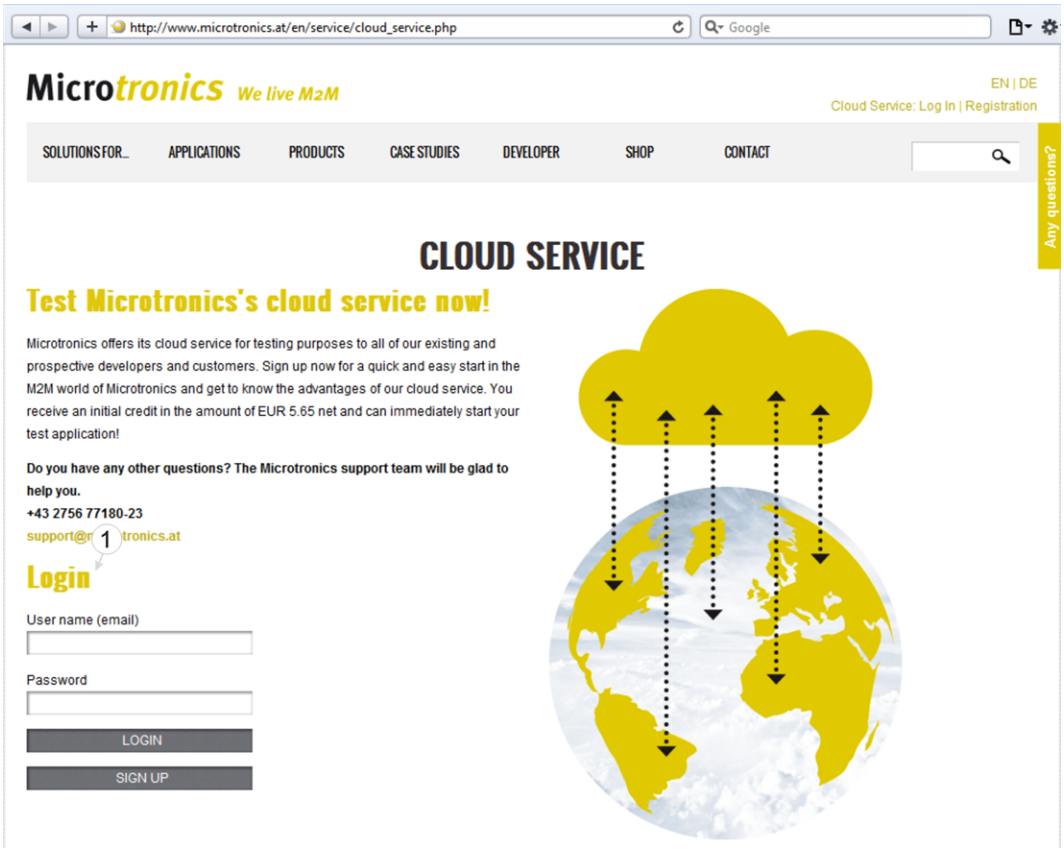
- PC mit Microsoft Betriebssystem (Win XP, Windows Vista, Windows 7 oder Windows 8)
- rapidM2M Developer Paket M120 (300144 )
- rapidM2M Toolset (Mindestversion 1.12, Download unter: www.microtronics.at/Toolset)
- Account am Cloud-Server (siehe "Neu registrieren und anmelden" auf Seite 117)

## 4.3 Schritt-für-Schritt-Anleitung

1. Sollten Sie noch nicht über einen Account am Cloud-Server verfügen, können Sie über die Homepage der Firma Microtronics Engineering GmbH(cloud.microtronics.at) gratis einen Account anlegen. Detaillierte Informationen zum Anlegen eines Accounts am Cloud-Server finden Sie im Kapitel "Neu registrieren und anmelden" auf Seite 117.

2. Installieren Sie das rapidM2M Toolset . Es steht unter folgender Adresse gratis zum Download bereit: www.microtronics.at/Toolset. Detaillierte Informationen zur Installation des rapidM2M Toolset finden Sie im Kapitel "Installation" auf Seite 143.

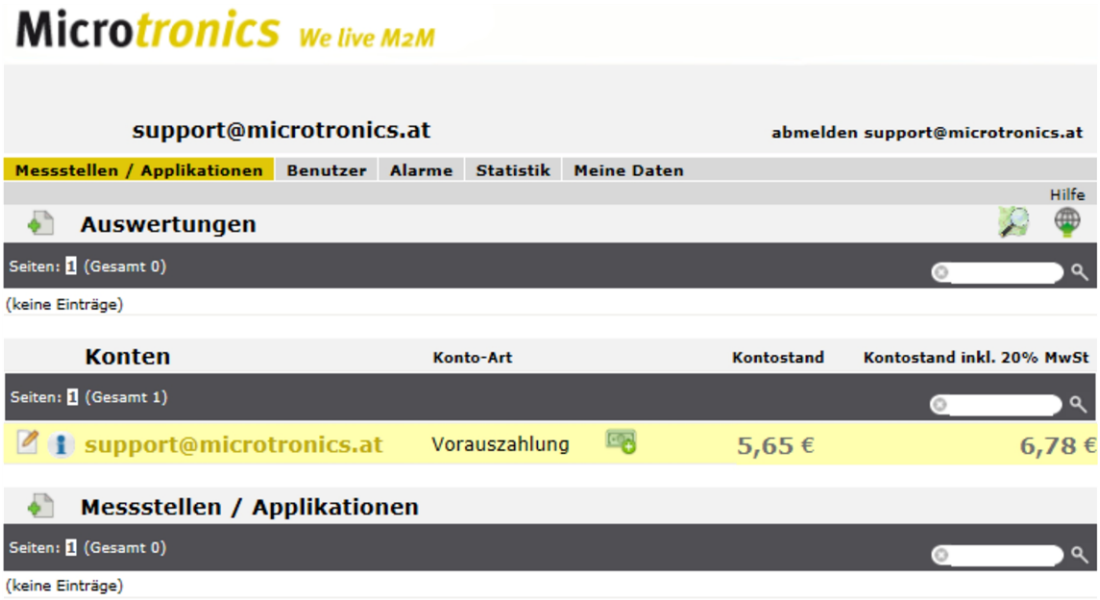
3. Loggen Sie sich am Cloud-Server (cloud.microtronics.at) ein. Der Benutzername ist die E-Mailadresse, die Sie beim Anlegen des Accounts angegeben haben.




Übersicht der Homepage

| | |
|---|---|
| 1 Zugang zum Cloud-Server | |

4. Sie befinden sich nun im Bereich "Messstellen / Applikationen" des Cloud-Servers (siehe "Bereich "Messstellen"" auf Seite 120).




Bereich "Messstellen / Applikationen", in dem alle zur Verfügung stehenden Messstellen angezeigt werden

5. Legen Sie eine Messstelle an (siehe "Anlegen einer neuen Messstelle" auf Seite 125 für Details).



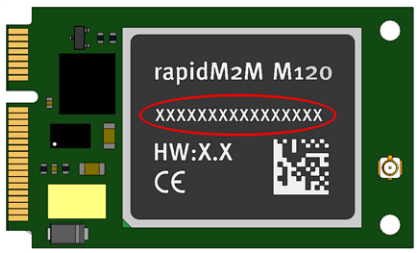

Anlegen der Messstelle

| | | | |
|---|---|---|---|
| 1 | Symbol "Neue Messstelle / Applikation hinzufügen" | 3 | Eingabefenster für das Anlegen einer neuen Messstelle |
| 2 | Feld zur Eingabe der Seriennummer | 4 | Button "hinzufügen" |

Die Seriennummer finden Sie auf der Verpackung des rapidM2M Developer Paket M12x sowie auf dem rapidM2M Modul M120 selbst.

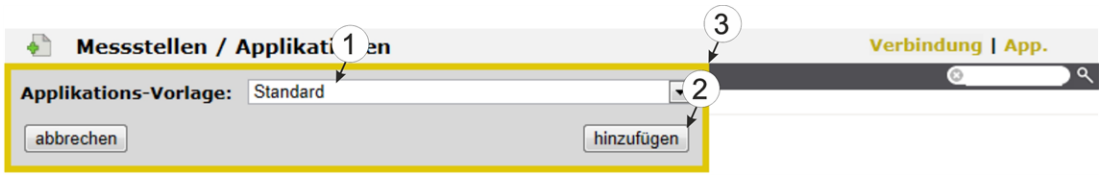



Verpackung des rapidM2M Developer Paket M12x




rapidM2M Modul M120

Bei der "Applikations-Vorlage" wählen Sie bitte "Standard".




Auswahl der Applikations-Vorlage

| | | | |
|---|---|---|---|
| 1 | Dropdown-Liste der verfügbaren Applikations-Vorlagen | 3 | Eingabefenster zur Auswahl der Applikations-Vorlage |
| 2 | Button "hinzufügen" | | |

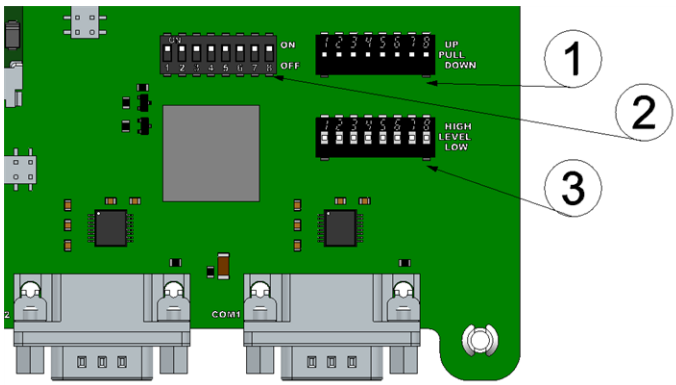
6. In der Liste der Messstellen wird nun die neu hinzugefügte Messstelle angezeigt.




Bereich "Messstellen / Applikationen", in dem alle zur Verfügung stehenden Messstellen angezeigt werden

| | |
|---|---|
| 1 | öffnet die Eingabemaske zur Konfiguration der Messstelle (siehe "Messstellenkonfiguration" auf Seite 193) |
| 2 | löscht die Messstelle / Applikation. |
| 3 | Seriennummer des mit der Messstelle verknüpften Geräts. Durch Klicken auf die Seriennummer wird die Eingabemaske zur Konfiguration des Geräts geöffnet (siehe "Gerätekonfiguration" auf Seite 195). |
| 4 | Name der Messstelle / Applikation. Durch Klicken auf den Namen wird die Eingabemaske zur Konfiguration der Messstelle / Applikation geöffnet (siehe "Messstellenkonfiguration" auf Seite 193). |
| 5 | Liste der Messstellen / Applikationen |
| 6 | Status des Messgeräts. Neben dem Statussymbol werden auch, abhängig vom aktuellen Betriebszustand, weitere Symbole eingeblendet. (siehe "Messstellen" auf Seite 123 für Details) |
| 7 | Schaltflächen zum Wechsel zwischen der Anzeige von verbindungsspezifischen und applikationsspezifischen Steuerelemente |
| 8 | Verbindungs- bzw. applikationsspezifische Steuerelemente der Messstelle / Applikation (siehe "Messstellen" auf Seite 123 für Details). |

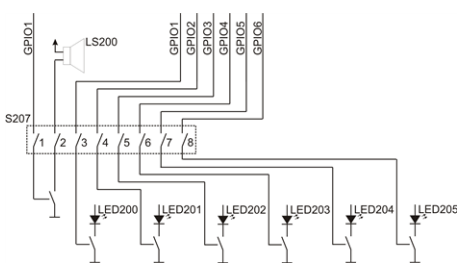
7. Als Vorbereitung für das Script, das bei diesem Tutorial verwendet werden soll, müssen die Dip-Switches S207 - S208 wie folgt eingestellt werden:



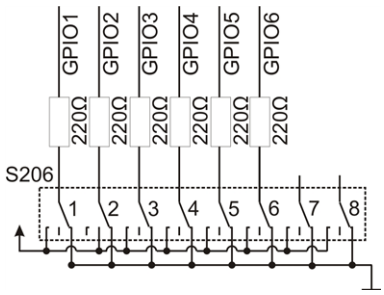

Position der Dip-Switches S207 - S208

| | |
|---|---|
| 1 | Dip-Switch (S206) zum Aktivieren der Pull up bzw. Pull down Widerstände für den jeweiligen GPIO (siehe "Eingabe von digitalen Signalen" auf Seite 74) |
| 2 | Dip-Switch (S207) zum Aktivieren der LEDs für den jeweiligen GPIO und des Buzzers (LS200), der durch den GPIO1 angesteuert wird (siehe "Ausgabe von digitalen Signalen" auf Seite 73) |
| 3 | Dip-Switch (S208) zum Festlegen, ob der jeweilige GPIO durch Drücken der zugeordneten Taste auf "low" oder "high" gesetzt wird (siehe "Eingabe von digitalen Signalen" auf Seite 74) |

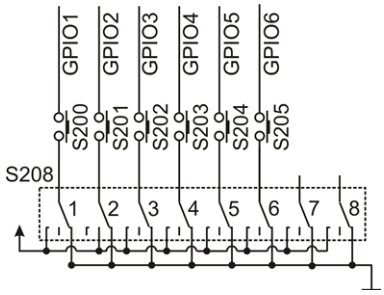
| | Dip-Switch (S207) | Dip-Switch (S206) | Dip-Switch (S208) |
|---|---|---|---|
| 1 | OFF (Buzzers deaktiviert) | o | o |
| 2 | OFF (Buzzers deaktiviert) | - (Pull down) | + ( Tastendruck zieht GPIO auf "high") |
| 3 | ON | o | o |
| 4 | ON | o | o |
| 5 | ON | o | o |
| 6 | ON | o | o |
| 7 | ON | | |
| 8 | ON | | |




Dip-Switch (S207)

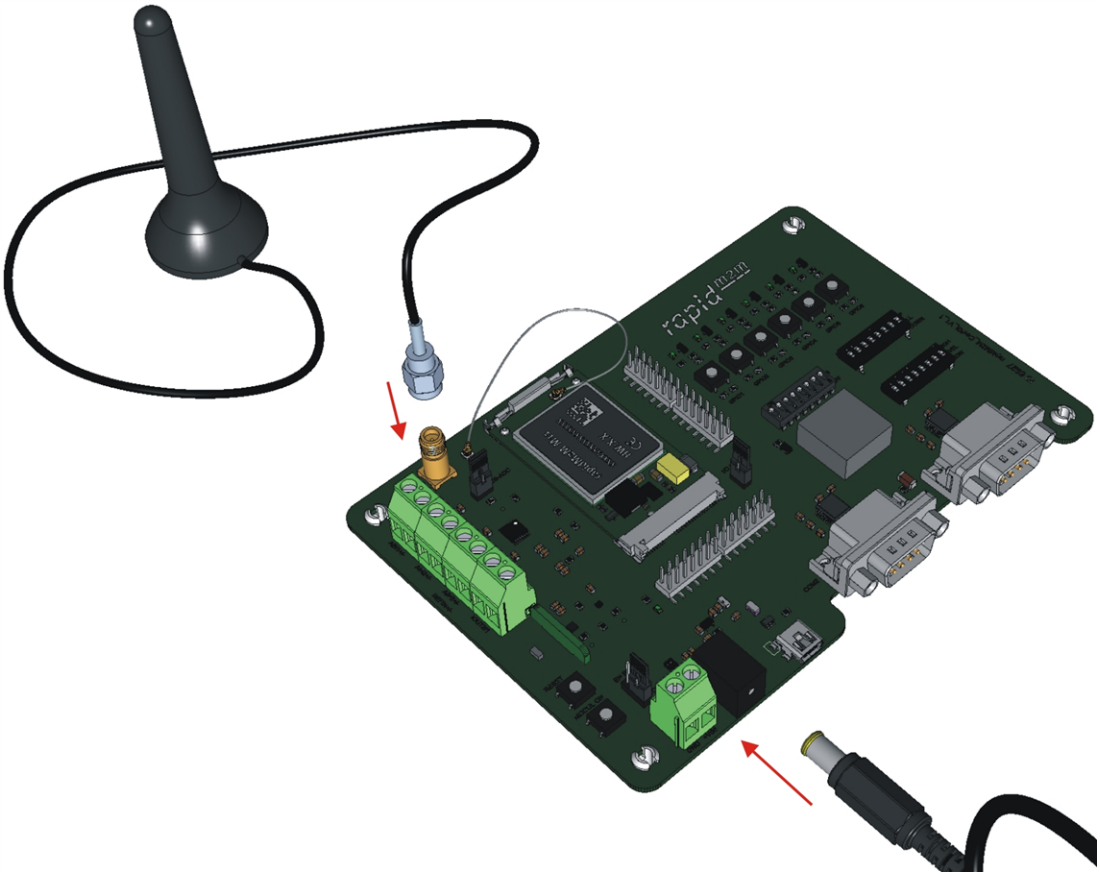



Dip-Switch (S206)



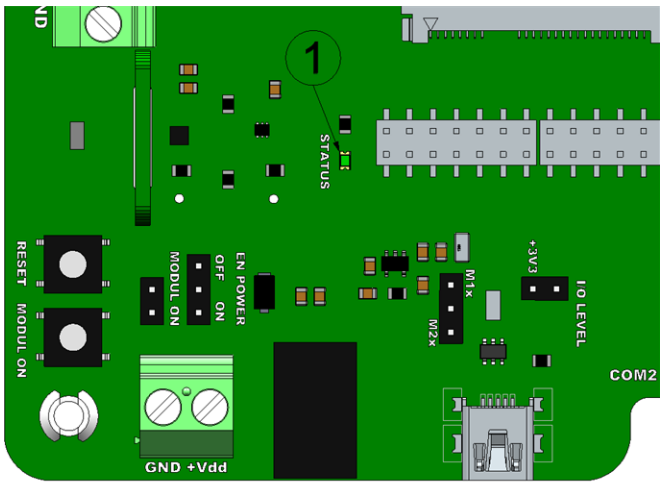

Dip-Switch (S208)

8. Verbinden Sie nun zuerst die Magnetfußantenne SMA-M 2,5m und anschließend das Steckernetzgerät mit dem rapidM2M Developer Paket M12x . Alle Jumper sind im Auslieferungszustand so gesetzt, dass das M120 Modul beim Herstellen der Energieversorgung startet.



Anschließen der Antenne und der Versorgung

Die Status-LED (LED100) leuchtet nach dem Herstellen der Energieversorgung ca. 1sec.




Position der Status-LED (LED100)

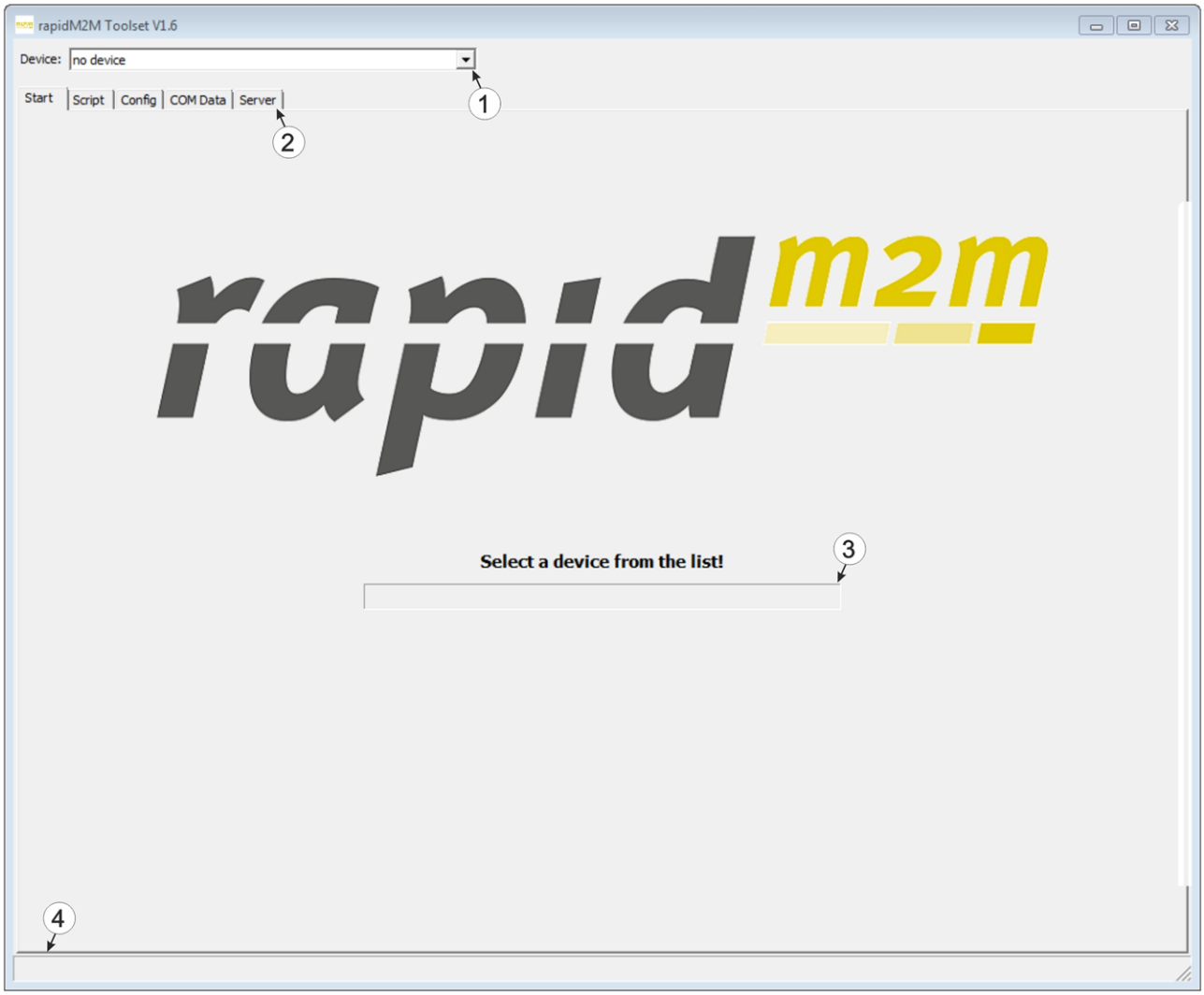
9. Im Auslieferungszustand befindet sich kein Script im Speicher des M120 Moduls, wodurch direkt nach dem Herstellen der Energieversorgung eine Verbindung zum Cloud-Server hergestellt wird. Eine erfolgreiche Kommunikation zwischen M120 Modul und Cloud-Server erkennen Sie anhand des Statussymbols.




Liste der Messstellen / Applikationen

| | | |
|---|---|---|
| 1 | Status des Messgeräts (siehe "Messstellen" auf Seite 123 für Details) | |
| |  | Die Verbindung mit dem Cloud-Server war erfolgreich. |
| |  | Das M120 Modul hat sich noch nicht mit dem Cloud-Server verbunden. |
| 2 | Zeigt an, dass das Modul mit dem Server verbunden ist und Daten überträgt. Dieses Symbol wird wieder ausgeblendet nachdem alle Daten übertragen wurden. | |
| 3 | Zeitpunkt der letzten Verbindung. Durch einen Klick wird die Verbindungsliste geöffnet. | |
| 4 | Die Verbindung erfolgte via GSM/GPRS. Das Symbol gibt die Signalstärke an. Ein Klick auf das Symbol zeigt Informationen zur Funkzelle, die bei der letzten Verbindung verwendet wurde sowie zu den bei der letzten Positionsbestimmung berücksichtigten Funkzellen an. | |

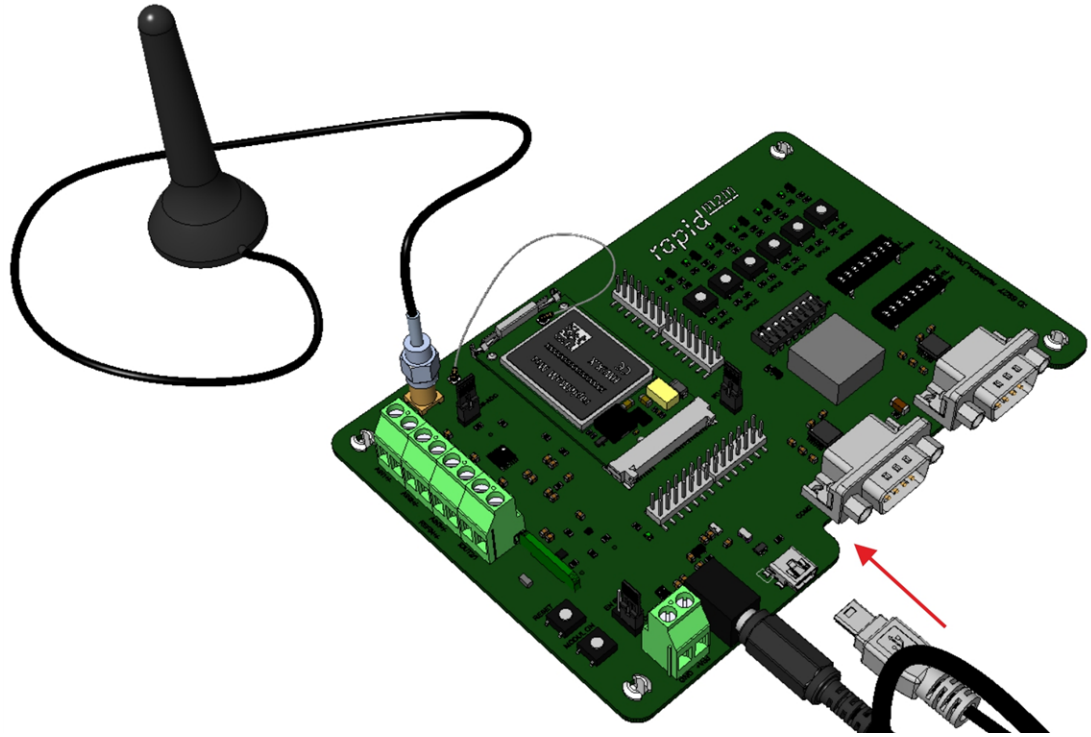
10. Starten Sie das rapidM2M Toolset .




Karteireiter "Start" (kein M120 Modul ausgewählt)

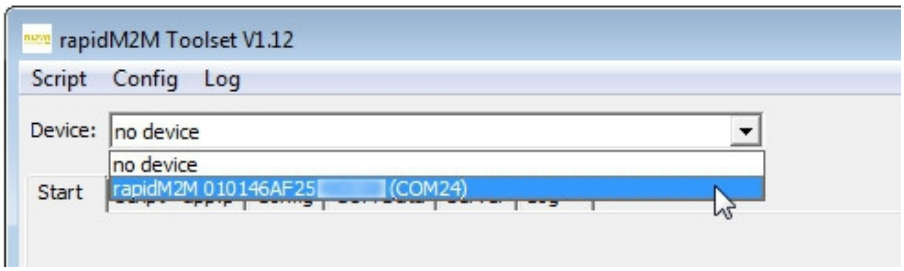
| | |
|---|---|
| 1 Auswahl des M120 Moduls | 3 Fortschrittsbalken für den Aufbau der USB-Verbindung zum M120 Modul |
| 2 Auswahl der Karteireiter für die verschiedenen Funktionen des rapidM2M Toolset , die auch ohne verbundenem Modul verfügbar sind | 4 Anzeige des Kommunikationsstatus auf der USB-Schnittstelle zum M120 Modul |

11. Verbinden Sie das rapidM2M Developer Paket M12x mit der USB-Schnittstelle Ihres PCs.



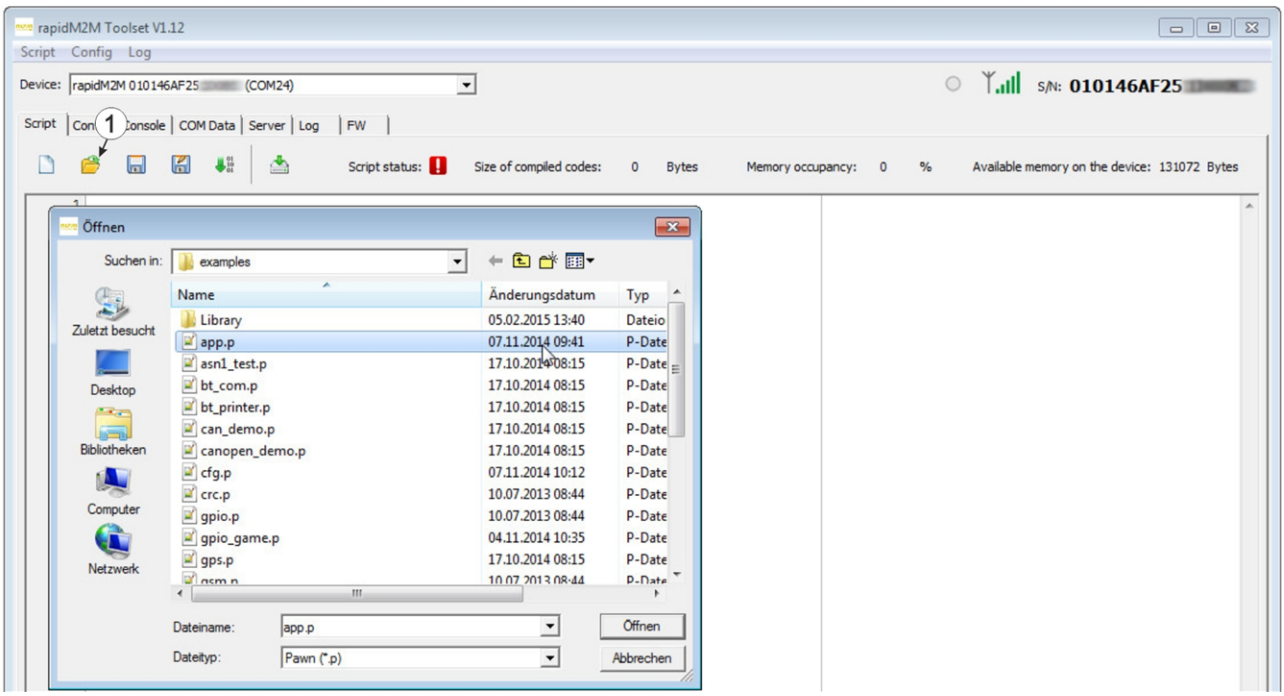
Anschließen des USB-Kabels

12. Wählen Sie das M120 Modul aus.




Modulauswahl

13. Öffnen Sie das mit dem rapidM2M Toolset gelieferte Script "app.p". Es befindet sich im Unterordner "examples" des Installationsverzeichnisses des rapidM2M Toolset .

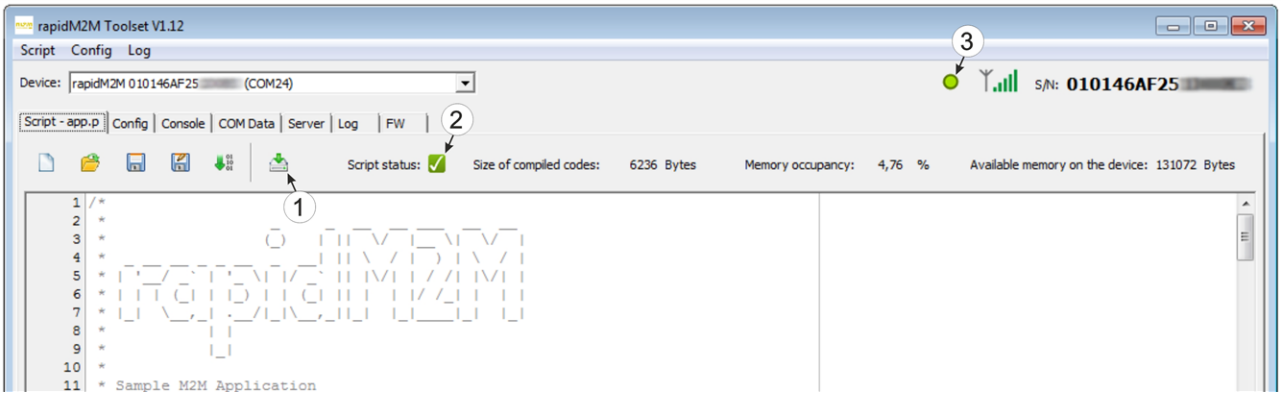



Dialog zum Öffnen eines gespeicherten Scripts.

| | |
|---|---|
| **1** öffnet ein gespeichertes Script | |

Detaillierte Erklärungen zum Script "app.p" finden Sie im Kapitel "Erklärung der rapidM2M spezifischen Codeblöcke des Scripts "app.p"" auf Seite 45.

14. Übertragen Sie das geöffnete Script in das M120 Modul ( -Symbol). Daraufhin wechselt die Anzeige des Script Status in den Zustand "Script wurde erfolgreich in das M120 Modul geladen" ( -Symbol). Die Anzeige des GSM-Verbindungsstatus wechselt nach kurzem Blinken (Verbindung wird hergestellt) auf grün (verbunden), da im Script "app.p" die Verbindungsart standardmäßig auf „online“ gesetzt ist.



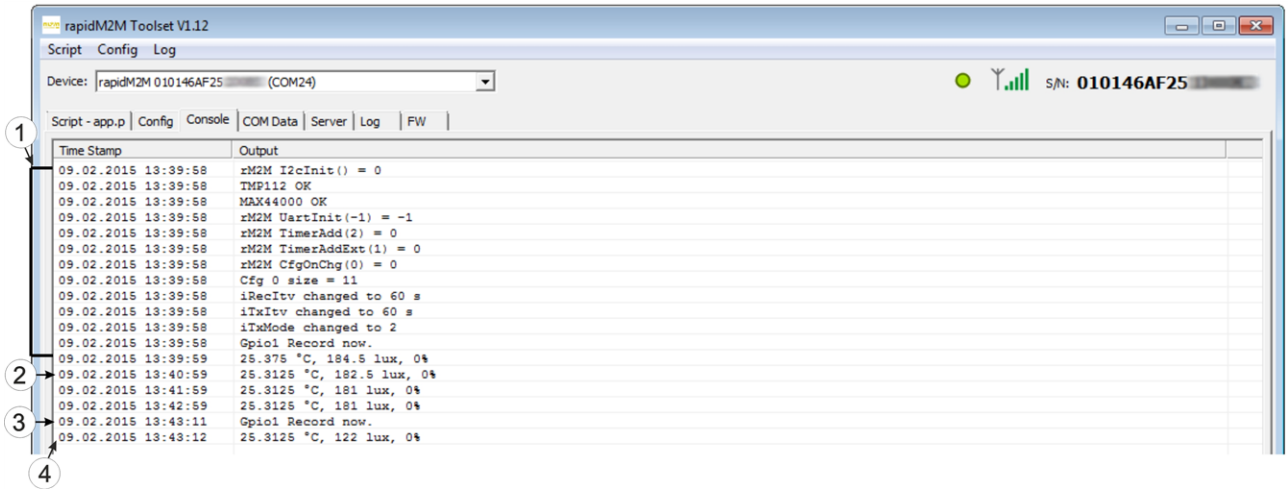

Übertragen des Scripts in das M120 Modul

| | |
|---|---|
| 1 | überträgt das geöffnete Script in das Modul. Dabei wird das Script zuerst gespeichert und dann kompiliert. Das kompilierte PAWN-Binary (*.amx) wird im selben Ordner wie das Script gespeichert. |
| 2 | Status der Script-Übertragung (siehe Script Status) |
| 3 | Anzeige der GSM-Signalstärke (siehe GSM-Signalstärke) |

Der folgende Schritt ist nicht zwingend erforderlich.

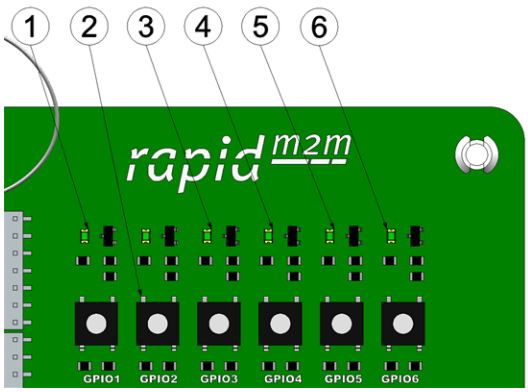
15. Wenn Sie zum Karteireiter "Console" wechseln, können Sie sich die Debug-Informationen ansehen, die vom Script "app.p" ausgegeben werden. Dazu gehört auch die Anzeige der Messwerte (Temperatur, Umgebungslicht und Entfernung), die von den beiden Sensoren gelesen und anschließend zum Cloud-Server übertragen werden. Standardmäßig erzeugt das Script "app.p" alle 60sec. einen neuen Messwert. Sie können aber auch durch Drücken der Taste S201, die mit GPIO2 verbunden ist, eine sofortige Messung auslösen. Um eine deutliche Veränderung der Messwerte zu erreichen, decken Sie den Umgebungslicht- und IR-Näherungssensors MAX44000 (U301) ab oder berühren (nicht empfohlen) Sie den Temperatursensor TMP112 (U300).

> ***Wichtiger Hinweis:*** *Bitte beachten Sie, dass die empfindlichen elektronischen Komponenten des rapidM2M Developer Paket M12x bei Berührung durch statische Elektrizität beschädigt werden können.*

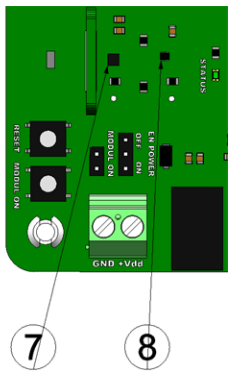



Anzeige der Debug-Informationen

| | |
|---|---|
| 1 | Initialisierung |
| 2 | Messwerte, die durch den Ablauf des 60sec. Intervalls erzeugt wurden |
| 3 | Messwerterzeugung wurde durch Drücken der Taste S201 ausgelöst. |
| 4 | Messwerte, die durch den Tastendruck erzeugt wurden (Erkennbar daran, dass der Timstamp unmittelbar auf den der Tastendruckmeldung folgt.). |




Verwendete Bedienelemente




Verwendete Sensoren

| | |
|---|---|
| 1 | LED200 (GPIO1): "alive" Signal (blinkt im Sekundentakt) |
| 2 | Taste S201 (GPIO2): Messung sofort auslösen |
| 3 | LED202 (GPIO3): per Konfigurationsparameter ein-/ausschaltbar |
| 4 | LED203 (GPIO4): per Konfigurationsparameter ein-/ausschaltbar |
| 5 | LED204 (GPIO5): per Konfigurationsparameter ein-/ausschaltbar |
| 6 | LED205 (GPIO6): per Konfigurationsparameter ein-/ausschaltbar |
| 7 | Temperatursensor TMP112 (U300): Messwert "Temperatur [°C]" |
| 8 | Umgebungslicht- und IR-Näherungssensor MAX44000 (U301): Messwerte "Umgebungslicht [lux]" und "Entfernung [%]" |

16. Die Messwerte werden nun bereits erfasst und aufgrund der Verbindungsart „online“ sofort nach der Ermittlung zum Cloud-Server übertragen. Die Daten werden momentan noch in Rohmessdatenbereich ("rm2mraw") gespeichert. Daher wird in der Anzeige des zur Verfügung stehenden Datenzeitraums noch "KEINE DATEN" signalisiert. Diese Anzeige bezieht sich auf den Messdatenkanal 0 ("histdata0"). Um die vom Script "app.p" erzeugten Messdaten bzw. die bereitgestellte Konfiguration über die Oberfläche des Cloud-Servers nutzen zu können, muss dem Server die Struktur der Messdaten sowie der Konfiguration bekannt sein. Dazu muss der sogenannte myDatanet Connector definiert werden. Eine Beschreibung dazu folgt in den nächsten Schritten.




Liste der Messstellen / Applikationen

| | |
|---|---|
| 1 Anzeige des zur Verfügung stehenden Datenzeitraums | 2 Zeigt an, dass das Modul mit dem Server verbunden ist und Daten überträgt. |

17. Öffnen Sie die Eingabemaske zur Konfiguration der Messstelle / Applikation durch einen Klick auf den Namen der Messstelle / Applikation in der Liste der Messstellen / Applikationen.

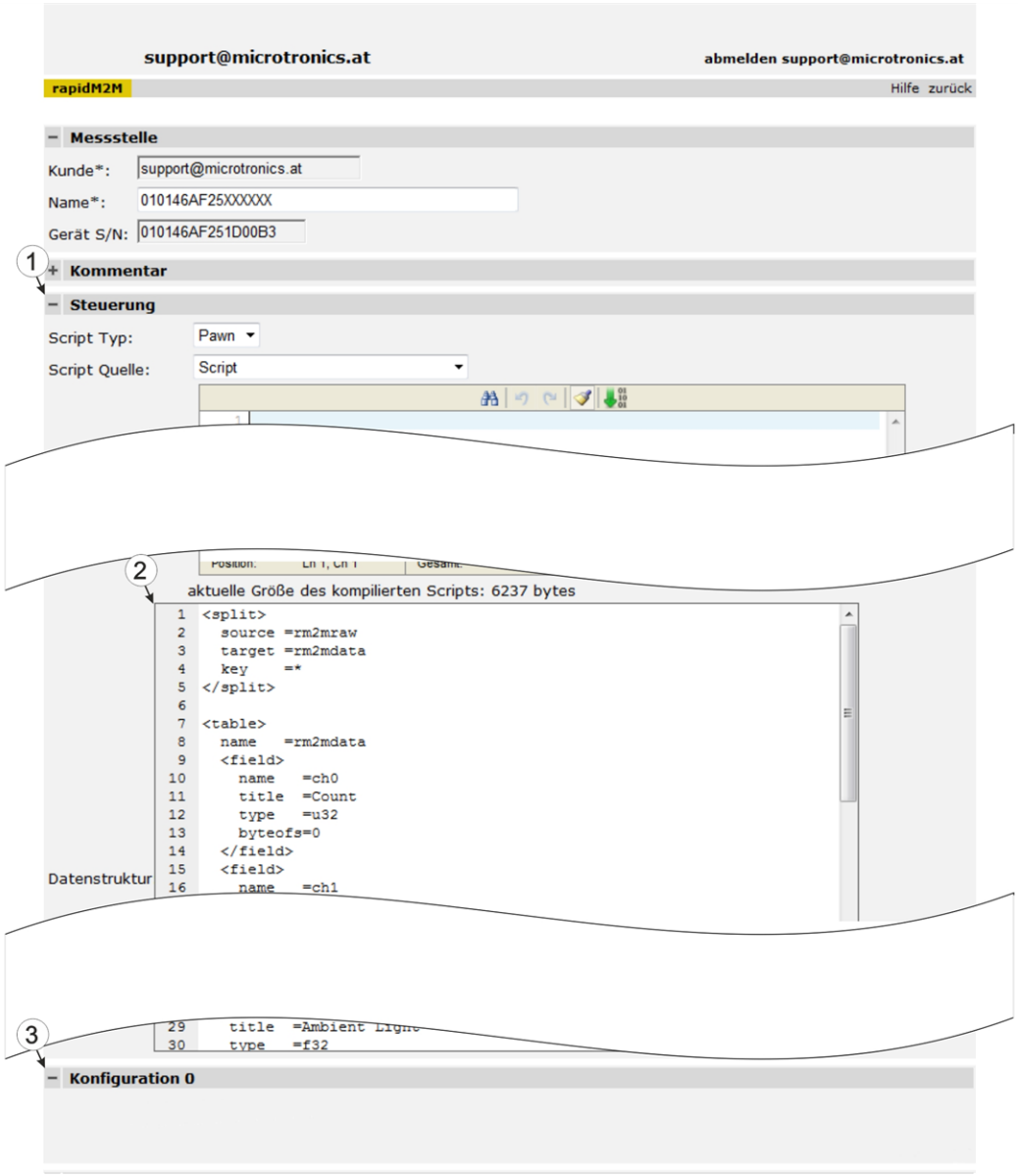



Liste der Messstellen / Applikationen

| | |
|---|---|
| 1 Name der Messstelle / Applikation. Durch Klicken auf den Namen wird die Eingabemaske zur Konfiguration der Messstelle / Applikation geöffnet (siehe "Messstellenkonfiguration" auf Seite 193). | |

18. Klappen Sie den Konfigurationsabschnitt "Steuerung" durch einen Klick auf das "+"-Symbol neben der Beschriftung "Steuerung" auf. Kopieren Sie den Inhalt der mit dem rapidM2M Toolset gelieferten Datei "app_connector.txt", die sich im Unterordner "examples" des Installationsverzeichnisses des rapidM2M Toolset befindet, in das Eingabefenster zum Konfigurieren des myDatanet Connector . Klicken Sie anschließend auf den Button "speichern", der sich ganz unten in der Eingabemaske zur Konfiguration der Messstelle / Applikation befindet. Dadurch wird die aktuelle Eingabemaske geschlossen und Sie gelangen zurück zur Liste der Messstellen / Applikationen.

    Eine detaillierte Erklärung finden Sie im Kapitel "Erklärung der in der Datei "app_connector.txt" enthaltenen Connector-Definition" auf Seite 50.




Eingabemaske zur Konfiguration der Messstelle / Applikation

| | |
|---|---|
| 1 | Konfigurationsabschnitt zur Auswahl des Script Typs, der Script Quelle und der Angabe der Datenstruktur (myDatanet Connector ). Detaillierte Informationen finden Sie im Kapitel "Steuerung" auf Seite 193. |
| 2 | Eingabefenster zum Konfigurieren des myDatanet Connector (siehe "myDatanet Connector " auf Seite 283) |
| 3 | Konfigurationsabschnitt zur Anzeige bzw. Veränderung der Parameter des Konfigurationsspeicherblocks 0. Solange dem Cloud-Server der Aufbau des Konfigurationsspeicherblocks nicht bekannt ist (myDatanet Connector nicht definiert), ist dieser Abschnitt leer. |

19. Da nun die Daten mittels Split-tag beim Empfang in den Messdatenkanal 0 ("histdata0") kopiert werden (siehe "Split-tag" auf Seite 284) und der Cloud-Server die Struktur der vom Script "app.p" erzeugten Messdaten kennt (siehe "Datenstruktur" auf Seite 286), kann der zur Verfügung stehende Datenzeitraum korrekt angezeigt werden.

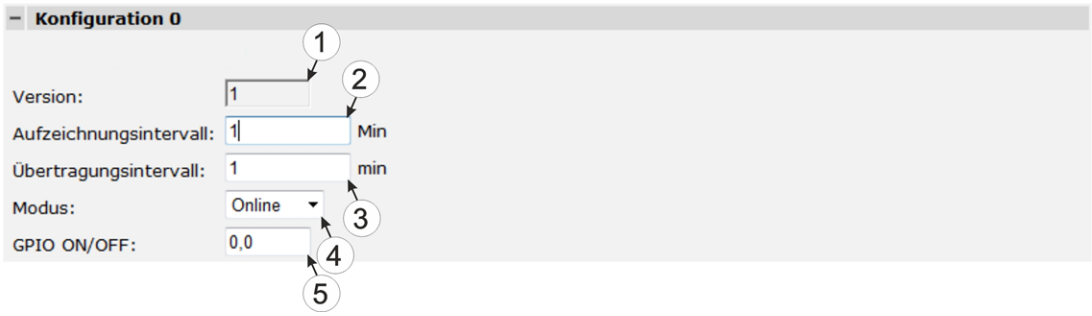



Liste der Messstellen / Applikationen

| | |
|---|---|
| 1 | Anzeige des zur Verfügung stehenden Datenzeitraums |

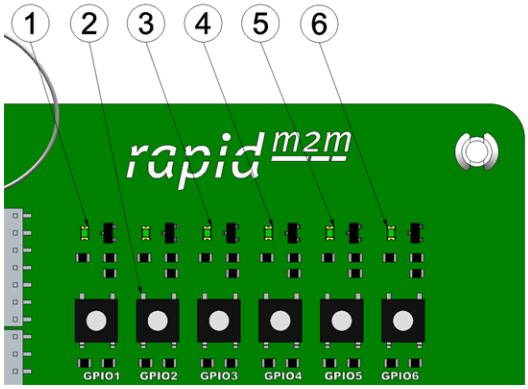
Der folgende Schritt ist nicht zwingend erforderlich.

20. Öffnen Sie die Eingabemaske zur Konfiguration der Messstelle / Applikation erneut und testen Sie die Auswirkungen von Konfigurationsänderungen.




Eingabemaske zur Konfiguration der Messstelle / Applikation, Abschnitt "Konfiguration 0"

| | | |
|---|---|---|
| 1 | Anzeige der aktuellen Version der Konfiguration (Read Only). Wird durch das Script vorgegeben und sollte geändert werden, wenn sich die Struktur der Konfiguration ändert (z.B. durch einen weiteren Parameter). | |
| 2 | zeitlicher Abstand der Messdatenaufzeichnungen | |
| 3 | zeitlicher Abstand der Übertragungen | |
| 4 | Intervall | Das Gerät meldet sich im Übertragungsintervall. |
| | Wakeup | Das Gerät meldet sich im Übertragungsintervall und kann über den Server in den ALOHA-Übertragungsmodus versetzt werden.  |
| | Online | Das Gerät trennt die Verbindung nicht und übermittelt kontinuierlich die Messdaten. |
| 5 | Ermöglicht das gezielte Ein- bzw. Ausschalten der LEDs.<br><br>Bit0: LED202 (GPIO3); Bit gesetzt -> LED ein<br>Bit1: LED203 (GPIO4); Bit gesetzt -> LED ein<br>Bit2: LED204 (GPIO5); Bit gesetzt -> LED ein<br>Bit3: LED205 (GPIO6); Bit gesetzt -> LED ein<br><br>Wird dieser Parameter auf "8" gesetzt, leuchtet das LED205 (GPIO6). | |




Verwendete Bedienelemente

| | |
|---|---|
| 1 | LED200 (GPIO1): "alive" Signal (blinkt im Sekundentakt) |
| 2 | Taste S201 (GPIO2): Messung sofort auslösen |
| 3 | LED202 (GPIO3): per Konfigurationsparameter ein-/ausschaltbar |
| 4 | LED203 (GPIO4): per Konfigurationsparameter ein-/ausschaltbar |
| 5 | LED204 (GPIO5): per Konfigurationsparameter ein-/ausschaltbar |
| 6 | LED205 (GPIO6): per Konfigurationsparameter ein-/ausschaltbar |

21. Legen Sie zur Darstellung der Messwerte eine Auswertung an (siehe "Auswertungen" auf Seite 121).



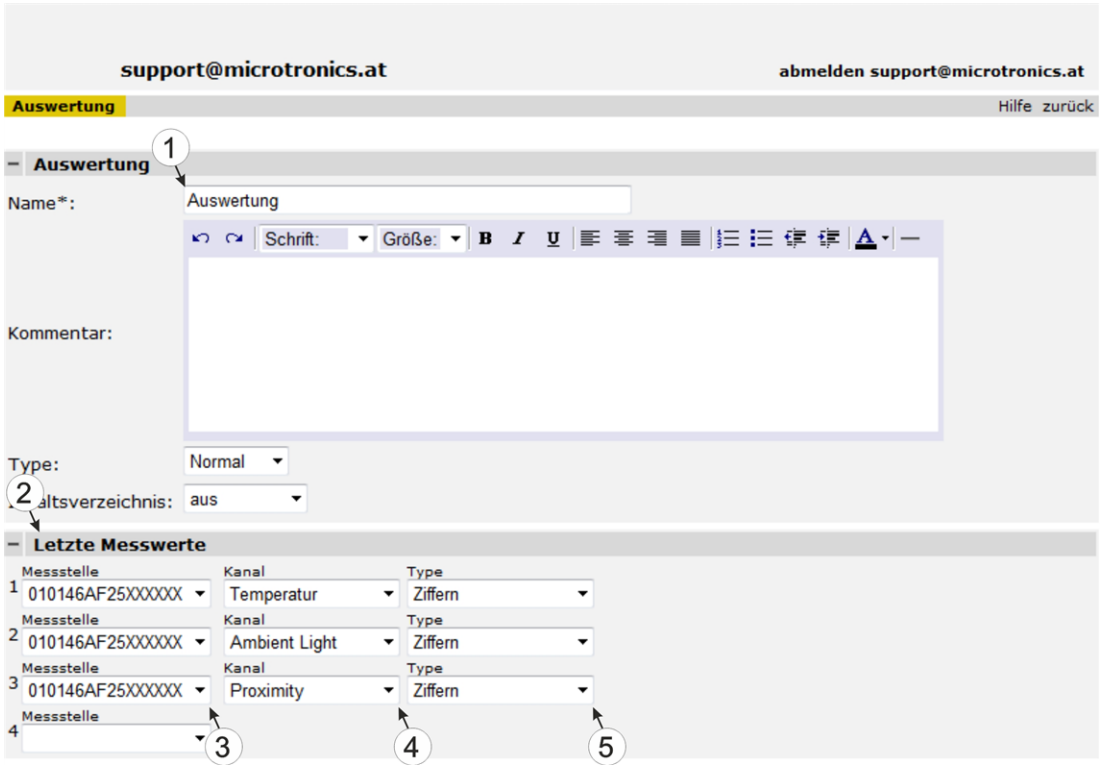

Bereich "Messstellen / Applikationen", in dem alle zur Verfügung stehenden Messstellen angezeigt werden

| | | | |
|---|---|---|---|
| 1 | fügt eine neue Auswertung hinzu | 2 | Liste der Auswertungen |

22. Konfigurieren Sie die Anzeigeelemente "letzte Messwerte" wie folgt. Dadurch werden die letzten Messwerte der entsprechenden Messkanäle in der Liste der Auswertungen neben dem Namen der Auswertung angezeigt. Durch Klicken des Buttons "speichern" gelangen Sie wieder zum Bereich "Messstellen / Applikationen" in dem alle zur Verfügung stehenden Messstellen angezeigt werden.

    Eine detaillierte Anleitung zu den Auswertungen finden Sie im Benutzerhandbuch für myDataweb-Server (206.886).

| | Messstelle | Kanal | Typ |
|---|---|---|---|
| 1 | {Seriennummer} | Temperature | Ziffern |
| 2 | {Seriennummer} | Ambient Light | Ziffern |
| 3 | {Seriennummer} | Proximity | Ziffern |



Eingabemaske zur Grundkonfiguration der Auswertung

| | |
|---|---|
| 1 | frei wählbarer Name für die Auswertung |
| 2 | Konfiguration der Anzeigeelemente "letzte Messwerte", die in der Liste der Auswertungen neben dem Namen der Auswertung angezeigt werden (siehe "Auswertungen" auf Seite 121). |
| 3 | Auswahl der Messstelle. Das Element kann Daten von unterschiedlichen Messstellen beziehen. Es stehen alle Messstellen des aktuellen Kunden zur Verfügung. |
| 4 | Kanal der Messstelle, der angezeigt werden soll |
| 5 | Auswahl des Typs des zu verwendenden Anzeigeelements |

Der folgende Schritt ist nicht zwingend erforderlich.

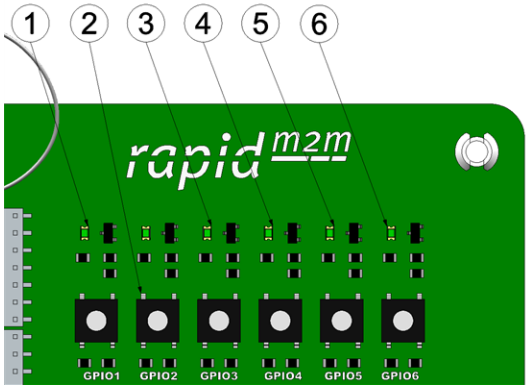
23. Direkt neben dem Namen der Auswertung werden nun die aktuellen Messwerte angezeigt. Vorausgesetzt Sie haben die Standardkonfiguration nicht verändert (Aufzeichnungsintervall: 1min., Modus: Online), werden die Messwerte alle 60sec. aktualisiert. Sie können aber auch durch Drücken der Taste S201, die mit GPIO2 verbunden ist, eine sofortige Messung auslösen. Um eine deutliche Veränderung der Messwerte zu erreichen, decken Sie den Umgebungslicht- und IR-Näherungssensor MAX44000 (U301) ab oder berühren (nicht empfohlen) Sie den Temperatursensor TMP112 (U300).

    ***Wichtiger Hinweis:*** *Bitte beachten Sie, dass die empfindlichen elektronischen Komponenten des rapidM2M Developer Paket M12x bei Berührung durch statische Elektrizität beschädigt werden können.*

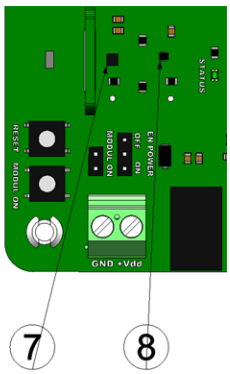



Bereich "Messstellen / Applikationen", in dem alle zur Verfügung stehenden Messstellen angezeigt werden

| | |
|---|---|
| 1 | öffnet die Eingabemaske zur Grundkonfiguration der Auswertung |
| 2 | Durch Klicken auf den Namen der Auswertung wird die Eingabemaske zur Konfiguration und der Auswahl der Elemente geöffnet. |
| 3 | Anzeigeelemente für den letzten Messwert eines Messkanals (max. 4 pro Auswertung). Die Auswahl der Messstelle, des Kanals und des Typs des Anzeigeelements erfolgt in der Grundkonfiguration der Auswertung (siehe "Grundkonfiguration" auf Seite 122). |



Verwendete Bedienelemente



Verwendete Sensoren

| | |
|---|---|
| 1 | LED200 (GPIO1): "alive" Signal (blinkt im Sekundentakt) |
| 2 | Taste S201 (GPIO2): Messung sofort auslösen |
| 3 | LED202 (GPIO3): per Konfigurationsparameter ein-/ausschaltbar |
| 4 | LED203 (GPIO4): per Konfigurationsparameter ein-/ausschaltbar |
| 5 | LED204 (GPIO5): per Konfigurationsparameter ein-/ausschaltbar |
| 6 | LED205 (GPIO6): per Konfigurationsparameter ein-/ausschaltbar |
| 7 | Temperatursensor TMP112 (U300): Messwert "Temperatur [°C]" |
| 8 | Umgebungslicht- und IR-Näherungssensor MAX44000 (U301): Messwerte "Umgebungslicht [lux]" und "Entfernung [%]" |

# 4.4 Erklärung der rapidM2M spezifischen Codeblöcke des Scripts "app.p"

## 4.4.1 Initialisieren der Sensoren

Initialisiert die I²C-Schnittstelle des M120 Moduls und den Temperatursensor TMP112 (U300) sowie den Umgebungslicht- und IR-Näherungssensor MAX44000 (U301), die sich am rapidM2M Developer Board befinden. Zudem werden Debug-Informationen mittels der Funktionen "print" und "printf" (siehe "Consolen Funktionen" auf Seite 201) auf die Standardausgabe gedruckt, die über den Karteireiter "Console" (siehe "Karteireiter "Console"" auf Seite 151) des rapidM2M Toolset ausgewertet werden können.

```
/* init I2C #0 with 400kHz clock */
iResult = rM2M_I2cInit(0, 400000, 0);
printf("rM2M_I2cInit() = %d\r\n", iResult);

/* init TMP112 sensor */
if(TMP112_Init(TMP112_I2C_ADR1) < OK)
{
  print("TMP112 Error\r\n");
}
else print("TMP112 OK\r\n");

/* init MAX44000 sensor */
if(MAX44000_Init(MAX44000_I2C_ADR) < OK)
{
  print("MAX44000 Error\r\n");
}
else print("MAX44000 OK\r\n");
```

### 4.4.2 Initialisieren der Timer

Initialisiert den 1sec. Timer, der für die Erzeugung des Aufzeichnungs- und Übertragungsintervalls verwendet wird und legt die Callback-Funktion fest, die beim Ablauf des Timers aufgerufen wird. Die Callback-Funktion löst bei Ablauf des Übertragungsintervalls die Verbindung zum Cloud-Server aus. Ebenso führt sie bei Ablauf des Aufzeichnungsintervalls die Aufzeichnung durch und gibt die aktuellen Messwerte mittels der Funktion "printf" (siehe "Consolen Funktionen" auf Seite 201) über die Standardausgabe aus.

```
/* init 1s Timer, used for general program flow */
iIdx = funcidx("Timer1s");
iResult = rM2M_TimerAdd(iIdx);
printf("rM2M_TimerAdd(%d) = %d\r\n", iIdx, iResult);
```

Initialisiert einen 100ms Timer mit Autoreload. Dieser wird für das Lesen der Werte von den Sensoren (TMP112 (U300) und MAX44000 (U301)) verwendet. Zusätzlich wird der Timer für die Überwachung, ob die Taste S201 (GPIO2) zum Auslösen einer sofortigen Aufzeichnung gedrückt wurde, genutzt. Zudem wird die Callback-Funktion festgelet, die beim Ablauf des Timers aufgerufen wird.

```
/* init 100ms Timer, used for sensor sampling */
iIdx = funcidx("Timer100ms");
iResult = rM2M_TimerAddExt(iIdx, true, 100);
printf("rM2M_TimerAddExt(%d) = %d\r\n", iIdx, iResult);
```

### 4.4.3 Callback für die Konfigurationsbehandlung festlegen

Legt die Callback-Funktion fest, die aufgerufen werden soll, wenn sich einer der Konfigurationsspeicherblöcke geändert hat.

```
/* set config change callback */
iIdx = funcidx("ReadConfig");
iResult = rM2M_CfgOnChg(iIdx);
printf("rM2M_CfgOnChg(%d) = %d\r\n", iIdx, iResult);
```

### 4.4.4 Verbindungsart setzen

Setzt die in der Konfiguration gespeicherte Verbindungsart. Standardmäßig ist im Scrpit "app.p" der Modus "online" hinterlegt. Daher wird die Verbindung zum Cloud-Server sofort an dieser Codestelle aufgebaut.

```
/* set communication mode */
rM2M_TxSetMode(iTxMode);
```

### 4.4.5 Messdaten speichern

Die Funktion "rM2M_GetTime()" liest den aktuellen Zeitstempel aus (Sekunden seit 31.12.1999).

> ***Hinweis:*** *Dieser wird nur als weiterer Messwert für die Aufzeichnung verwendet. Er wird nicht als Zeitstempel für den Datensatz verwendet. Die Funktion "rM2M_RecData()" erzeugt den Zeitstempel für den Datensatz selbständig, da der erste Parameter "0" ist.*

Durch den mehrfachen Aufruf der Funktion "rM2M_SetPacked()" wird der Datenbereich des Messdatensatzes zusammengestellt. Aus diesem Aufbau ergibt sich auch die Definition des myDatanet Connector (siehe "Messdatenkanal 0" auf Seite 51). Der Funktion "rM2M_RecData()" wird danach dieser Datenbereich zur Aufzeichnung übergeben.

```
/* get actual time */
iTimestamp = rM2M_GetTime();

/* build record data */
rM2M_SetPacked(aRecData,  0, iCount,     4, true);
rM2M_SetPacked(aRecData,  4, iTimestamp, 4, true);
rM2M_SetPacked(aRecData,  8, fTemp,      4, true);
rM2M_SetPacked(aRecData, 12, fALS,       4, true);
rM2M_SetPacked(aRecData, 16, iProx,      4, true);

/* record data with actual timestamp */
rM2M_RecData(0, aRecData, ciRecDataSize);
```

### 4.4.6 Überwachung der Taste S201 zum Auslösen einer sofortigen Aufzeichnung

Überprüft zyklisch den Zustand der Taste S201 (GPIO2). Falls ein Tastendruck erkannt wird, wird der Zähler für die verbleibende Zeit bis zur nächsten Aufzeichnung auf 0 gesetzt, wodurch die Aufzeichnung unverzüglich ausgelöst wird.

```
/* poll gpio #1 for user interaction */
iGpio = rM2M_GpioGet(1);
iAction = (iGpio_old ^ iGpio) & iGpio;

if(iAction & 0x1)
{
  /* record now */
  iRecTimer = 0;
  print("Gpio1 Record now.\r\n");
}
iGpio_old = iGpio;
```

### 4.4.7 Lesen der Sensorwerte

```
/* read actual temperature from TMP112 sensor */
if(TMP112_Read(TMP112_I2C_ADR1, TMP112_TEMP_REG, iTemp) == OK)
{
  /* calculate °C */
  fTemp = iTemp/256.0;
}

/* read ambient light value from MAX44000 sensor */
if(MAX44000_Read(MAX44000_I2C_ADR, MAX44000_ALS_REG, iALS) == OK)
{
  /* check overflow bit */
  if(iALS & 0x4000) iALS = 0x3FFF;

  /* calculate lux (0.5lux/LSB) */
  fALS = iALS * 0.5;
}

/* read proximity value from MAX44000 sensor */
if(MAX44000_Read(MAX44000_I2C_ADR, MAX44000_PROX_REG, iProx) == OK)
{
  /* calculate proximity as [%] */
  iProx = (iProx * 100) / 0x4000;
}
```

### 4.4.8 Lesen der Konfiguration

```
/* read config */
iSize = rM2M_CfgRead(cfg, 0, aData, CFG_SIZE);
printf("Cfg %d size = %d\r\n", cfg, iSize);
```

Zuerst wird überprüft, ob beim Lesen des Konfigurationsblocks mittels der Funktion "rM2M_CfgRead()" zumindest 1 Byte gelesen werden konnte. Im ersten Byte befindet sich beim Scrpit "app.p" die Versionsnummer der Konfiguration. Sollte sich die Versionsnummer geändert haben (z.B. durch einen weiteren Parameter) oder die Konfiguration nicht die erwartete Größe besitzen, muss eine Defautkonfiguration erzeugt und gespeichert werden. Durch den mehrfachen Aufruf der Funktion "rM2M_SetPacked()" wird der Datenbereich des Konfigurationsblocks zusammengestellt. Aus diesem Aufbau ergibt sich auch die Definition des myDatanet Connector (siehe "Konfigurationsspeicherblock 0" auf Seite 52). Der Funktion "rM2M_CfgWrite()" wird danach dieser Datenbereich zur Speicherung übergeben.

```
/* Check Version of Config */
if(iSize > 0)
{
  rM2M_GetPacked(aData, 0, iVersion, 1, true);
}

/* check if config object #0 is available */
if(iSize < CFG_SIZE || iVersion != CFG_VERSION)
{
  /* create default config */
  iRecItv = ITV_RECORD;
  iTxItv = ITV_TRANSMISSION;
  iTxMode = TXMODE;
  iGpioOut = GPIOOUT;
  iVersion = CFG_VERSION;

  rM2M_SetPacked(aData,  0, iVersion,  1, true);
  rM2M_SetPacked(aData,  1, iRecItv,   4, true);
  rM2M_SetPacked(aData,  5, iTxItv,    4, true);
  rM2M_SetPacked(aData,  9, iTxMode,   1, true);
  rM2M_SetPacked(aData, 10, iGpioOut,  1, true);

  /* write to config #0 */
  rM2M_CfgWrite(0, 0, aData, CFG_SIZE);
  iSize = CFG_SIZE;
  print("created new Config #0\r\n");
}
```

Schaltet die LED202 - LED205 (GPIO3 - GPIO5) entsprechend der über die Oberfläche des Cloud-Servers eingegebenen Konfiguration ein bzw. aus.

```
/* set gpio outputs according to config */
rM2M_GpioSet(2, iGpioOut & 1);
rM2M_GpioSet(3, (iGpioOut>>1) & 1);
rM2M_GpioSet(4, (iGpioOut>>2) & 1);
rM2M_GpioSet(5, (iGpioOut>>3) & 1);
```

# 4.5 Erklärung der in der Datei "app_connector.txt" enthaltenen Connector-Definition

## 4.5.1 Split-tag

| Code | Erklärung |
|---|---|
| *<split>*<br>*source =rm2mraw*<br>*target =rm2mdata*<br>*key =**<br>*</split>* | Beginn des Split-tags<br>Die Daten aus dem Rohmessdatenbereich sollen kopiert werden.<br>Ziel für die zu kopierenden Daten ist der Messdatenkanal o[1)].<br>"*" bedeutet, dass alle Datensätze von source nach target kopiert werden.<br>Ende des Split-tags |

[1)] "rm2mdata" ist gleichbedeutend mit "histdatao". Aus Gründen der Abwärtskompatibilität wird "rm2mdata" zwar weiter unterstützt, es sollte aber "histdatao" verwendet werden.

Detaillierte Informationen finden Sie im Kapitel "Split-tag" auf Seite 284.

## 4.5.2 Messdatenkanal 0

Der Datenbereich des Messdatensatzes ist wie folgt aufgebaut:

```
rM2M_SetPacked(aRecData,  0, iCount,     4, true);
rM2M_SetPacked(aRecData,  4, iTimestamp, 4, true);
rM2M_SetPacked(aRecData,  8, fTemp,      4, true);
rM2M_SetPacked(aRecData, 12, fALS,       4, true);
rM2M_SetPacked(aRecData, 16, iProx,      4, true);
```

Daher muss der myDatanet Connector wie folgt definiert werden:

| Code | Erklärung |
|---|---|
| `<table>` | Beginn des Table-Tags |
| `  name   =rm2mdata` | Die folgende Beschreibung gilt für Messdatenkanal 0[1]. |
| `  <field>` | Beginn einer Feld-Definition für dem Messwert "Count" |
| `    name   =ch0` | Eine Beschreibung der Attribute finden Sie unter |
| `    title  =Count` | "Attribute der Feld-Definition" auf Seite 287. |
| `    type   =u32` | |
| `    byteofs=0` | |
| `  </field>` | |
| `  <field>` | Beginn einer Feld-Definition für dem Messwert "Timestamp" |
| `    name   =ch1` | |
| `   title  =Timestamp` | |
| `   type   =u32` | |
| `   byteofs=4` | |
| `  </field>` | |
| `  <field>` | Beginn einer Feld-Definition für dem Messwert "Temperatur" |
| `   name   =ch2` | |
| `   title  =Temperatur` | |
| `   type   =f32` | |
| `   byteofs=8` | |
| `  </field>` | |
| `  <field>` | Beginn einer Feld-Definition für dem Messwert "Ambient Light" |
| `   name   =ch3` | |
| `   title  =Ambient Light` | |
| `   type   =f32` | |
| `   byteofs=12` | |
| `  </field>` | |
| `  <field>` | Beginn einer Feld-Definition für dem Messwert "Proximity" |
| `   name   =ch4` | |
| `   title  =Proximity` | |
| `   type   =u32` | |
| `   byteofs=16` | |
| `  </field>` | |
| `</table>` | |

[1] "rm2mdata" ist gleichbedeutend mit "histdatao". Aus Gründen der Abwärtskompatibilität wird "rm2mdata" zwar weiter unterstützt, es sollte aber "histdatao" verwendet werden.

Detaillierte Informationen finden Sie im Kapitel "Datenstruktur" auf Seite 286.

## 4.5.3 Konfigurationsspeicherblock 0

Der Datenbereich des Konfigurationsblocks wie folgt aufgebaut ist:

```
rM2M_SetPacked(aData,  0, iVersion,  1, true);
rM2M_SetPacked(aData,  1, iRecItv,   4, true);
rM2M_SetPacked(aData,  5, iTxItv,    4, true);
rM2M_SetPacked(aData,  9, iTxMode,   1, true);
rM2M_SetPacked(aData, 10, iGpioOut,  1, true);
```

muss der myDatanet Connector  wie folgt definiert werden:

| Code | Erklärung |
|---|---|
| `<table>` | Beginn des Table-Tags |
| `  name   =rm2mconfig0` | Die folgende Beschreibung gilt für die Konfiguration 0 [1]. |
| `  <field>` | Beginn einer Feld-Definition für dem Parameter "Version" |
| `    name   =field0` | Eine Beschreibung der Attribute finden Sie unter |
| `    title  =Version` | "Attribute der Feld-Definition" auf Seite 287. |
| `    decpl  =0` | |
| `    type   =u8` | |
| `    units  =` | |
| `    byteofs=0` | |
| `    edit   =8` | |
| `  </field>` | |
| `  <field>` | Beginn einer Feld-Definition für dem Messwert |
| `    name   =field1` | "Aufzeichnungsintervall" |
| `    title  =Aufzeichnungsintervall` | |
| `    units  =Min` | |
| `    decpl  =0` | |
| `    vscale =0.01666666667` | |
| `    type   =u32` | |
| `    byteofs=1` | |
| `  </field>` | |
| `  <field>` | Beginn einer Feld-Definition für dem Messwert |
| `    name   =field2` | "Übertragungsintervall" |
| `    title  =Übertragungsintervall` | |
| `    units  =min` | |
| `    decpl  =0` | |
| `    vscale =0.01666666667` | |
| `    type   =u32` | |
| `    byteofs=5` | |
| `  </field>` | |
| `  <field>` | Beginn einer Feld-Definition für dem Messwert "Modus" |
| `    name   =field3` | |
| `    title  =Modus` | |
| `    editmask=0=Intervall;1=Wakeup;2=Online` | |
| `    type   =u8` | |
| `    byteofs=9` | |
| `  </field>` | |
| `  <field>` | Beginn einer Feld-Definition für dem Messwert "Proximity" |
| `    name   =field4` | |
| `    title  =GPIO ON/OFF` | |
| `    type   =u8` | |
| `    byteofs=10` | |
| `  </field>` | |
| `</table>` | |

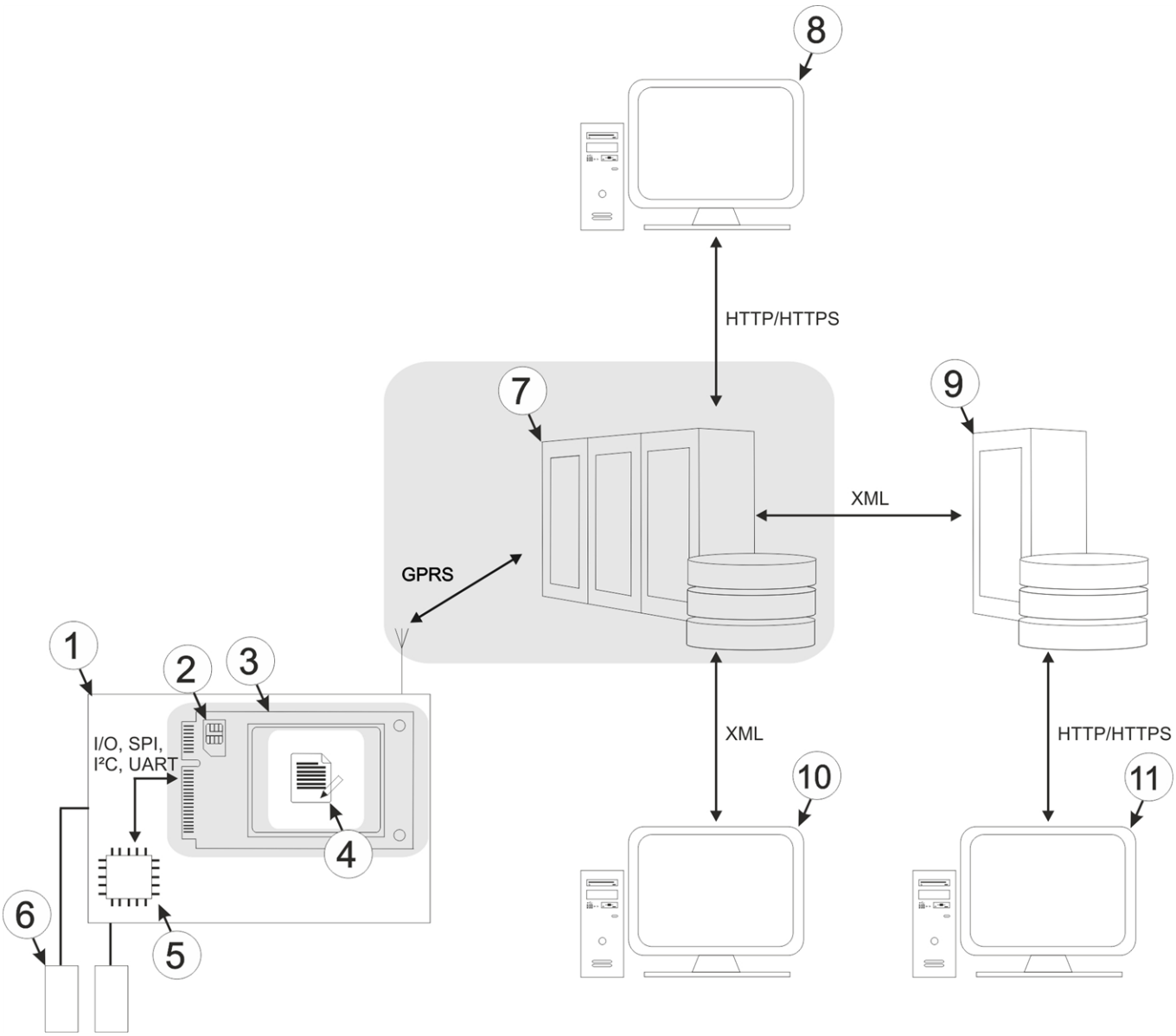
[1] "rm2mconfigo" ist gleichbedeutend mit "configo". Aus Gründen der Abwärtskompatibilität wird "rm2mconfigo" zwar weiter unterstützt, es sollte aber "configo" verwendet werden.

Detaillierte Informationen finden Sie im Kapitel "Datenstruktur" auf Seite 286.

6872
6872
6872

# Kapitel 5 Funktionsprinzip

Der Grundgedanke des rapidM2M -Moduls ist eine sogenannte Storage-2-Storage-Datenübertragung. Für diese Art der Datenübermittlung müssen sowohl das M120 Modul als auch der Cloud-Server nicht über den logischen Inhalt der Datenblöcke Bescheid wissen. Es geht also nur darum, einen Block Daten von A nach B zu transportieren. In der unten abgebildeten Grafik sind alle Komponenten, die Teil des rapidM2M sind, grau hinterlegt. Alle Komponenten, die vom Kunden bereitgestellt/erstellt werden müssen, sind schwarz dargestellt.




Funktionsprinzip des rapidM2M

| | |
|---|---|
| 1 | Gerät (kundenspezifische Hardware) mit eingesetztem M120 Modul |
| 2 | Managed Service SIM-Karte mit inkludierter Datenübertragung |
| 3 | M120 Modul (siehe "rapidM2M Modul M120 " auf Seite 99) |
| 4 | Durch den Kunden erstellte Applikation (Pawn Script), die sich um den Datenaustausch mit der Messelektronik und die Datenaufzeichnung kümmert (siehe "Pawn Script" auf Seite 197) |
| 5 | kundenspezifische Messelektronik |
| 6 | Sensor, der mit dem Gerät (kundenspezifische Hardware) verbunden ist (optional) |
| 7 | Cloud-Server, zu dem die Daten übertragen werden. |
| 8 | Client, der mittels Web-Browser auf die Oberfläche des Cloud-Server (siehe "Cloud-Server" auf Seite 117) zugreift. Dazu muss allerdings der myDatanet Connector richtig konfiguriert sein (siehe "myDatanet Connector " auf Seite 283). |
| 9 | kundenspezifischer Server, der den Clients eine eigene Oberfläche zur Verfügung stellt. Die Daten bezieht der kundenspezifische Server über die XML-Schnittstelle des Cloud-Servers (siehe "XML-Schnittstelle" auf Seite 157). |
| 10 | Client, auf dem ein PC-Programm läuft, das seine Daten über die XML-Schnittstelle des Cloud-Servers (siehe "XML-Schnittstelle" auf Seite 157) bezieht. |
| 11 | Client, der mittels Web-Browser auf die Oberfläche des kundenspezifischen Servers zugreift |

Funktionen und Komponenten, die durch rapidM2M bereitgestellt werden:

- M120 Modul:

  Programmierbares (siehe "Pawn Script" auf Seite 197) Kommunikationsmodul mit integriertem Speicher und standardisierte Mikrocontroller-Schnittstellen (siehe "Serielle Schnittstellen" auf Seite 110)

- 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.

- Cloud-Server

  Datenbank für die Speicherung der Messdaten und Konfigurationen. Der Zugriff auf die Daten erfolgt entweder über die XML-Schnittstelle des Servers (siehe "XML-Schnittstelle" auf Seite 157) oder über die Web-Oberfläche des Servers. Wenn über die Web-Oberfläche auf den Server zugegriffen werden soll muss dem Cloud-Server die Struktur der Datenblöcke bekannt gegeben werden. Dies erfolgt mittels des myDatanet Connector (siehe "myDatanet Connector " auf Seite 283).

Funktionen und Komponenten, die durch den Kunden bereitgestellt werden:

- Messelektronik:

  Elektronik zur Aufnahme des M120 Moduls und Erfassung der Messwerte. Diese kann entweder einen eigenen Messkontroller enthalten, der seine Daten über eine Schnittstelle an das M120 Modul überträgt oder nur die Sensorelektronik. Es ist möglich, die gesamte Mess-Applikation in das M120 Modul zu programmieren, wodurch auf den eigenen Messkontroller verzichtet werden kann.

- Applikation (Pawn Script):

  Die Firmeware des M120 Moduls kümmert sich lediglich um die Synchronisation der Messdaten und Konfigurationen zwischen M120 Modul und Cloud-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 M120 Moduls (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-

- Kundenspezifischer Server mit Web-Oberfläche für die Clients:

  Dadurch ist es möglich eine eigene Web-Oberfläche für die Clinets zu erstellen. Die Daten werden dabei vom kundenspezifischen Server über die XML-Schnittstelle (siehe "XML-Schnittstelle" auf Seite 157) aus dem Cloud-Server gelesen.

# 5.1 Empfohlene Vorgehensweise

## 5.1.1 Entwicklung einer M2M / IoT Anwendung

Es wird empfohlen bei der Entwicklung einer M2M / IoT Anwendung mit der Definition des myDatanet Connector (siehe "myDatanet Connector " auf Seite 283) zu beginnen. Durch ihn werden die verschiedenen Datenstrukturen (Messdaten, Konfigurationen usw. ) festgelegt, die sowohl für das Script als auch für den Cloud-Server gültig sind. Für den Zugriff auf die Daten des Cloud-Servers über die XML-Schnittstelle 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.*

## 5.2 Automatische Auswahl des GSM-Netzes

Da das rapidM2M Developer Paket M12x mit einem SIM-Chip ausgestattet ist, der eine Mobilfunkverbindung über eine Vielzahl internationaler Serviceprovider gewährleistet (siehe www.microtronics.at/footprint ), ist eine Auswahl des GSM-Netzes, in das sich das Gerät einbuchen soll, erforderlich. Diese erfolgt automatisch vom Gerät. Dabei wird auch berücksichtigt, wenn die Verbindung über ein bestimmtes GSM-Netz zwar möglich ist, es aber bei Verwendung dieses GSM-Netzes immer wieder zu Übertragungsfehlern kommt. Ein solches GSM-Netz wird in eine "Banlist" eingetragen und beim nächsten Einbuchungsversuch ignoriert.

Folgende Bedingungen führen dazu, dass sich das rapidM2M Developer Paket M12x in ein GSM-Netz einbucht:

- PowerOn (nur falls kein Script vorhanden ist bzw. das Script aufgrund eines Fehlers deaktiviert wurde)
- Auslösen der Übertragung, wenn die Verbindungsart „Intervall“ (siehe "rM2M_TxSetMode()") ausgewählt wurde
- Aktivieren der Verbindungsart „Intervall & Wakeup“ oder „Online“ (siehe "rM2M_ TxSetMode()")

1. Das rapidM2M Developer Paket M12x überprüft, ob es beim letzten Einbuchungsversuch zu Fehlern kam.

   Keine Fehler:

   Das rapidM2M Developer Paket M12x bucht sich in dasselbe GSM-Netz ein, das auch beim letzten Versuch verwendet wurde.

   Fehler:

   Weiter mit Schritt 2

   > ***Hinweis:*** *Ein Übertragungsfehler ist nicht gleichbedeutend mit einem Fehler beim Einbuchen in das GSM-Netzwerk. Es ist möglich, dass das Einbuchen in das GSM-Netzwerk korrekt funktioniert hat, aber es dennoch zu einem Übertragungsfehler kommt.*

Die folgenden Schritte werden nur ausgeführt, wenn bei Schritt 1 ein Fehler erkannt wurde:

2. Das Gerät sucht nach GSM-Netzen in seinem Empfangsbereich und erstellt eine Liste der verfügbaren GSM-Netze.
3. Die Liste mit den verfügbaren GSM-Netzen wird durchsucht und das stärkste GSM-Netz gewählt, das sich nicht auf der "Banlist" befindet und zu einem der vom integrierten SIM-Chip unterstützten Serviceprovider (siehe www.microtronics.at/footprint ) gehört.

   > ***Hinweis:*** *Eine Erklärung dazu, unter welchen Bedingungen ein GSM-Netz in die "Banlist" eingetragen wird, finden Sie unter "Banlist" auf Seite 60.*

### 5.2.1 Banlist

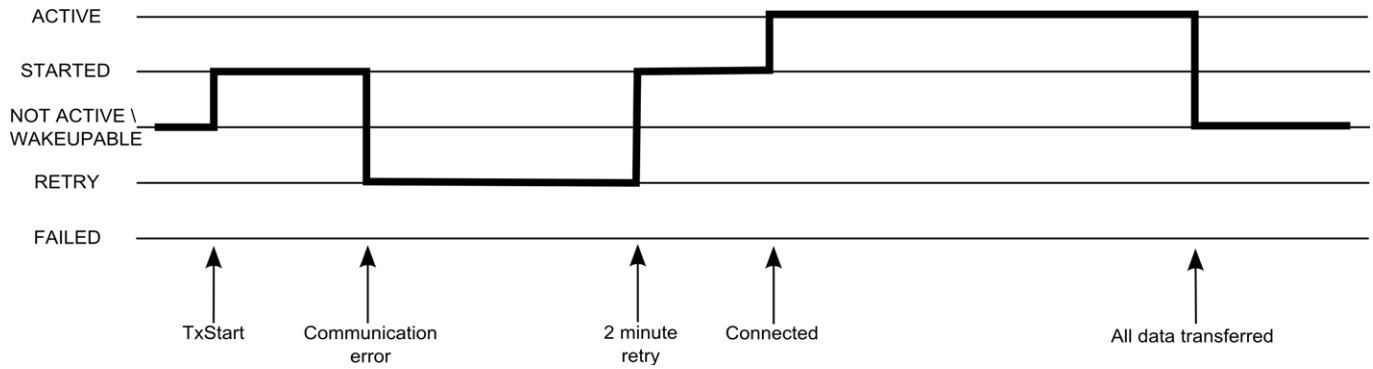
Die Banlist bietet Platz für 5 Einträge und wird nicht flüchtig gespeichert. D.h. sie steht auch nach einem PowerOn weiter zur Verfügung.

Kommt es zu einen GPRS-Fehler (siehe Fehlercodes -975 bis -993 im Kapitel "Modemfehler" auf Seite 298), wird ein interner Zähler um 2 erhöht. Wurde eine Übertragung fehlerfrei abgeschlossen, d.h. alle Daten wurden übertragen, wird dieser interne Zähler um 1 reduziert. Erreicht der interne Zähler den Wert 6 , wird das aktuelle GSM-Netzwerk in die Banlist eingetragen. Beim Auftreten des "MODEM CMME ERROR"-Fehlers (siehe "Modemfehler" auf Seite 298) mit dem Parameter 32 (NETWORK NOT ALLOWED) oder 529 (MODEM CMME ERROR) wird das entsprechende GSM-Netzwerk sofort auf die Banlist gesetzt. Die Banlist wird gelöscht, wenn aus der Liste der verfügbaren GSM-Netze im Empfangsbereich unter Berücksichtigung der Banlist kein GSM-Netzwerk ausgewählt werden kann. Nach dem Löschen der Banlist wird erneut nach GSM-Netzwerken im Empfangsbereich gesucht und die Netzauswahl mit der jetzt leeren Banlist durchgeführt (siehe Schritt 3 im Kapitel "Automatische Auswahl des GSM-Netzes" auf Seite 59).

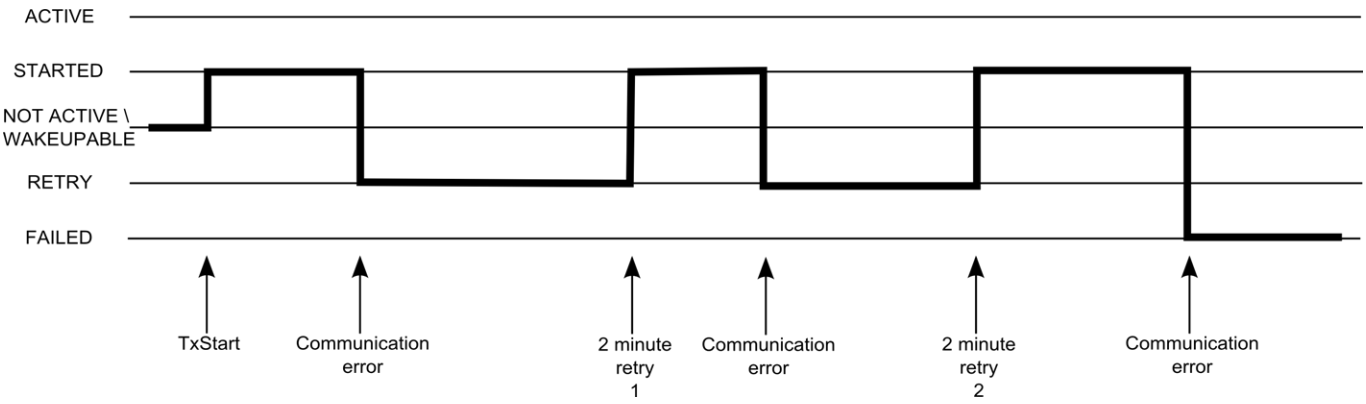
Zusätzlich zum Aufbau der GPRS-Verbindung wird auch die Qualität der Datenübertragung überwacht. Schlägt die Datenübertragung trotz bestehenden GPRS-Verbindung 10 mal in Folge fehl, wird das aktuell gewählte GSM-Netz ebenfalls auf die Banlist gesetzt. Auch hier gilt, sollte unter Berücksichtigung der Banlist kein weiteres GSM-Netz mehr zur Verfügung stehen, wird die Banlist gelöscht und die Auswahl beginnt erneut beim stärksten GSM-Netz.

## 5.3 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 Cloud-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 Cloud-Server ausgelöst werden (siehe "Messstellen" auf Seite 123).*

RETRY
*Das System warten 2min bis zum erneuten Verbindungsaufbauversuch.*

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

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

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

### 5.3.1 Verbindungsabbruch im "online"-Modus

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

### 5.3.2 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.

## 5.4 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 Cloud-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 Cloud-Servers möglich ist (siehe "Messstellen" auf Seite 123).

# 5.5 Registrierungsspeicherblöcke

Diese 4 Blöcke mit je 1kB bieten die Möglichkeit gerätespezifische Daten abzulegen und mit dem Cloud-Server zu synchronisieren. Die Blöcke unterscheiden sich nur hinsichtlich ihrer Zugriffsmöglichkeiten. Daraus ergeben sich vordefinierte Verwendungszwecke, die in der folgenden Tabelle beschrieben werden:

| Speicherblock | Zugriff | Verwendungszweck |
|---|---|---|
| Systemspezifische Daten | | |
| REG_SYS_OTP [1)] | **lesbar:** Script, Cloud-Server | Systeminformationen, die einmalig im Zuge der Produktion geschrieben werden |
| REG_SYS_FLASH [1)] | **lesbar:** Script, Cloud-Server | Systeminformationen, die im Betrieb veränderbar sein müssen |
| Applikationsspezifische Daten | | |
| REG_APP_OTP | **lesbar:** Script, Cloud-Server<br>**schreibbar:** Script | 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, Cloud-Server<br>**schreibbar:** Script | applikationsspezifische Informationen, die im Betrieb veränderbar sein müssen |

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

> ***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 313) erforderlich.*

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

## 5.5.1 REG_SYS_FLASH

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

**latestAppVersion**
*Die aktuellste verfügbare Version der im Gerät installierten Applikation [16-Bit Unsigned]*

**installedAppVersion**
*Momentan im Gerät installierte Version der Applikation [16-Bit Unsigned]*

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

### 5.5.2 REG_APP_OTP

Durch Speichern des "Product Identity Profile" (PIP) in diesem Registrierungsspeicherblock können am Cloud-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].*

**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 Cloud-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 Cloud-Server gefunden wurden und "pipCtxAutocreate=1" bzw. das Feld nicht vorhanden ist, wird die Messstelle neu angelegt.

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

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

## 5.6 File Transfer

Es ist möglich beliebig viele Dateien für den File Transfer zu registrieren (siehe "FT_Register()"). Dabei muss der Funktion "FT_Register()" eine Callback Funktion (siehe "Callback Funktionen" auf Seite 242) 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 241) zu behandeln. Das mit dem rapidM2M Toolset gelieferte Script "t.b.d." enthält ein Beispiel für die Behandlung der File Transfer Kommandos. 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.

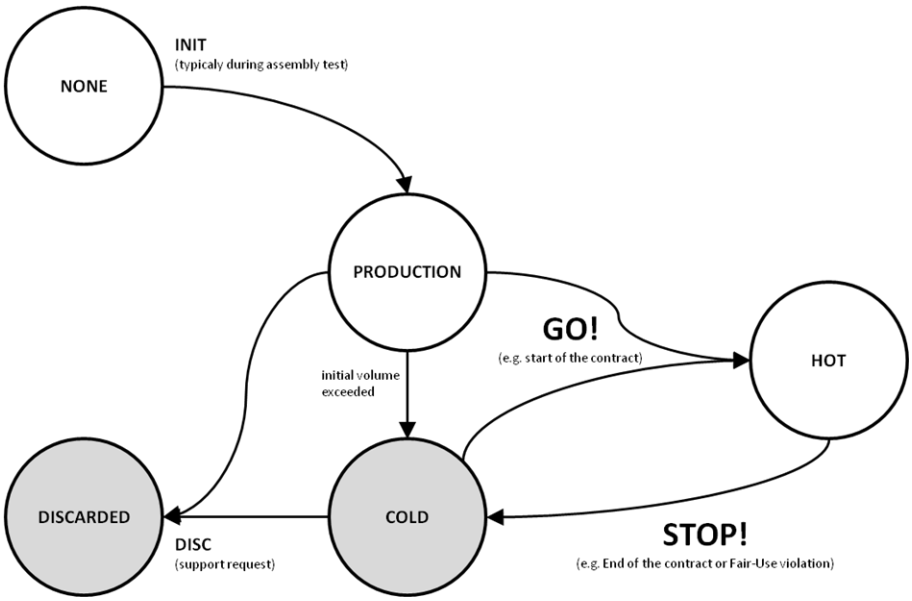
Beim Empfang eines File Transfer Kommandos wird eine Sitzung gestartet, welche nach 15sec. automatisch beendet wird, wenn das Script das Kommando nicht korrekt behandelt. Um Konflikte zu vermeiden, kann immer nur eine Sitzung nach der anderen aktiv sein.

## 5.7 Bedeutung des SIM-Status

Das Gerät erhält vom Cloud-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 313) 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.

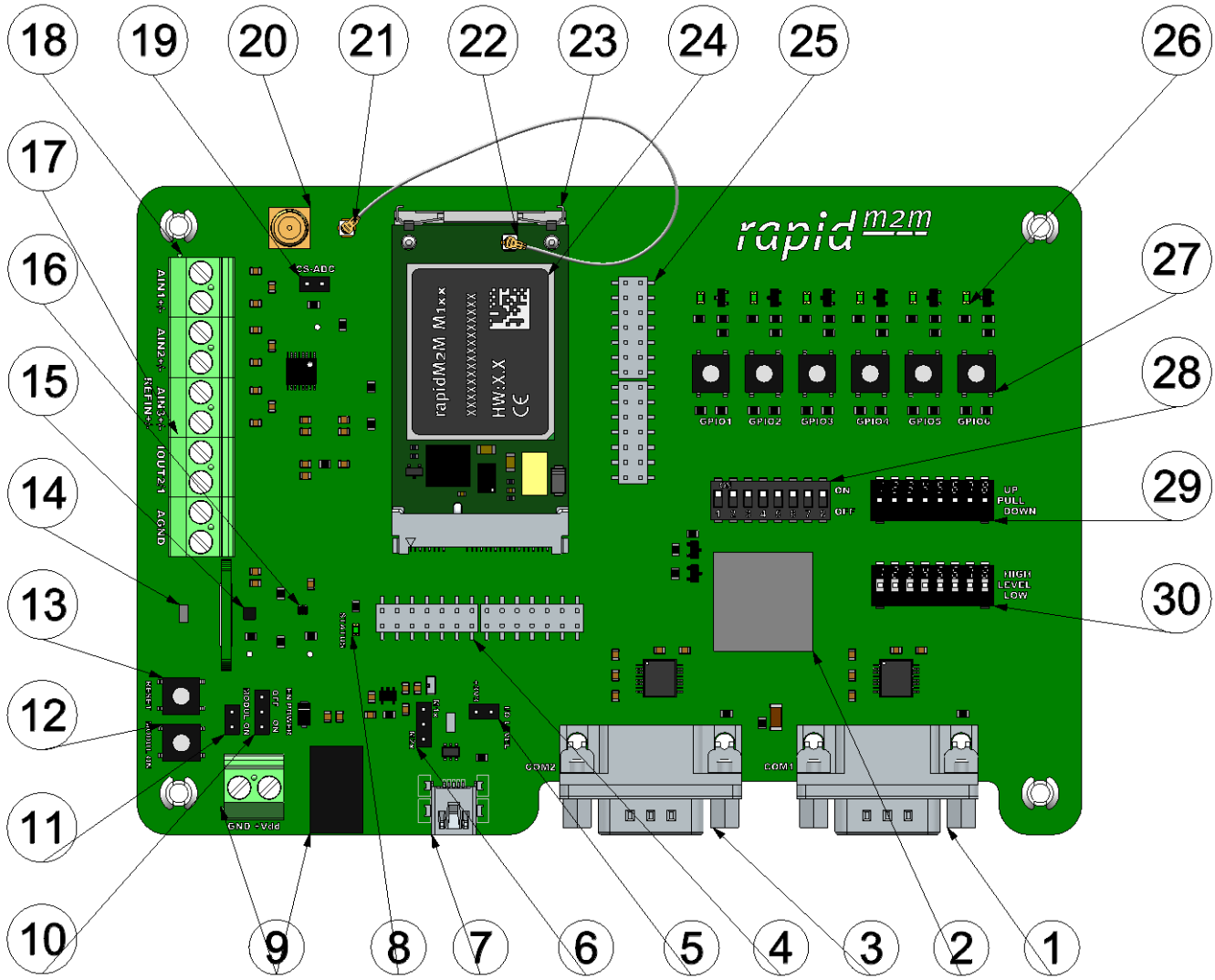



Statediagramm der SIM-Zustände

6872
6872
6872

# Kapitel 6 rapidM2M Developer Board

## 6.1 Übersicht



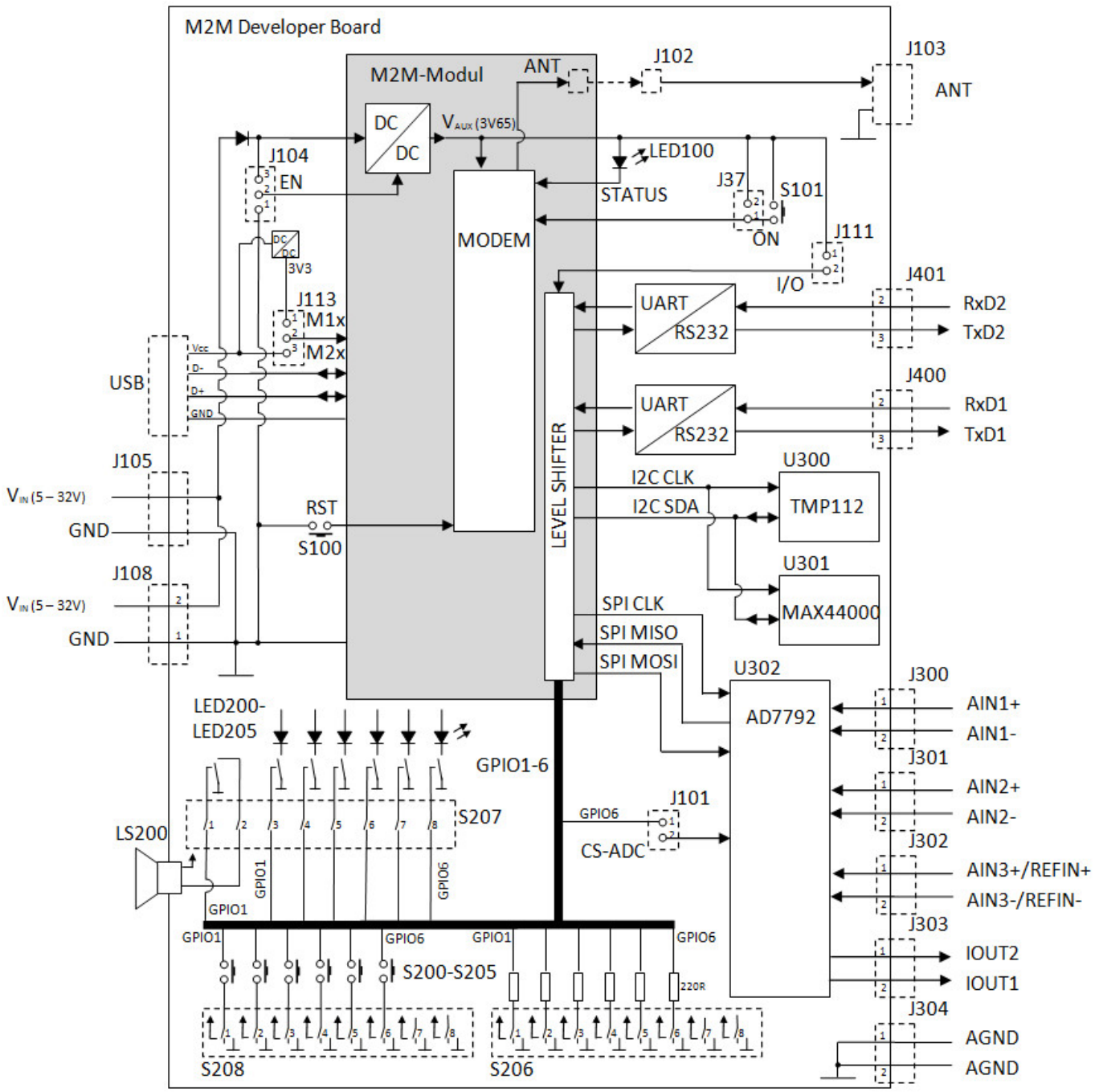

Übersicht rapidM2M Developer Paket M12x

| | |
|---|---|
| **1** COM1 (J400)<br>(siehe "RS232-Schnittstellen" auf Seite 71) | **16** Temperatursensor (U300)<br>(siehe "Temperatursensor TMP112 (U300)" auf Seite 84) |
| **2** Buzzer (LS200)<br>(siehe "Ausgabe von digitalen Signalen" auf Seite 73) | **17** Konstantstromausgänge (J303 und J304)<br>(siehe "ADC AD7792 (U302)" auf Seite 83) |
| **3** COM2 (J401)<br>(siehe "RS232-Schnittstellen" auf Seite 71) | **18** Analoge Differenzeingänge AIN1 bis AIN3 (J300-J302)<br>(siehe "ADC AD7792 (U302)" auf Seite 83) |
| **4** Stiftleiste, auf der die Pins mit den ungeraden Nummern des Moduls aufgelegt sind (J106 und J109)<br>(siehe "Stiftleiste auf der die Pins mit den ungeraden Nummern des M120 Moduls aufgelegt sind (J106 und J109)" auf Seite 76) | **19** Jumper zum Verbinden des GPIO6 mit der CS-Leitung des ADC (J101)<br>(siehe "ADC AD7792 (U302)" auf Seite 83) |
| **5** Jumper zur Versorgung der Level Shifter (J111)<br>(siehe "Jumper zur Versorgung der Level Shifter (J111)" auf Seite 75) | **20** SMA Antennenanschluss (J103)<br>(siehe "Anschluss der GSM-Antenne" auf Seite 96) |
| **6** Jumper zur Auswahl der USB-Versorgung für das Modul (J113)<br>(siehe "Jumper zur Auswahl der USB-Versorgung für das Modul (J113)" auf Seite 75) | **21** U.FL Antennenanschluss zum Verbinden des Moduls mit dem Developer Board (J102)<br>(siehe "Einsetzen des M120 Moduls" auf Seite 93) |
| **7** Mini-B USB (J402)<br>(siehe "USB-Schnittstelle" auf Seite 80) | **22** U.FL Antennenanschluss des Moduls<br>(siehe "Einsetzen des M120 Moduls" auf Seite 93) |
| **8** Status-LED (LED100)<br>(siehe "Status-LED (LED100)" auf Seite 80) | **23** Aufnahme des Moduls<br>(siehe "Aufnahme des M120 Moduls " auf Seite 85) |
| **9** Versorgung (DC-Hohlstecker (J105) oder Klemmleiste (J108) )<br>(siehe "Versorgung (J105 bzw. J108) " auf Seite 80) | **24** rapidM2M Modul M120<br>(siehe "rapidM2M Modul M120 " auf Seite 99) |
| **10** Jumper zum Aktivieren der Energieversorgung des Moduls (J104)<br>(siehe "Jumper zum Aktivieren der Energieversorgung des M120 Moduls (J104) " auf Seite 81) | **25** Stiftleiste, auf der die Pins mit den geraden Nummern des Moduls aufgelegt sind (J107 und J110)<br>(siehe "Stiftleiste auf der die Pins mit den geraden Nummern des M120 Moduls aufgelegt sind (J107 und J110)" auf Seite 78) |
| **11** Jumper zum Einschalten des Wireless GSM-Controllers am Modul (J37)<br>(siehe "Jumper zum Einschalten des Wireless GSM-Controllers am rapidM2M Modul M120 (J37)" auf Seite 81) | **26** LED zur Anzeige des Status des jeweiligen GPIO (LED200-LED205)<br>(siehe "Ausgabe von digitalen Signalen" auf Seite 73) |

| | |
|---|---|
| **12** Taster zum Einschalten des Wireless GSM Controllers am Modul (S101)<br><br>(siehe "Taster zum Einschalten des Wireless GSM-Controllers am rapidM2M Modul M120 (S101)" auf Seite 81) | **27** Taste zur Simulation eines Impulses für den jeweiligen GPIO (S200-S205)<br><br>(siehe "Eingabe von digitalen Signalen" auf Seite 74) |
| **13** Taste zum Reseten des Wireless GSM Controllers am Modul (S100)<br><br>(siehe "Taste zum Reseten des Wireless GSM-Controllers am M120 Modul (S100)" auf Seite 82) | **28** Dip-Switch (S207) zum Aktivieren der LEDs für den jeweiligen GPIO und des Buzzers (LS200), der durch den GPIO1 angesteuert wird<br><br>(siehe "Ausgabe von digitalen Signalen" auf Seite 73) |
| **14** IR-LED für den Näherungssensor (LED300)<br><br>(siehe "Umgebungslicht und IR-Näherungssensor MAX44000 (U301)" auf Seite 85) | **29** Dip-Switch (S206) zum Aktivieren der Pull up bzw. Pull down Widerstände für den jeweiligen GPIO<br><br>(siehe "Eingabe von digitalen Signalen" auf Seite 74) |
| **15** Umgebungslicht und IR-Näherungssensor (U301)<br><br>(siehe "Umgebungslicht und IR-Näherungssensor MAX44000 (U301)" auf Seite 85) | **30** Dip-Switch (S208) zum Festlegen, ob der jeweilige GPIO durch Drücken der zugeordneten Taste auf "low" oder "high" gesetzt wird<br><br>(siehe "Eingabe von digitalen Signalen" auf Seite 74) |

## 6.1.1 Blockschaltbild

***Hinweis:*** *Ein detailliertes Blockschaltbild des ins rapidM2M Developer Board eingesetzten M120 Moduls finden Sie im Kapitel "Blockschaltbild" auf Seite 102. Alle Signale (z.B. SPI, I²C, GPIO,..) des 52pin Steckers des M120 Moduls sind über die Stiftleisten J106, J109 (siehe "ungerade Pinnummern") und J107, J110 (siehe "gerade Pinnummern") des rapidM2M Developer Board zugängig. Zugunsten der Übersichtlichkeit wurden diese Stiftleisten nicht ins Blockschaltbild eingezeichnet.*




Blockschaltbild des Developer Boards

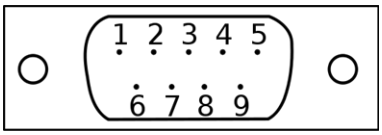
## 6.1.2 RS232-Schnittstellen

Am Anschluss COM1 (J400) des Developer Boards ist die UART1 des M120 Moduls zugängig. Am Anschluss COM2 (J401) des Developer Boards ist die UART2 des M120 Moduls zugängig. Die Umwandlung auf den RS232-Pegel erfolgt dabei jeweils durch einen Pegelwandler, der sich auf dem rapidM2M Developer Board befindet (siehe "Blockschaltbild" auf Seite 70).

> ***Wichtiger Hinweis:*** *Die RS232-Pegelwandler können nicht per Jumper deaktiviert werden. Wenn Sie die Signale direkt über die Pins der Stiftleiste auf der die ungeraden Nummern des M120 Moduls aufgelegt sind, einspeisen wollen (siehe "Stiftleiste auf der die Pins mit den ungeraden Nummern des M120 Moduls aufgelegt sind (J106 und J109)" auf Seite 76), müssen die RS232-Pegelwandler ausgelötet werden.*

| | |
|---|---|
| Baudrate | 300-115200 |
| Stopbits | 1, 2 |
| Parität | N, E, O |
| Datenbits | 7, 8 |
| Modi | ASCII |

Die Richtung der Signale entspricht jener eines DTE (z.B. PC).




9pol. Sub-D(m)

**Belegung des Sub-D Steckers**

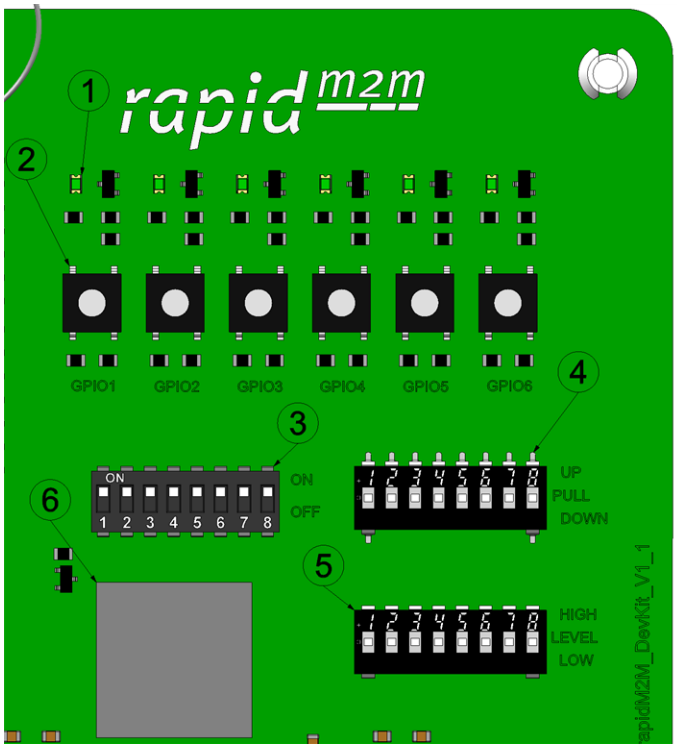
| Pin | Signal |
|---|---|
| 1 | NC |
| 2 | RXD |
| 3 | TXD |
| 4 | NC |
| 5 | GND |
| 6 | NC |
| 7 | RTS |
| 8 | CTS |
| 9 | NC |

## 6.1.3 Simulationsschaltungen

*Hinweis: Einen detaillierten Schaltplan der Simulationsschaltungen finden Sie im Kapitel "GPIO" auf Seite 90.*

Am rapidM2M Developer Board befinden sich 6 baugleiche Schaltungen, die es ermöglichen den Zustand des jeweiligen GPIOs mittels LED (LED200-LED205) anzuzeigen oder einen Zustand mittels Taster (S200-S205) einzugeben.

Der GPIO1 kann zusätzlich auch noch zur Ansteuerung des Buzzers LS200 verwendet werden.



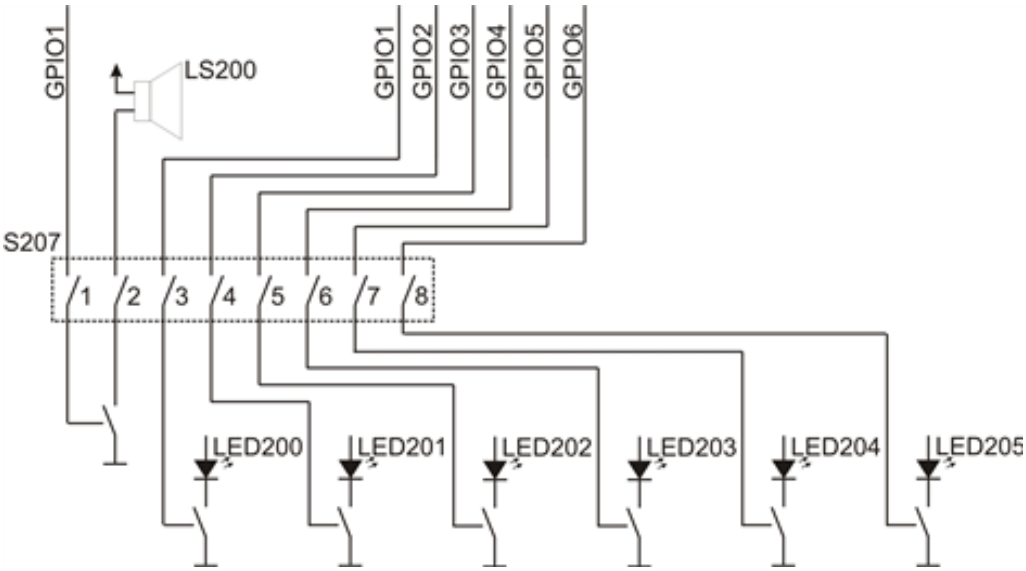

Detailansicht: Simulationschaltungen

| | | | |
|---|---|---|---|
| 1 | LED (LED200) zur Anzeige des Status des GPIO1 | 4 | Dip-Switch (S206) zum Aktivieren der Pull up bzw. Pull down Widerstände für den jeweiligen GPIO |
| 2 | Taste (S200) zur Simulation eines Impulses für den GPIO1 | 5 | Dip-Switch (S208) zum Festlegen, ob der jeweilige GPIO durch Drücken der zugeordneten Taste auf "Low" oder "High" gesetzt wird |
| 3 | Dip-Switch (S207) zum Aktivieren der LEDs für den jeweiligen GPIO und des Buzzers (LS200), der durch den GPIO1 angesteuert wird. | 6 | Buzzer (LS200) |

### 6.1.3.1 Ausgabe von digitalen Signalen

Um einen der GPIOs als Ausgang zu verwenden, muss dieser erst mittels PAWN-Script-Funktion "rM2M_GpioDir()" als Ausgang konfiguriert werden. Anschließend kann die Funktion "rM2M_ GpioSet()" verwendet werden, um den entsprechenden GPIO auf "high" oder "low" zu setzen. Bei der Schalterstellung "ON" des Dip-Switches S207 leuchtet die LED, wenn der zugehörige GPIO auf "high" gesetzt wird bzw. gibt der Buzzer einen Ton ab, wenn der GPIO1 auf "low" gesetzt wird.

> ***Hinweis:*** *Der entsprechende Pull up- bzw. Pull down-Widerstand muss deaktiviert (mittlere Stellung des Dip-Switches S206) sein, wenn Sie einen GPIO als Ausgang benutzen wollen. Die Ausgangsstufe des GPIO ist nicht in der Lage den Pegel entgegen dem Widerstand zu schalten. D.h. ist der Pull up aktiv, kann die Ausgangsstufe den Pegel nicht auf "low" ziehen.*



Prinzipschaltbild für den Fall, dass die GPIOs als Ausgänge verwendet werden sollen

**Dip-Switch S207 zum Aktivieren der LEDs und des Buzzers**

| | |
|---|---|
| 1 | GPIO1 mit dem FET zur Ansteuerung des Buzzers (LS200) verbinden |
| 2 | Ansteuer-FET mit dem Buzzer (LS200) verbinden |
| 3 | GPIO1 mit dem FET zur Ansteuerung des LED200 verbinden |
| 4 | GPIO2 mit dem FET zur Ansteuerung des LED201 verbinden |
| 5 | GPIO3 mit dem FET zur Ansteuerung des LED202 verbinden |
| 6 | GPIO4 mit dem FET zur Ansteuerung des LED203 verbinden |
| 7 | GPIO5 mit dem FET zur Ansteuerung des LED204 verbinden |
| 8 | GPIO6 mit dem FET zur Ansteuerung des LED205 verbinden |

> ***Hinweis:*** *Bitte beachten Sie, dass GPIO6 mittels Jumper "CS-ADC" (J101) auch noch mit der CS-Leitung des ADC (AD7792) verbunden ist (siehe "Blockschaltbild" auf Seite 70). Entfernen Sie diesen Jumper, falls Sie die Simulationsschaltung für GPIO6 verwenden möchten (siehe "Übersicht" auf Seite 67).*

### 6.1.3.2 Eingabe von digitalen Signalen

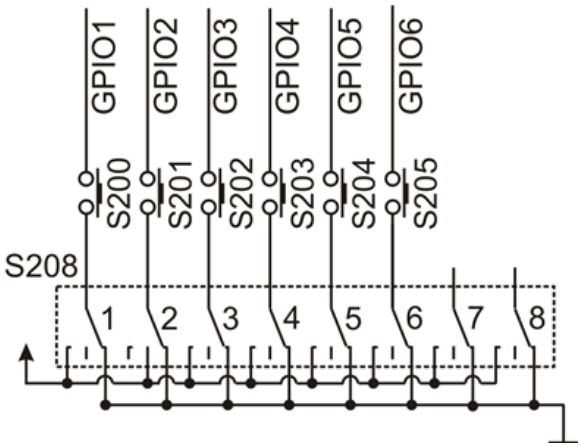
Um einen der GPIOs als Eingang zu verwenden, muss dieser erst mittels PAWN-Script-Funktion "rM2M_GpioDir()" als Eingang konfiguriert werden. Der am GPIO anliegende Pegel kann daraufhin mit der Funktion "rM2M_GpioGet()" eingelesen werden. Über den Dip-Switch S208 kann festgelegt werden, ob der dem GPIO zugeordnete Taster (S200-S205) den Eingang beim Drücken auf "high" oder "low" ziehen soll. Die folgenden beiden Tabellen erklären die Funktion des Dip-Switches S208.

**Erklärung der Schalterstellung des Dip-Switches S208**

| | |
|---|---|
| + | Durch Drücken der Taste wird der GPIO auf "high" gezogen. |
| 0 | --- |
| - | Durch Drücken der Taste wird der GPIO auf "low" gezogen. |

**Dip-Switch S208 zum Festlegen, ob der GPIO Durch drücken der Taste auf "low" oder "high" gesetzt wird**

| | |
|---|---|
| 1 | GPIO1 |
| 2 | GPIO2 |
| 3 | GPIO3 |
| 4 | GPIO4 |
| 5 | GPIO5 |
| 6 | GPIO6 |
| 7 | NC |
| 8 | NC |



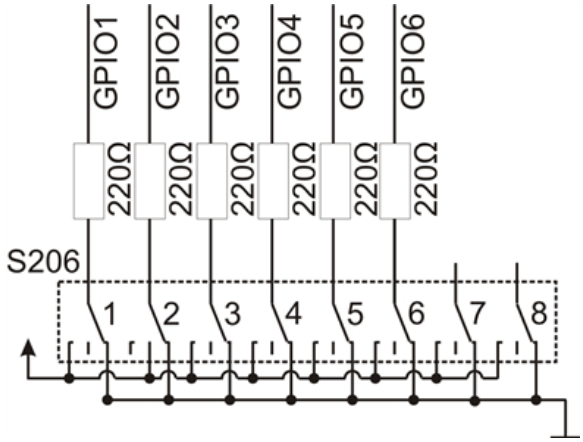

Prinzipschaltbild: Dip-Switch S208 zum Festlegen, ob der GPIO durch Drücken der Taste auf "low" oder "high" gesetzt wird

Über Dip-Switch S206 können Pull up- bzw. Pull down-Widerstände wie in den beiden folgenden Tabellen beschrieben, zugeschaltet werden.

**Erklärung der Schalterstellung des Dip-Switches S206**

| | |
|---|---|
| + | 220Ω Pull up |
| 0 | --- |
| - | 220Ω Pull down |

Dip-Switch S206 zum Aktivieren der der Pull up- bzw. Pull down-Widerstände

| | | |
|---|---|---|
| 1 | GPIO1 |  |
| 2 | GPIO2 | |
| 3 | GPIO3 | |
| 4 | GPIO4 | |
| 5 | GPIO5 | |
| 6 | GPIO6 | |
| 7 | NC | |
| 8 | NC | Prinzipschaltbild: Dip-Switch S206 zum Aktivieren der Pull up bzw. Pull down Widerstände |

## 6.1.4 Jumper zur Versorgung der Level Shifter (J111)

Um die Signalpegel des M120 Moduls an die umgebende Hardware anzupassen, ist das Modul mit Level Shifter für alle I/O-Signale ausgestattet. Für die Anpassung muss von der umgebenden Hardware eine zusätzliche Versorgungsspannung für die Level Shifter im Bereich von 2,8...5,5V bereitgestellt werden. Über den Jumper J111 des Developer Boards wird $V_{AUX}$ des M120 Moduls als Versorgung für die Level Shifter zurück ins Modul geführt (siehe "Blockschaltbild" auf Seite 70).

*Hinweis: Der Jumper J111 sollte immer geschlossen sein.*

## 6.1.5 Jumper zur Auswahl der USB-Versorgung für das Modul (J113)

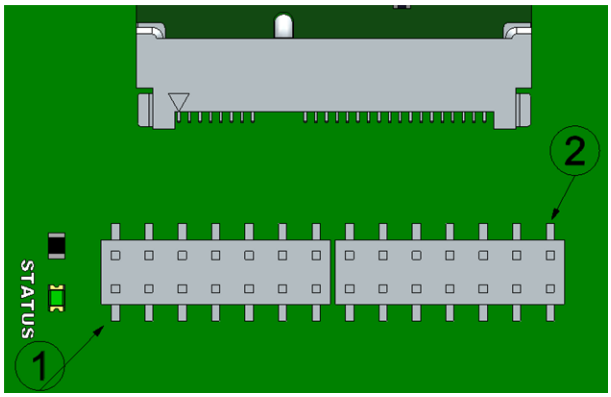
Mit Hilfe des Jumpers J113 kann die USB-Versorgung für das ins rapidM2M Developer Board eingesetzte Modul festgelegt werden (siehe "Blockschaltbild" auf Seite 70). Dadurch ist es möglich sowohl Module der Serien "M1x" als auch "M2x" zu verwenden.

Erklärung zu den Auswahlmöglichkeiten des Jumper J113

| Position | USB-Versorgung | Module |
|---|---|---|
| M1x | 3,3V | rapidM2M Modul M11 (300068 )<br>rapidM2M Modul M120 (300087 ) |
| M2x | 5V | |

*Hinweis: Bevor Sie die USB Verbindung zwischen rapidM2M Developer Board und Ihrem PC herstellen stellen Sie sicher, dass der Jumper J113 passend zu dem von Ihnen verwendeten Modul gesetzt ist.*

## 6.1.6 Stiftleiste auf der die Pins mit den ungeraden Nummern des M120 Moduls aufgelegt sind (J106 und J109)




Stiftleiste auf der die Pins mit den ungeraden Nummern des M120 Moduls aufgelegt sind

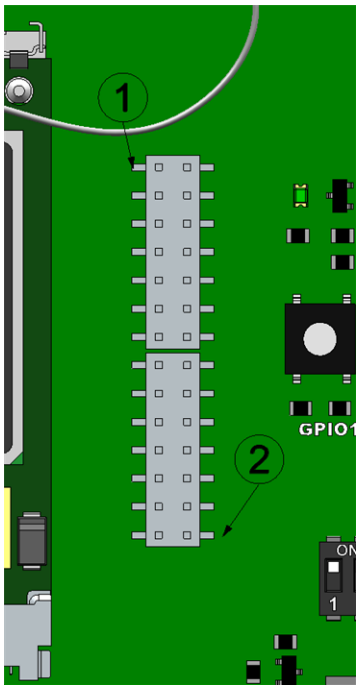
| | |
|---|---|
| **1** Pin1 | **2** Pin28 |

| Pin | Typ | Bezeichnung | Beschreibung |
|---|---|---|---|
| 1 | I | Modul_ON | Pin zum Einschalten des Wireless GSM Controllers am rapidM2M Modul M120 |
| 2 | I | EN_Power | Pin zum Aktivieren des DC/DC-Wandlers, der aus der an $V_{IN}$ angelegten Spannung die interne Versorgungsspannung $V_{AUX}$ erzeugt |
| 3 | O | LED_Status | Open Drain Ausgang zum Anschluss einer Status-LED |
| 4 | I | Reset | Pin zum Reseten des Wireless GSM Controllers am rapidM2M Modul M120 |
| 5-6, 13 | | GND | Masse |
| 7-8 | | | reserviert für Erweiterungen[1)] |
| 9 | I | $V_{IN}$ | Versorgungsspannung (4,5...32VDC ) für das rapidM2M Modul M120 |
| 10, 15 | I/O | $V_{AUX}$ | 3,65V Spannungsausgang, wenn das rapidM2M Modul M120 über $V_{IN}$versorgt wird<br>Versorgungsspannungseingang (3,2...4,5V ), wenn an $V_{IN}$ keine Spannung angelegt wird |
| 11 | I/O | GPIO2 | GPIO zum Austausch von Signalen zw. M120 und der umgebenden Hardware |
| 12 | I/O | GPIO3 | |
| 14 | I/O | GPIO1 | |
| 16 | I | I/O Level | 2,8...5,5V Versorgung der Level Shifter zur Anpassung der Signalpegel des M120 Moduls an die umgebende Hardware |
| 17 | I | INT0 | Eingänge über die die umgebende Hardware Interrupts am rapidM2M Modul M120 auslösen kann |
| 18 | I | INT1 | |
| 19-20 | | GND | Masse |
| 21 | O | RTS2 | RTS Leitung der UART2[2)] |
| 22 | I | CTS2 | CTS Leitung der UART2[2)] |
| 23 | O | TxD2 | TxD Leitung der UART2[2)] |
| 24 | I | RxD2 | RxD Leitung der UART2[2)] |
| 25 | O | RTS1 | RTS Leitung der UART1[2)] |
| 26 | I | CTS1 | CTS Leitung der UART1[2)] |
| 27 | O | TxD1 | TxD Leitung der UART1[2)] |
| 28 | I | RxD1 | RxD Leitung der UART1[2)] |

[1)] Es dürfen keine Signale an diese Pins angelegt werden.

[1)] Sollen die Signale direkt über diese Pins eingespeist werden, müssen die RS232-Pegelwandler des Developer Boards ausgelötet werden.

## 6.1.7 Stiftleiste auf der die Pins mit den geraden Nummern des M120 Moduls aufgelegt sind (J107 und J110)




Stiftleiste, auf der die Pins mit den geraden Nummern des M120 Moduls aufgelegt sind

| | |
|---|---|
| **1** Pin1 | **2** Pin28 |

| Pin | Typ | Bezeichnung | Beschreibung |
|---|---|---|---|
| 1, 27-28 | I | $V_{IN}$ | Versorgungsspannung (4,5...32VDC ) für das rapidM2M Modul M120 |
| 2, 18 | I/O | $V_{AUX}$ | 3,65V Spannungsausgang, wenn das rapidM2M Modul M120 über $V_{IN}$ versorgt wird<br>Versorgungsspannungseingang (3,2...4,5V ), wenn an $V_{IN}$ keine Spannung angelegt wird |
| 3-8 | | | reserviert für Erweiterungen[1)] |
| 9-10 | | GND | Masse |
| 11 | I/O | GPIO5 | GPIO zum Austausch von Signalen zw. M120 und der umgebenden Hardware |
| 12 | I/O | GPIO4 | |
| 13 | I | Reset | Pin zum Reseten des Wireless GSM Controllers am rapidM2M Modul M120 |
| 14 | | NC | |
| 15 | O | I2C_SCL | Clock-Leitung der $I^2C$-Schnittstelle |
| 16 | I/O | I2C_SDA | Daten-Leitung der $I^2C$-Schnittstelle |
| 17 | I | USB_3V3 | USB-Versorgung<br>***Wichtiger Hinweis:*** *An diesem Pin sind nur 3,3V anstelle der sonst üblichen 5V zulässig.* |
| 19 | I/O | USB_D- | "Data-"-Leitung der USB-Schnittstelle |
| 20 | I/O | USB_D+ | "Data+"-Leitung der USB-Schnittstelle |
| 21-22 | | GND | Masse |
| 23 | O | SPI_MOSI | "Master out, Slave in"-Leitung der SPI-Schnittstelle |
| 24 | I | SPI_MISO | "Master in, Slave out"-Leitung der SPI-Schnittstelle |
| 25 | O | SPI_CLK | Clock-Leitung der SPI-Schnittstelle |
| 26 | I/O | GPIO6 | GPIO zum Austausch von Signalen zw. M120 und der umgebenden Hardware |

[1)] Es dürfen keine Signale an diese Pins angelegt werden.

### 6.1.8 USB-Schnittstelle

Über die USB-Slave-Schnittstelle wird die Verbindung zu einem PC hergestellt. Dabei ist ausschließlich die Kommunikation mit dem rapidM2M Toolset vorgesehen. Es ist nicht möglich mittels PAWN-Script auf die USB-Schnittstelle zuzugreifen. Sowohl das Toolset als auch die USB-Schnittstelle werden nur während der Entwicklungsphase eines neuen Scripts benötigt. D.h., es ist nicht zwingend erforderlich, dass Sie die USB-Schnittstelle in Ihr Hardwaredesign mit aufnehmen. Eine detaillierte Beschreibung des rapidM2M Toolset finden Sie im Kapitel "rapidM2M Toolset " auf Seite 141. Das rapidM2M Toolset steht unter folgender Adresse gratis zum Download bereit:

**www.microtronics.at/Toolset**

***Wichtiger Hinweis:*** *Bitte beachten Sie bei der Integration des M120 Moduls in eine eigene Hardware, dass das rapidM2M Modul M120 mit einer USB-Versorgung von 3,3V arbeitet. Die Umwandlung von 5V auf 3,3V erfolgt durch einen DC/DC-Wandler, der sich am rapidM2M Developer Board befindet (siehe "Blockschaltbild" auf Seite 70). Dazu muss der Jumper zur Auswahl der USB-Versorgung (J113) auf Position "M1x" gesetzt sein.*

### 6.1.9 Status-LED (LED100)

Die Status-LED leuchte nach dem Einschalten des Wireless GSM Controllers des M120 Moduls ca. 1sec.

### 6.1.10 Versorgung (J105 bzw. J108)

Die Versorgung des Developer Boards kann entweder über den DC-Hohlstecker (J105) oder die Klemmleiste (J108) erfolgen. Der Versorgungsspannungseingang verfügt über einen Verpolungsschutz. Da im Auslieferungszustand der Jumper J37 ("MODUL ON") geschlossen und der Jumper J104 ("EN POWER") auf Position "ON" gesetzt ist, startet der Wireless GSM-Controller des M120 Moduls nach dem Herstellen der Energieversorgung des Developer Boards sobald er das "Modul_ON" Signal erkannt hat (ca. 1,5sec. nach Herstellung der Energieversorgung). Neben dem im Lieferumfang enthaltenen Steckernetzgerät können auch Netzgeräte verwendet werden, die die in den folgenden Kapiteln beschriebenen Bedingungen erfüllen.

#### 6.1.10.1 Klemmleiste (J108)

| | |
|---|---|
| U | 5...32VDC |
| $I_{min}$ | t.b.d. mA@5V<br>t.b.d. mA@12V<br>t.b.d. mA@24V |

**Belegung der Klemmleiste (J108)**

| | | |
|---|---|---|
| 1 | GND | Masse |
| 2 | $+V_{dd}$ | Versorgungsspannung: 5...32VDC |

#### 6.1.10.2 DC-Hohlstecker (J105)

| | |
|---|---|
| U | 5...32VDC |
| $I_{min}$ | t.b.d. mA@5V<br>t.b.d. mA@12V<br>t.b.d. mA@24V |
| Außen-/Innen-Ø | 5,5/2,1mm |
| Polarität | +Innen |

### 6.1.11 Jumper zum Aktivieren der Energieversorgung des M120 Moduls (J104)

Ist der Jumper J104 geöffnet oder sind Pin1 und Pin2 durch eine Steckbrücke verbunden, ist der DC/DC-Wandler des M120 Moduls inaktiv und somit die interne Versorgung des Moduls abgeschaltet.

Werden Pin2 und Pin3 verbunden, wird der DC/DC-Wandler aktiviert. Durch den Soft-Start des Wandlers dauert es 50ms bis die interne Versorgungsspannung des M120 Moduls den Maximalwert erreicht.

### 6.1.12 Taster zum Einschalten des Wireless GSM-Controllers am rapidM2M Modul M120 (S101)

50ms (Soft-Start) nachdem die Energieversorgung des M120 Moduls mittels Jumper J104 (siehe "Jumper zum Aktivieren der Energieversorgung des M120 Moduls (J104) " auf Seite 81) aktiviert wurde, kann der Wireless GSM Controller des M120 Moduls durch einen 1,5sec. langen Druck der Taste S101 eingeschaltet werden. Während der Startphase sollte die Reset-Taste S100 nicht betätigt werden. Die Status-LED (LED100) leuchtet während des Boot-Vorgangs ca. 1sec.

Ein Abschalten des Wireless GSM Controllers ist bei der aktuellen Firmware des M120 Moduls nicht vorgesehen.

Ein detaillierte Beschreibung des Timings finden Sie unter "ENCPU-Signal" auf Seite 108.

### 6.1.13 Jumper zum Einschalten des Wireless GSM-Controllers am rapidM2M Modul M120 (J37)

Der Jumper J37 ist parallel zur Taste S101 zum Einschalten des Wireless GSM-Controllers geschaltet (siehe "Blockschaltbild" auf Seite 70). D.h. statt den Wireless GSM-Controller am rapidM2M Modul M120 mittels Tastendruck einzuschalten, kann der Jumper J37 gesetzt werden. Vorausgesetzt der Jumper J104 (siehe "Jumper zum Aktivieren der Energieversorgung des M120 Moduls (J104) " auf Seite 81) ist auf Postion "ON" gesetzt, startet dadurch nach dem Herstellen der Energieversorgung der Wireless GSM-Controller des M120 Moduls sobald er das "Modul_ON" Signal erkannt hat (ca.

1,5sec. nach Herstellung der Energieversorgung). Während der Startphase sollte die Reset-Taste S100 nicht betätigt werden. Die Status-LED (LED100) leuchtet während des Boot-Vorgangs ca. 1sec.

Ein Abschalten des Wireless GSM Controllers ist bei der aktuellen Firmware des M120 Moduls nicht vorgesehen.

Ein detaillierte Beschreibung des Timings finden Sie unter "ENCPU-Signal" auf Seite 108.

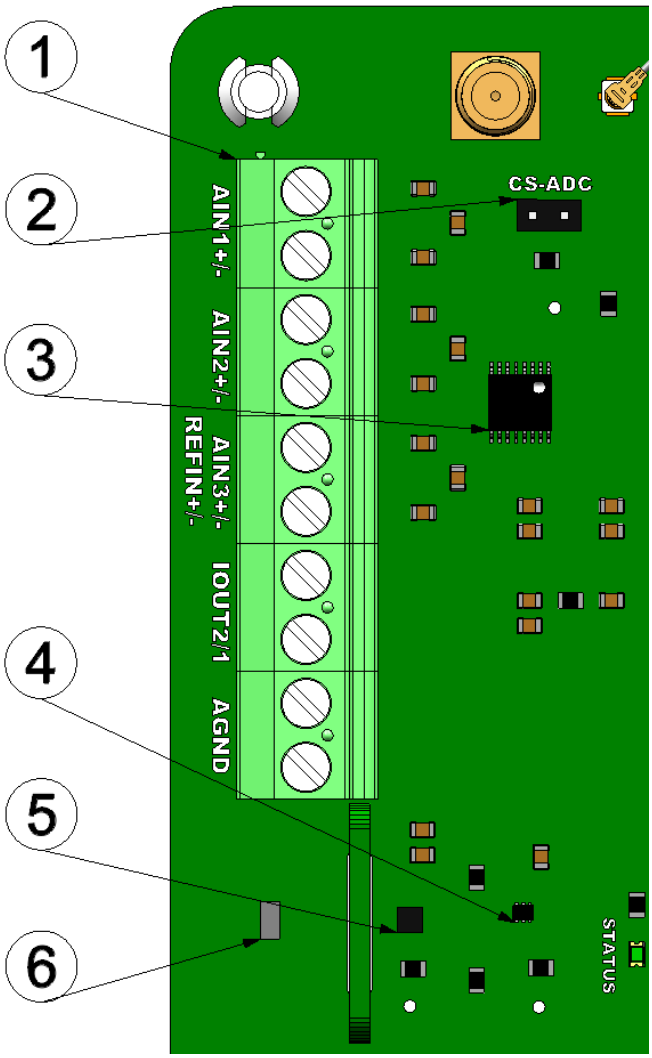
### 6.1.14 Taste zum Reseten des Wireless GSM-Controllers am M120 Modul (S100)

Diese Taste sollte nur als Notfall-Reset betrachtet werden, da eine Reset-Prozedur vom Wireless GSM Controller während des Boot-Vorgangs selbstständig durchgeführt wird. Sollte ein Reset dennoch erforderlich werden, muss die Taste S100 kurz (200µs ) gedrückt werden. Danach muss der Wireless GSM Controller des M120 Moduls mittels Taste S101 (siehe "Taster zum Einschalten des Wireless GSM-Controllers am rapidM2M Modul M120 (S101)" auf Seite 81) wieder eingeschaltet werden.

Ein detaillierte Beschreibung des Timings finden Sie unter "Reset-Signal" auf Seite 109.

### 6.1.15 Sensoren

Am rapidM2M Developer Board befinden sich 3 Sensor-Chips, die zusammen mit den Script-Beispielen (siehe "Beispiele" auf Seite 280) die Funktionsweise bzw. die Integration des M120 Moduls demonstrieren sollen. Der Temperatursensor TMP112 (U300) und der Umgebungslicht/IR-Näherungssensor MAX44000 (U301) sind mittels $I^2C$-Bus mit dem rapidM2M Modul M120 verbunden. Der ADC AD7792 (U302) ist mittels SPI-Bus mit dem rapidM2M Modul M120 verbunden.




Detailansicht: Sensoren

| | | | |
|---|---|---|---|
| 1 | Klemmleisten J300-J304: Differnzeingänge und Konstantstromausgänge des AD7792 (U302) | 4 | Temperatursensor TMP112 (U300) |
| 2 | Jumper zum Verbinden des GPIO6 mit der CS-Leitung des ADC (J101) | 5 | Umgebungslicht und IR-Näherungssensor MAX44000 (U301) |
| 3 | ADC AD7792 (U302) | 6 | IR-LED für den Näherungssensor MAX44000 (LED300) |

### 6.1.15.1 ADC AD7792 (U302)

Der ADC AD7792 (U302) bietet 3 Differenzeingänge und 2 Konstantstromausgänge. Es besteht auch die Möglichkeit eine externe Referenzspannung zu verwenden. Allerdings wird dadurch einer der Differenzeingänge belegt. Das mit dem rapidM2M Toolset gelieferte Script "spi.p" (siehe t.b.d.) enthält ein Beispiel für die Verwendung des AD7792 (U302).

> ***Hinweis:*** *Für die Verwendung des AD7792 (U302) muss der Jumper "CS-ADC" (J101), der die CS-Leitung des AD7792 (U302) mit dem GPIO6 verbindet, geschlossen sein.*

### Differenzeingänge

| | |
|---|---|
| Auflösung | 16 Bit |
| Eingangsspannungsbereich | $V_{ref}$ / Gain |
| interne Referenzspannung | 1,17V |
| externe Referenzspannung | nominal 2,5V, max3,3V |
| Gain | 1-128 |

### Konstantstromausgänge

| | |
|---|---|
| Ausgangsstrom | 10/210/1000µA |

### Klemmleisten J300-J304 (Analoge Differenzeingänge und Konstantstromausgänge)

| | | | |
|---|---|---|---|
| J300 | 1 | AIN1+ | Differenzeingang 1 |
| | 2 | AIN1- | |
| J301 | 1 | AIN2+ | Differenzeingang 2 |
| | 2 | AIN2- | |
| J302 | 1 | AIN3+ / REFIN+ | Differenzeingang 3 / Eingang für die externe Referenzspannung |
| | 2 | AIN3- / REFIN- | |
| J303 | 1 | IOUT2 | Konstantstromausgang 2 |
| | 2 | IOUT1 | Konstantstromausgang 1 |
| J304 | 1 | AGND | Analogmasse |
| | 2 | AGND | Analogmasse |

Weitere technische Daten und Funktionsbeschreibungen entnehmen Sie bitte dem Datenblatt des Herstellers "Analog Devices". Die Herstellerbezeichnung des ADC ist "AD7792BRUZ". Bei Unstimmigkeiten zwischen den in diesem Handbuch angegebenen technischen Daten und den im Datenblatt des Chip-Herstellers "Analog Devices" enthalten Daten, gelten immer jene des Chip-Herstellers "Analog Devices".

### 6.1.15.2 Temperatursensor TMP112 (U300)

Das mit dem rapidM2M Toolset gelieferte Script "i2c.p" (siehe t.b.d.) enthält ein Beispiel für die Verwendung der beiden (TMP112 und MAX44000) am $I^2C$-Bus angebundenen Sensoren.

| | |
|---|---|
| Messbereich | -40...+125°C |
| Auflösung | 0,0625°C |

Weitere technische Daten und Funktionsbeschreibungen entnehmen Sie bitte dem Datenblatt des Herstellers "Texas Instruments". Die Herstellerbezeichnung des Temperatursensors ist "TMP112AIDRLT". Bei Unstimmigkeiten zwischen den in diesem Handbuch angegebenen technischen Daten und den im Datenblatt des Chip-Herstellers "Texas Instruments" enthalten Daten, gelten immer jene des Chip-Herstellers "Texas Instruments".

#### 6.1.15.3 Umgebungslicht und IR-Näherungssensor MAX44000 (U301)

Beim MAX44000 (U301) handelt es sich um einen kombinierten Umgebungslicht und IR-Näherungssensor mit integrierter Treiberstufe für den Anschluss eines IR-LED. Das mit dem rapidM2M Toolset gelieferte Script "i2c.p" (siehe t.b.d.) enthält ein Beispiel für die Verwendung der beiden (TMP112 und MAX44000) am $I^2C$-Bus angebundenen Sensoren.

| | |
|---|---|
| Messbereich | 0,03...65535lux |

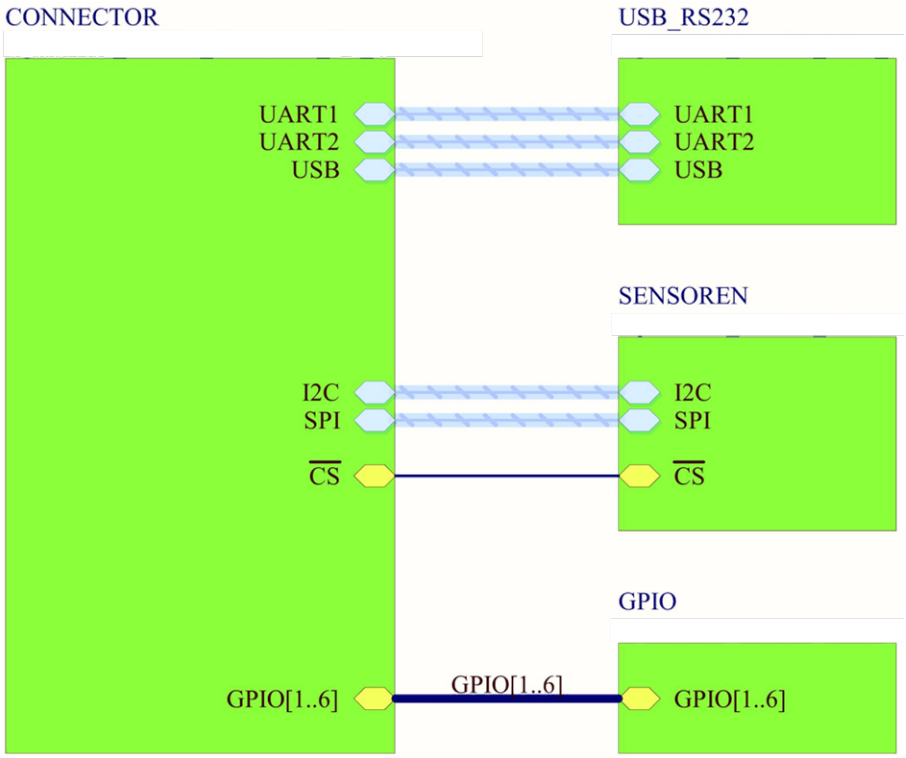
Weitere technische Daten und Funktionsbeschreibungen entnehmen Sie bitte dem Datenblatt des Herstellers "Maxim". Die Herstellerbezeichnung des Umgebungslicht und IR-Näherungssensors ist "MAX44000GDT+". Bei Unstimmigkeiten zwischen den in diesem Handbuch angegebenen technischen Daten und den im Datenblatt des Chip-Herstellers "Maxim" enthalten Daten, gelten immer jene des Chip-Herstellers "Maxim".

### 6.1.16 Aufnahme des M120 Moduls

Der Hersteller der beim rapidM2M Developer Board verwendeten Aufnahme für das M120 Modul ist "JAE Connectors". Die Herstellerbezeichnung dieses Steckers ist "MM60-52B1-G1-R850".

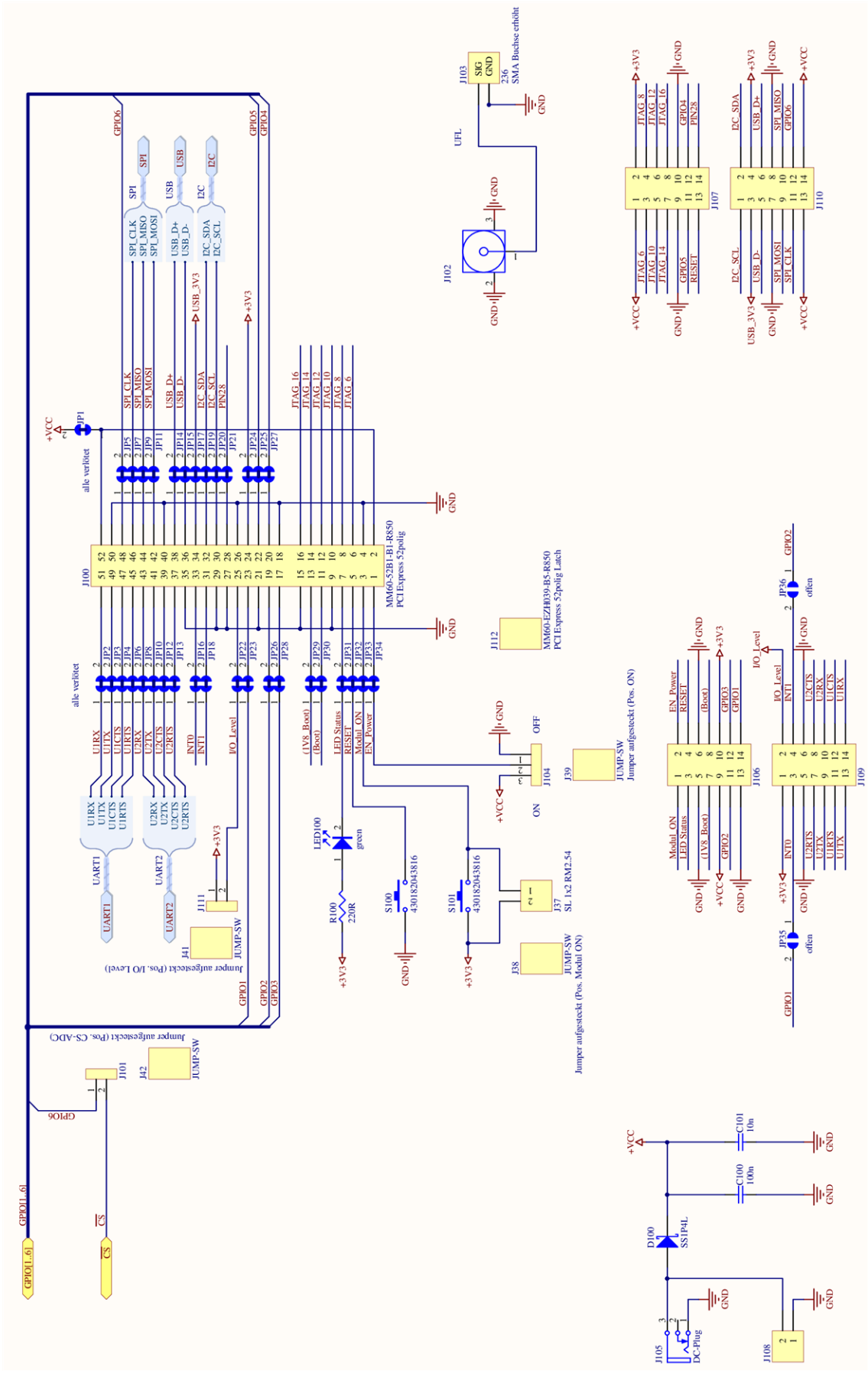
## 6.2 Schaltplan

### 6.2.1 Übersicht

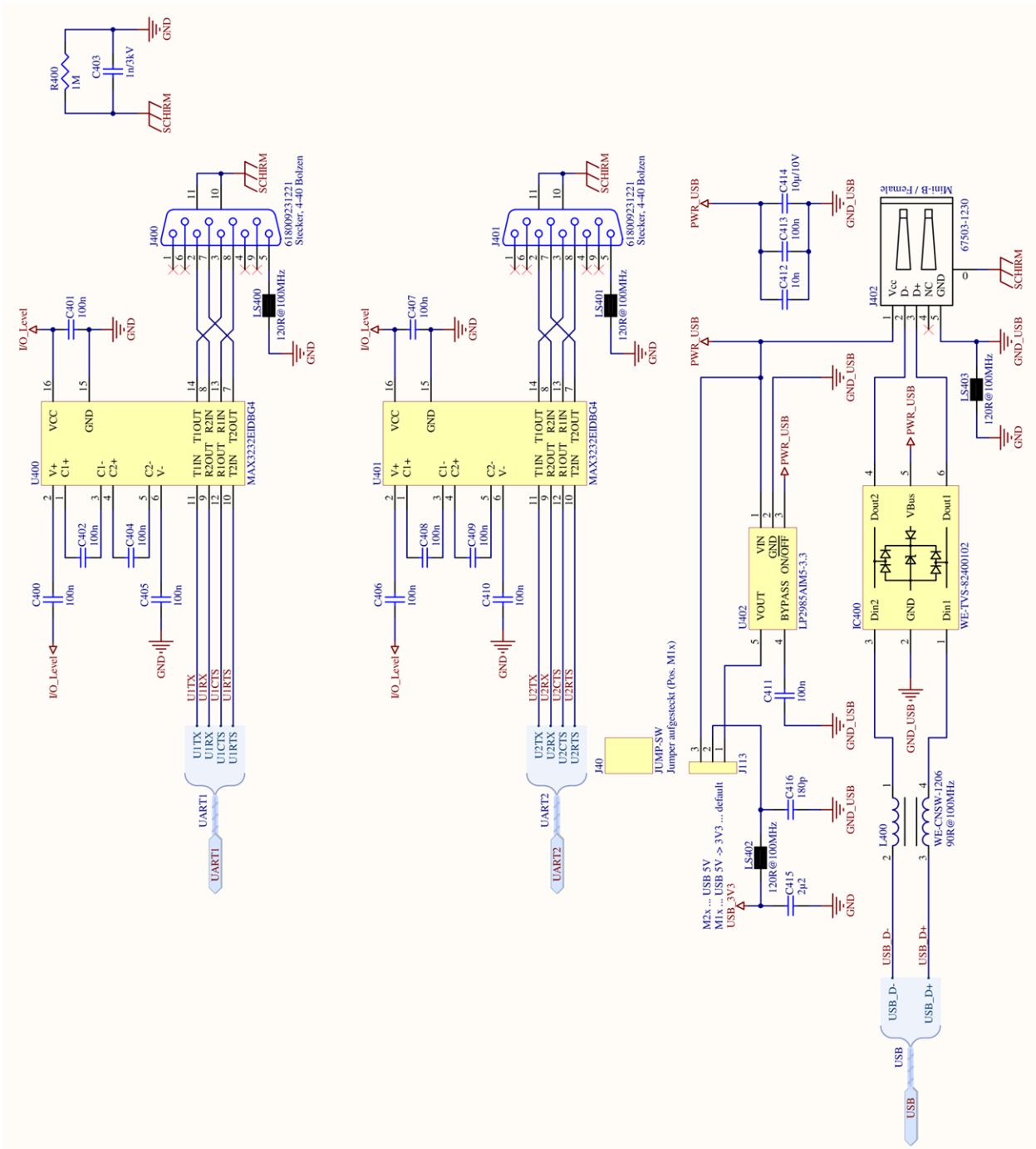



Schaltplanübersicht des Developer Boards

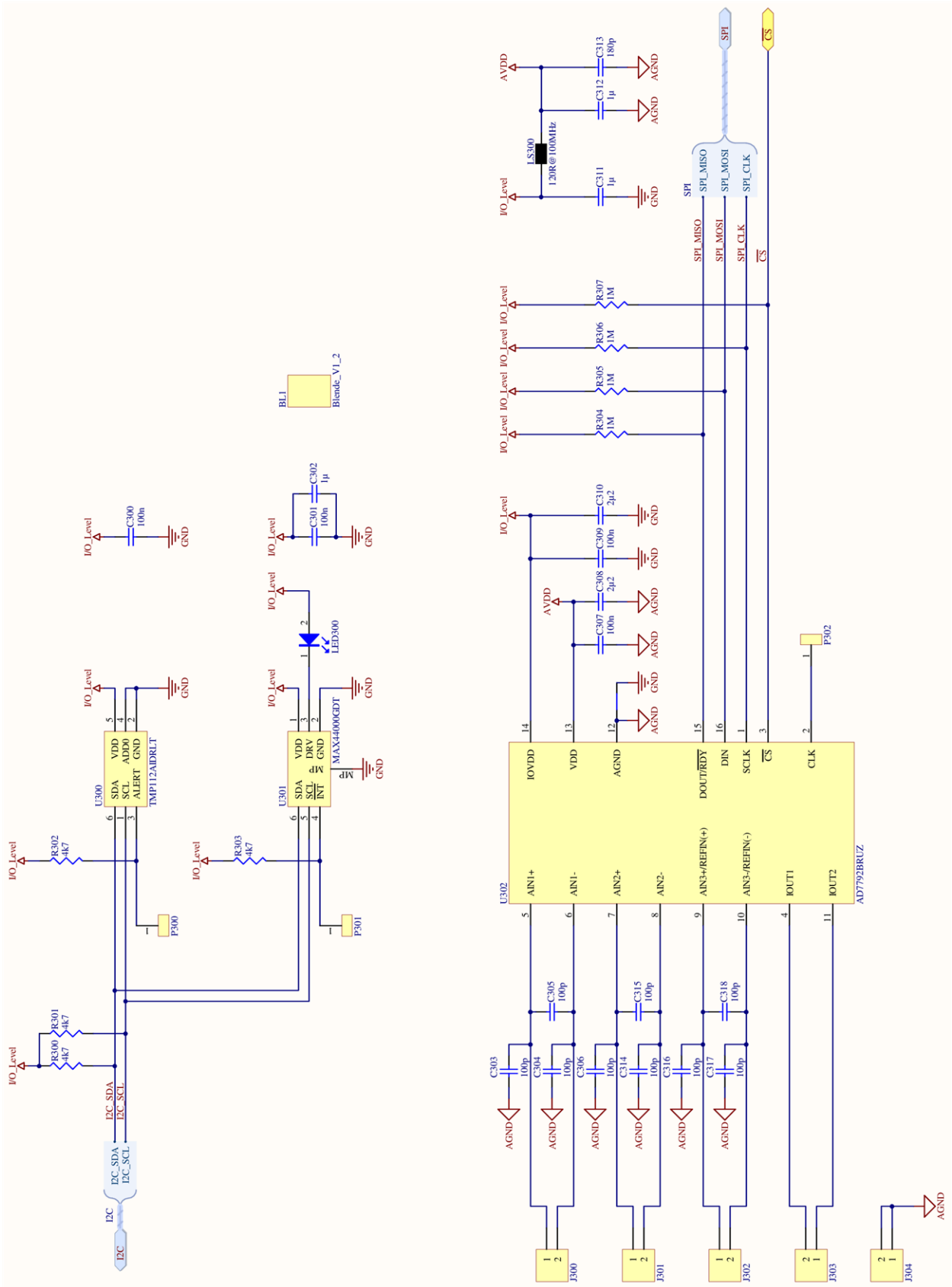
## 6.2.2 Connector



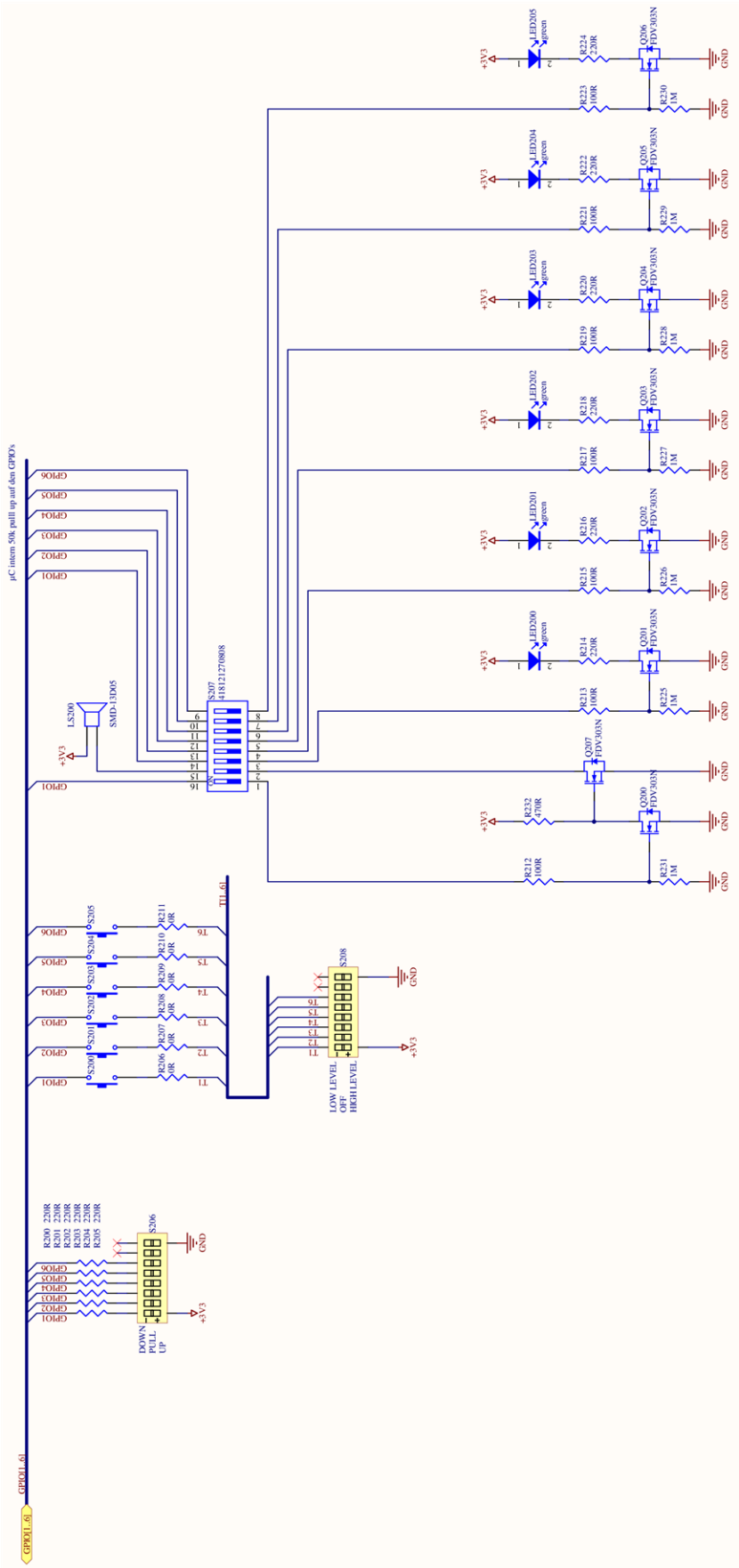
## 6.2.3 USB RS232



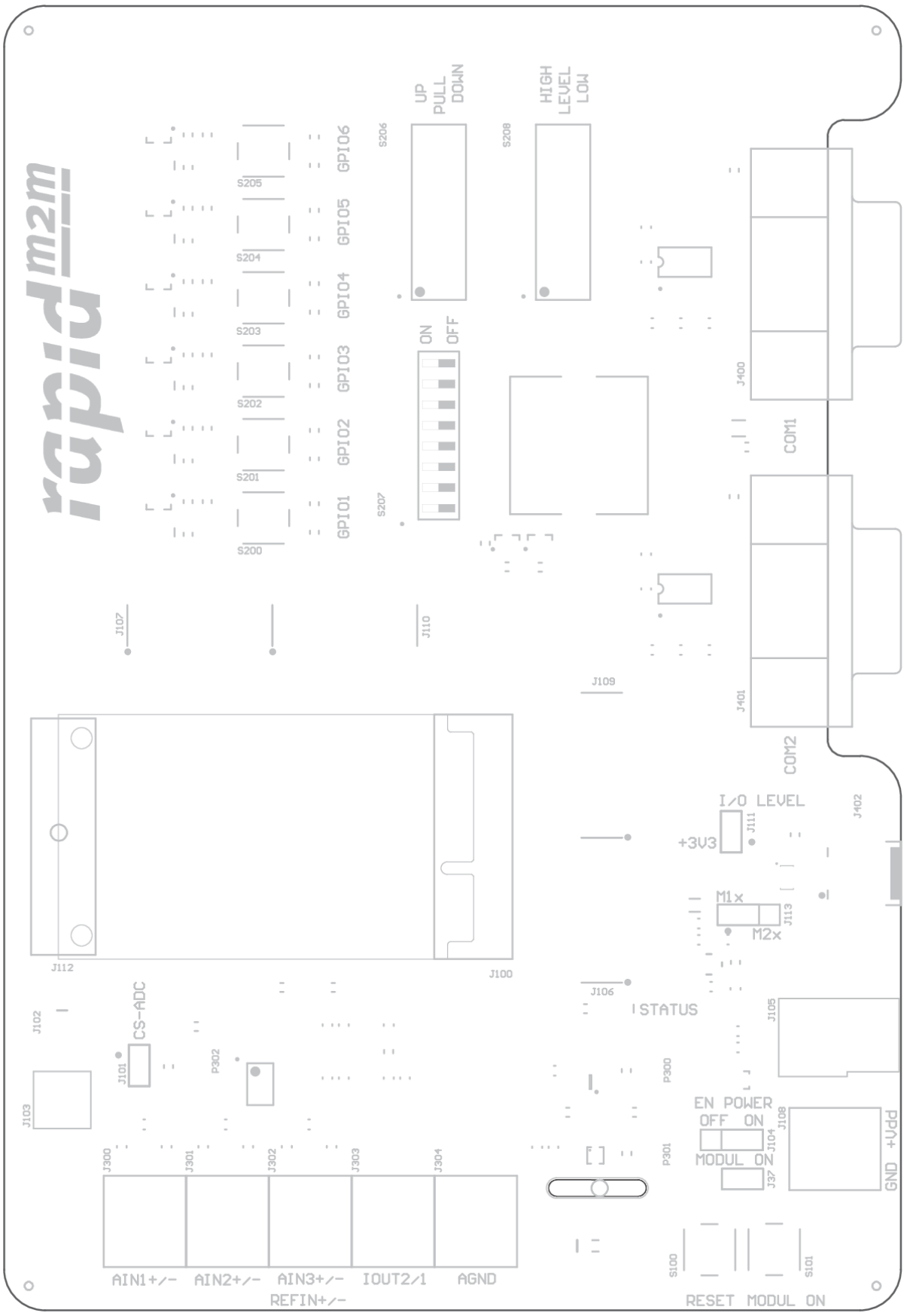
## 6.2.4 Sensoren



## 6.2.5 GPIO


GPIO1.6
GPIO1
GPIO2
GPIO3
GPIO4
GPIO5
GPIO6
µC intern 50k pull up auf den GPIOs
S207
44812127808
LS200
SMD-13D05
+3V3
GND
LED205
LED204
LED203
LED202
LED201
LED200
agreen
Q206
Q205
Q204
Q203
Q202
Q201
Q200
Q207
FDV303N
R224
R223
R222
R221
R220
R219
R218
R217
R216
R215
R214
R213
R212
R232
R230
R229
R228
R227
R226
R225
R231
220R
100R
1M
470R
S205
S204
S203
S202
S201
S200
R211
R210
R209
R208
R207
R206
10R
TTL 6V
S208
LOW LEVEL
OFF
HIGH LEVEL
R200
R201
R202
R203
R204
R205
S206
DOWN
PULL
UP

## 6.3 Layout


UP
PULL
DOWN
HIGH
LEVEL
LOW
ON
OFF
GPIO6
GPIO5
GPIO4
GPIO3
GPIO2
GPIO1
S200
S201
S202
S203
S204
S205
S206
S207
S208
J400
COM1
J401
COM2
J402
I/O LEVEL
J111
+3V3
M1x
M2x
J113
J112
J100
J107
J110
J109
J106
STATUS
J105
CS-ADC
J101
J102
J103
P302
P300
P301
EN POWER
OFF ON
J104
MODUL ON
J37
J108
GND +Vdd
J300
J301
J302
J303
J304
AIN1+/-
AIN2+/-
AIN3+/-
IOUT2/1
AGND
REFIN+/-
S100
S101
RESET MODUL ON

## 6.4 Elektrische Installation

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

### 6.4.1 Hinweise zur Vermeidung elektrostatischer Entladungen (ESE)

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

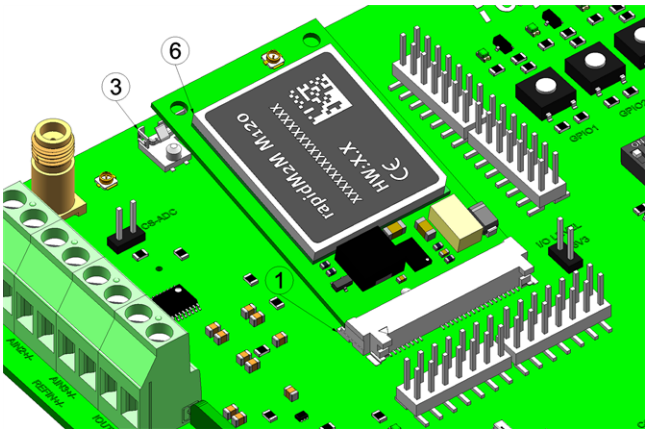
Die empfindlichen elektronischen Komponenten im Geräteinneren können durch statische Elektrizität beschädigt werden, was zur Beeinträchtigung der Geräteleistung bis hin zum Ausfall des Geräts führen kann. Der Hersteller empfiehlt die folgenden Schritte zur Vermeidung von Beschädigungen des Geräts durch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

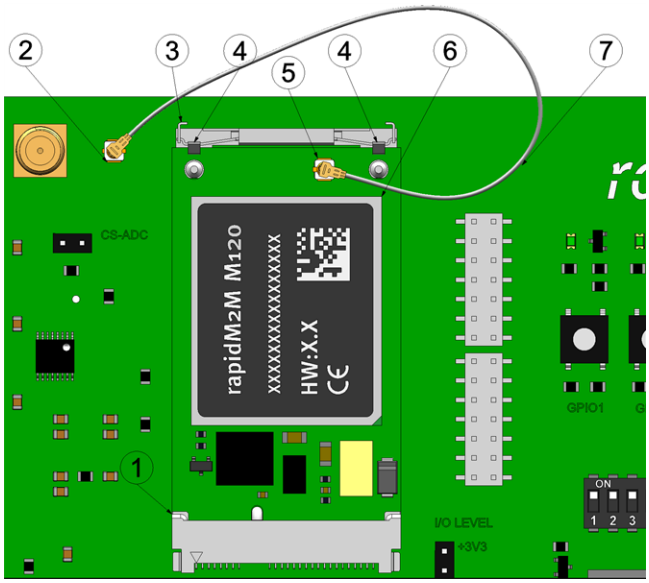
## 6.4.2 Einsetzen des M120 Moduls

***Wichtiger Hinweis:***

- *Das Einsetzen des M120 Moduls muss im stromlosen Zustand erfolgen!*
- *Unsachgemäße Handhabung kann zu Verletzungen und/oder Beschädigungen der Instrumente führen!*




Einsetzen des M120 Moduls Schritt 1




Einsetzen des M120 Moduls Schritt 2

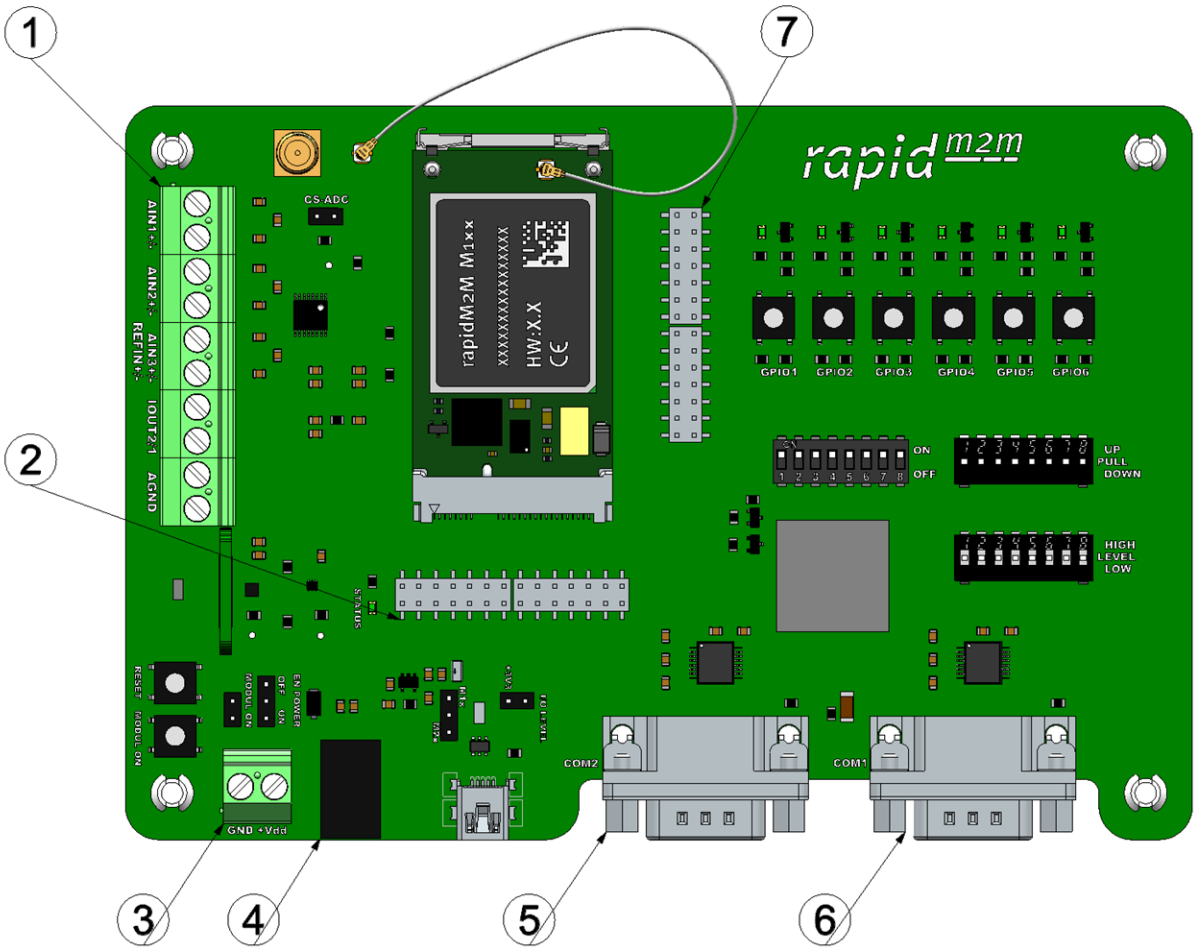
| | | | |
|---|---|---|---|
| 1 | 52pin Steckverbinder (mechanisch kompatibel zu "PCI Express Mini Card") | 5 | U.FL Antennenanschluss des M120 Modul |
| 2 | U.FL Antennenanschluss des Developer Boards (J102) | 6 | M120 Modul |
| 3 | Arretierung | 7 | U.FL Antennenkabel |
| 4 | Haken der Arretierung | | |

1. Setzen Sie das M120 Modul (6) entsprechend der Abbildung "Einsetzen des M120 Moduls Schritt 1 " auf Seite 93 in den 52pin Steckverbinder (1) des Developer Boards ein. Das Modul muss dabei so eingesetzt werden, dass das Typenschild nach dem einrasten sichtbar bleibt.
2. Üben Sie dann auf die am weitesten vom 52pin Steckverbinder (1) entfernte Seite des M120 Moduls so lange einen leichten Druck aus, bis das Modul in die Arretierung (3) einrastet. Zum Lösen des Moduls drücken Sie die beiden Haken der Arretierung (4) nach hinten.
3. Verbinden Sie anschließend den U.FL Antennenanschluss des M120 Moduls (5) mittels des im Lieferumfang enthaltenen U.FL Antennenkabels mit dem U.FL Antennenanschluss des Developer Boards (2).

## 6.4.3 Anschluss der Sensoren, der Aktoren und der Versorgung

***Wichtiger Hinweis:***

- *Alle Verkabelungsarbeiten sollten im stromlosen Zustand erfolgen!*
- *Befolgen Sie bestehende gesetzliche bzw. betriebliche Richtlinien!*
- *Unsachgemäße Handhabung kann zu Verletzungen und/oder Beschädigungen an den Instrumenten führen!*




Anschluss des Sensors, der Aktoren und der Versorgung

| | | | |
|---|---|---|---|
| 1 | Klemmleisten J300-J304: Differenzeingänge und Konstantstromausgänge des AD7792 (U302) | 5 | COM2: 9pol. D-Sub male (J400) |
| 2 | Stiftleiste (J106 und J109) auf der die Pins mit den ungeraden Nummern des M120 Moduls aufgelegt sind | 6 | COM1: 9pol. D-Sub male (J401) |
| 3 | Versorgung: Klemmleiste (J108) | 7 | Stiftleiste (J107 und J110), auf der die Pins mit den geraden Nummern des M120 Moduls aufgelegt sind |
| 4 | Versorgung: DC-Hohlstecker (J105) 5,5/2,1mm (Außen-/Innendurchmesser) | | |

### Klemmleisten J300-J304 (Analoge Differenzeingänge und Konstantstromausgänge)

| | | | |
|---|---|---|---|
| J300 | 1 | AIN1+ | Differenzeingang 1 |
| | 2 | AIN1- | |
| J301 | 1 | AIN2+ | Differenzeingang 2 |
| | 2 | AIN2- | |
| J302 | 1 | AIN3+ / REFIN+ | Differenzeingang 3 / Eingang für die externe Referenzspannung |
| | 2 | AIN3- / REFIN- | |
| J303 | 1 | IOUT2 | Konstantstromausgang 2 |
| | 2 | IOUT1 | Konstantstromausgang 1 |
| J304 | 1 | AGND | Analogmasse |
| | 2 | AGND | Analogmasse |

### Belegung der Klemmleiste (J108)

| | | |
|---|---|---|
| 1 | GND | Masse |
| 2 | $+V_{dd}$ | Versorgungsspannung: 5...32VDC |

### Belegung des DC-Hohlsteckers (J105) 5,5/2,1mm (Außen-/Innendurchmesser)

| | | |
|---|---|---|
| Außenkontakt | GND | Masse |
| Innenkontakt | $+V_{dd}$ | Versorgungsspannung: 5...32VDC |

1. Verbinden Sie zuerst Ihre Sensoren und Aktoren mit dem rapidM2M Developer Board . Achten Sie dabei auf Stromlosigkeit!

   ***Hinweis:*** *Unter Umständen sind für den Anschluss an COM1 (J401) oder COM2 (J400) Adapter erforderlich. Im Kapitel Ersatzteile und Zubehör (siehe "Ersatzteile und Zubehör" auf Seite 303) finden Sie eine Auswahl an verfügbaren Adaptern.*

2. Schließen Sie die Antenne an (siehe "Anschluss der GSM-Antenne" auf Seite 96).
3. Vergewissern Sie sich, dass der Jumper zur Versorgung der Level Shifter (J111) geschlossen ist und beim Jumper zum Aktivieren der Energieversorgung (J104) Pin2 und Pin3 verbunden sind (Position "ON") (siehe "Übersicht" auf Seite 67).
4. Versorgen können Sie das rapidM2M Developer Board entweder über den DC-Hohlstecker (4) oder die Klemmleiste (3). Verwenden Sie dazu entweder das im Lieferumfang enthaltene Steckernetzgerät oder jedes beliebige Netzgerät, das eine Spannung im Bereich von 5...32VDC liefert. Soll die Versorgung des Developer Boards über die Klemmleiste (3) erfolgen, sollten Sie die Kabel im stromlosen Zustand mit der Klemmleiste (3) verbinden.

5. Schalten Sie jetzt die Spannungsversorgung der Sensoren, der Aktoren und des Developer Boards ein.

   Sollte der Jumper zum Einschalten des Wireless GSM-Controllers (J37 "MODUL ON") geschlossen sein (Auslieferungszustand, siehe "Übersicht" auf Seite 67), startet der Controller des M120 Moduls nach dem Herstellen der Energieversorgung des Developer Boards sobald er das "Modul_ON" Signal erkannt hat (ca. 1,5sec. nach Herstellung der Energieversorgung). Daraufhin sollte die Status-LED ca. 1sec. leuchten, um den Boot-Vorgang anzuzeigen (siehe "Status-LED (LED100)" auf Seite 80). Anschließend bucht sich das rapidM2M Developer Paket M12x ins Netz ein und nimmt die Verbindung zum Cloud-Server auf.

Der folgende Schritt ist nur erforderlich, wenn der Jumper zum Einschalten des Wireless GSM-Controllers (J37 "MODUL ON") nicht geschlossen ist.

6. Drücken Sie als nächstes die Taste zum Einschalten des Wireless GSM Controllers S101 (siehe "Taster zum Einschalten des Wireless GSM-Controllers am rapidM2M Modul M120 (S101)" auf Seite 81). Daraufhin sollte die Status-LED ca. 1sec. leuchten um den Boot-Vorgang anzuzeigen (siehe "Status-LED (LED100)" auf Seite 80). Anschließend bucht sich das rapidM2M Developer Paket M12x ins Netz ein und nimmt die Verbindung zum Cloud-Server auf.

Der folgende Schritt ist nicht zwingend erforderlich.

7. Überprüfen Sie, ob die Verbindung zum Cloud-Server korrekt funktioniert hat (siehe "Kommunikation mit dem Gerät testen" auf Seite 190).

### 6.4.4 Anschluss der GSM-Antenne

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

Die im Lieferumfang enthaltene Magnetfußantenne SMA-M 2,5m (300085) wird direkt mit dem Antennenstecker (siehe "Übersicht" auf Seite 67) des Developer Boards verbunden.

Sollte die Länge des 2,5m Anschlusskabels der Magnetfußantenne SMA-M 2,5m (300085) nicht ausreichen, können Sie eine 2,5m Antennenverlängerung SMA-M/SMA-F 2,5 m (206.807) verwenden.

1. Stellen Sie sicher, dass das rapidM2M Developer Board spannungslos ist.
2. Verbinden Sie, falls Sie eine Antennenverlängerung benötigen, diese zuerst mit der Antenne.
3. Verbinden Sie die Antennenverlägerung bzw. die Antenne direkt mit dem Antennenanschluss des Developer Boards (siehe "Übersicht" auf Seite 67).

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

4. Stellen Sie die Spannungsversorgung des Developer Boards wieder her.

Der folgende Schritt ist nur erforderlich, wenn der Jumper zum Einschalten des Wireless GSM-Controllers (J37 "MODUL ON") nicht geschlossen ist.

5. Drücken Sie als nächstes die Taste zum Einschalten des Wireless GSM Controllers S101 (siehe "Taster zum Einschalten des Wireless GSM-Controllers am rapidM2M Modul M120 (S101)" auf Seite 81).

Der folgende Schritt ist nicht zwingend erforderlich.

6. Überprüfen Sie, ob die Verbindung zum Cloud-Server korrekt funktioniert hat (siehe "Kommunikation mit dem Gerät testen" auf Seite 190).

6872
6872
6872

# Kapitel 7 rapidM2M Modul M120

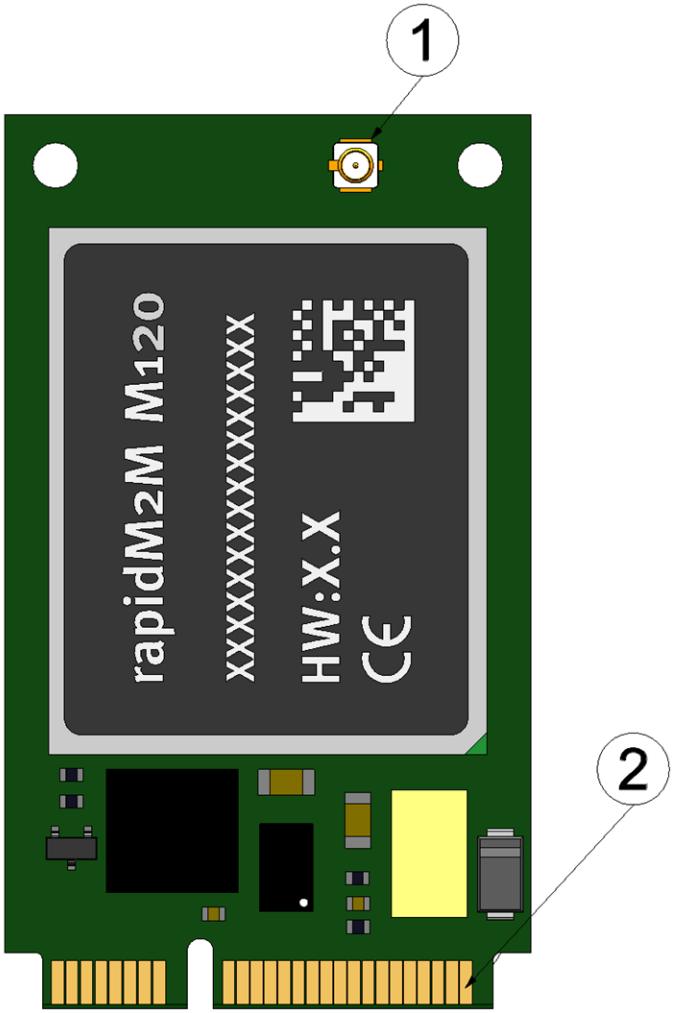
## 7.1 Technische Daten

| | |
|---|---|
| Spannungsversorgung | Versorgung über $V_{IN}$: 4,5...32VDC , max. 100mA @ 12V DC[1)]<br>oder<br>Versorgung über $V_{AUX}$: 3,2...4,5V , max. 294mA @ 3,8V DC[1)]<br><br>Zusätzliche Informationen finden Sie unter "Technische Details zur Energieversorgung" auf Seite 107. |
| Ausgangsspannung<br>(Nur bei Versorgung über $V_{IN}$) | 3,65V , max. 300mA |
| Eingangsbereich der Pegelwandler für I/Os | 2,8...5,5V<br><br>Zusätzliche Informationen finden Sie unter "Technische Details zu den Level Shiftern" auf Seite 110. |
| Gehäuse | Gewicht: 11g<br>Interface: 52pin Steckerleiste<br>Abmessungen (BHT): 30 x 5,5 x 51mm (ohne Antenne) |
| Betriebstemperatur | -20...+60°C, 15...90%rH nicht kondensierend |
| Lagertemperatur | -40...+85°C |
| Anzeige | Open Drain Ausgang zum Anschluss einer Status-LEDs<br><br>Zusätzliche Informationen finden Sie unter "Status-LED (LED100)" auf Seite 80. |
| Antennenanschluss | U.FL |

[1)] Die Werte wurden bei einer GSM-Signalstärke von -60dBm ermittelt. Während einer aktiven GSM Daten-Übertragung sind auch höhere Stromwerte möglich. Der Spitzenstrom hängt von der GSM Signalstärke, der Nutzdatenmenge und der Übertragungsgeschwindigkeit ab.

| | |
|---|---|
| Serielle Schnittstellen | 2 x 4-Wire UART<br>• Baudrate: 300-115200<br>• Stopbits: 1, 2<br>• Parität:N, E, O<br>• Datenbits: 7, 8<br><br>1 x SPI<br>• Modus: SPI-Master<br>• Clock-Polarität/Phase einstellbar<br>• Clock-Frequenz: 102kHz...13MHz<br>• Interface: 4-wire<br><br>1 x I²C<br>• Modus: Master<br>• Clock-Frequenz: 100kHz...400kHz<br><br>Zusätzliche Informationen finden Sie unter:<br>• "UART" auf Seite 110<br>• "SPI" auf Seite 111<br>• "I2C" auf Seite 112 |
| USB-Schnittstelle | 1 x Mini-B USB 2.0 Slave für die Verbindung mit einem PC. Für die Kommunikation mit dem rapidM2M Modul M120 muss am PC das rapidM2M Toolset installiert sein.<br><br>Zusätzliche Informationen finden Sie unter "USB-Schnittstelle" auf Seite 80. |
| I/O-Leitungen | 6 x GPIO<br>2 x Interrupt-Eingänge |
| Datenspeicher | 3 x 349.525 Byte interner Flash-Speicher<br>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. |
| Konfigurationsspeicher | 10 unabhängige Blöcke mit je 10kB |
| Registrierungsspeicher | 4 Blöcke mit je 1kB und vordefinierten Verwendungszwecken zur Ablage gerätespezifischer Daten<br><br>Zusätzliche Informationen finden Sie unter "Registrierungsspeicherblöcke" auf Seite 63. |
| Speicher für das PAWN-Binary | 256kB (unkomprimierte Größe)<br><br>Zusätzliche Informationen finden Sie unter "Speicherorganisation" auf Seite 105. |
| Datenübertragung | mittels GSM/GPRS Quad-Band Modem an den jeweiligen Cloud-Server |
| SIM | Das rapidM2M Modul M120 ist mit einem integrierten SIM-Chip versehen. |

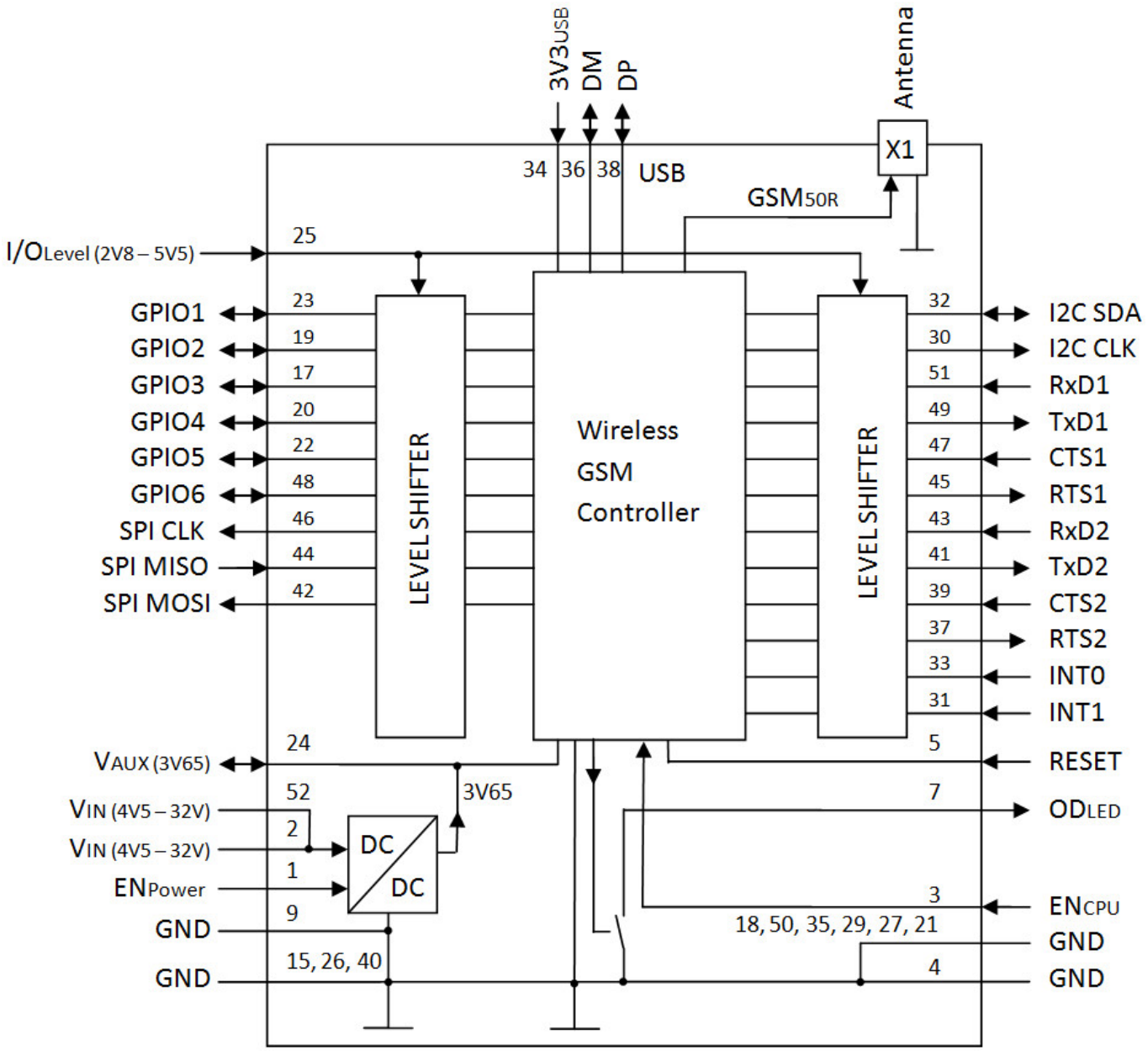
## 7.2 Übersicht




Übersicht M120 Modul

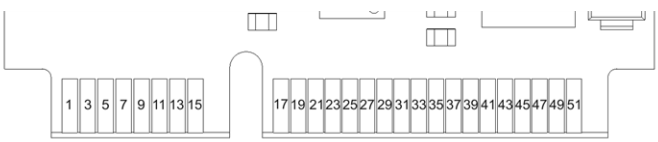
| | |
|---|---|
| **1** U.FL Antennenanschluss (X1) | **2** 52pin Steckerleiste (X2) |

## 7.2.1 Blockschaltbild

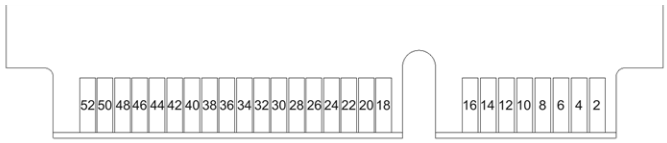



Blockschaltbild des M120 Moduls

## 7.2.2 52pin Steckerleiste (X2)




Steckerleiste Detailansicht (Ansicht von oben)

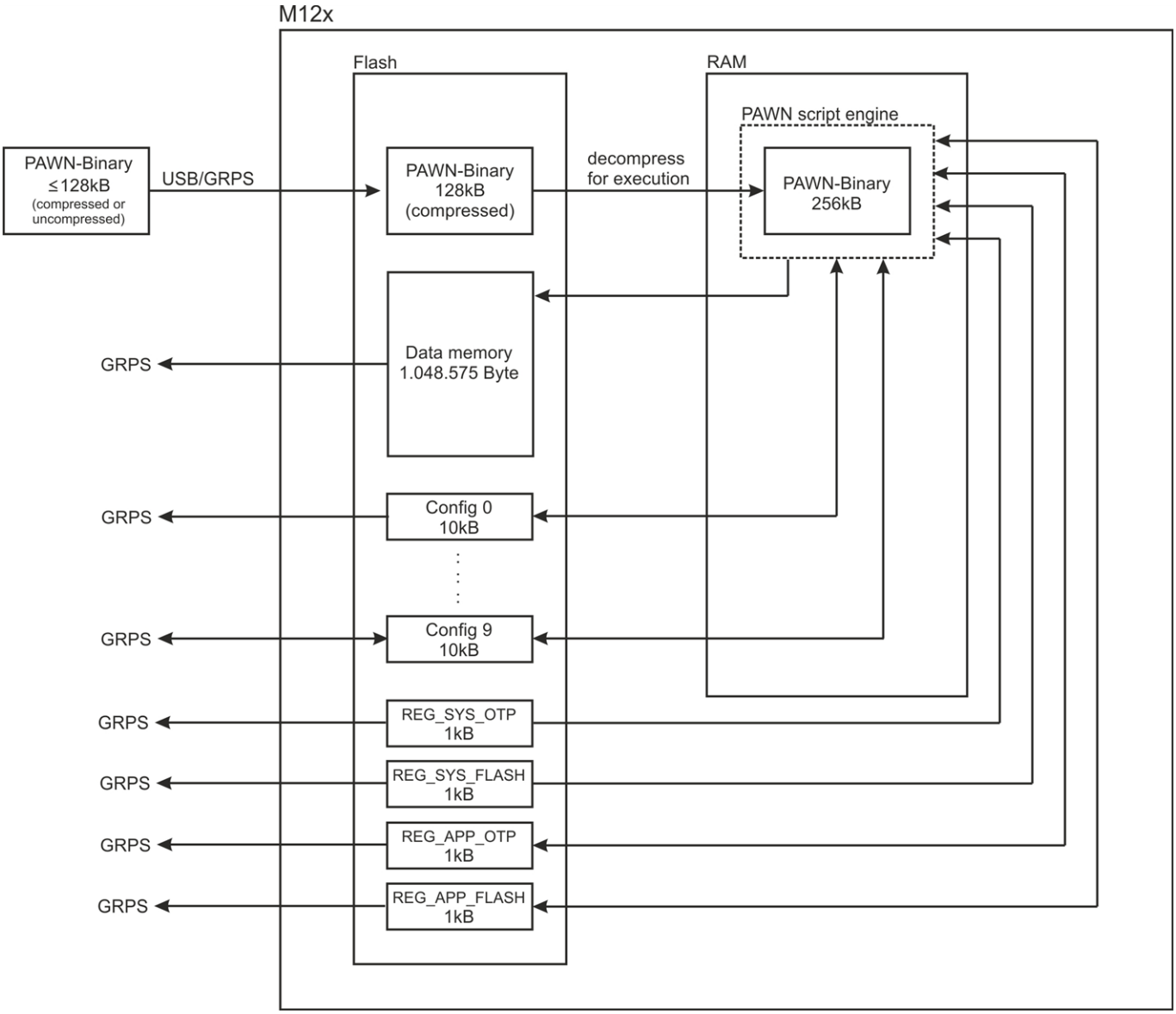



Steckerleiste Detailansicht (Ansicht von unten)

| Pin | Typ | Bezeichnung | Beschreibung |
|---|---|---|---|
| 1 | I | $EN_{Power}$ | Pin zum Aktivieren des DC/DC-Wandlers, der aus der an $V_{IN}$ angelegten Spannung die interne Versorgungsspannung $V_{AUX}$ erzeugt |
| 2, 52 | I | $V_{IN}$ | Versorgungsspannung (4,5...32VDC ) für das M120 Modul |
| 3 | I | $EN_{CPU}$ | Pin zum Einschalten des Wireless GSM Controllers am M120 Modul |
| 4, 9, 15, 18, 21, 26, 27, 29, 35, 40, 50 | | GND | Masse |
| 5 | I | Reset | Pin zum Reseten des Wireless GSM Controllers am M120 Modul |
| 6, 8, 10, 12, 14, 16, 28 | | NC | reserviert für Erweiterungen<br>Es dürfen keine Signale an diese Pins angelegt werden. |
| 7 | O | $OD_{LED}$ | Open Drain Ausgang zum Anschluss einer Status-LEDs |
| 11, 13, | | | reserviert für Erweiterungen<br>Es dürfen keine Signale an diese Pins angelegt werden. |
| 17 | I/O | GPIO3 | GPIO zum Austausch von Signalen zw. M120 und der umgebenden Hardware |
| 19 | I/O | GPIO2 | |
| 20 | I/O | GPIO4 | |
| 22 | I/O | GPIO5 | |
| 23 | I/O | GPIO1 | |
| 24 | I/O | $V_{AUX}$ | 3,65V Spannungsausgang, wenn das M120 Modul über $V_{IN}$ versorgt wird<br>Versorgungsspannungseingang (3,2...4,5V ), wenn an $V_{IN}$ keine Spannung angelegt wird |
| 25 | I | I/O Level | 2,8...5,5V Versorgung der Level Shifter zur Anpassung der Signalpegel des M120 Moduls an die umgebende Hardware |
| 30 | O | I2C CLK | Clock-Leitung der $I^2C$-Schnittstelle |
| 31 | I | INT1 | Eingang über den die umgebende Hardware Interrupts am M120 Modul auslösen kann. |
| 32 | I/O | I2C SDA | Daten-Leitung der $I^2C$-Schnittstelle |
| 33 | I | INT0 | Eingang über den die umgebende Hardware Interrupts am M120 Modul auslösen kann. |

| Pin | Typ | Bezeichnung | Beschreibung |
|---|---|---|---|
| 34 | I | $3V3_{USB}$ | USB-Versorgung<br>ACHTUNG: An diesem Pin sind nur 3,3V anstelle der sonst üblichen 5V zulässig. |
| 36 | I/O | DM | "Data-"-Leitung der USB-Schnittstelle |
| 37 | O | RTS2 | RTS Leitung der UART2 |
| 38 | I/O | DP | "Data+"-Leitung der USB-Schnittstelle |
| 39 | I | CTS2 | CTS-Leitung der UART2 |
| 41 | O | TxD2 | TxD-Leitung der UART2 |
| 42 | O | SPI MOSI | "Master out, Slave in"-Leitung der SPI-Schnittstelle |
| 43 | I | RxD2 | RxD-Leitung der UART2 |
| 44 | I | SPI MISO | "Master in, Slave out"-Leitung der SPI-Schnittstelle |
| 45 | O | RTS1 | RTS-Leitung der UART1 |
| 46 | O | SPI CLK | Clock-Leitung der SPI-Schnittstelle |
| 47 | I | CTS1 | CTS-Leitung der UART1 |
| 48 | I/O | GPIO6 | GPIO zum Austausch von Signalen zw. M120 und der umgebenden Hardware |
| 49 | O | TxD1 | TxD-Leitung der UART1 |
| 51 | I | RxD1 | RxD-Leitung der UART1 |

## 7.2.3 Speicherorganisation




Organisation des Speichers des M120 Moduls

Für die Übertragung ins rapidM2M Modul M120 darf die Größe des PAWN-Binarys 128kB nicht überschreiten. Falls nötig, kann das PAWN-Binary mittels Compiler-Anweisung (#pragma amxcompress ‹0-3›) komprimiert werden. Im Flash Speicher des M120 Moduls sind das PAWN-Binary (128kB ), die 10 Konfigurationsblöcke (je 10kB ), die 4 Registrierungsspeicherblöcke (je 1kB ) und die Messdaten (3 x 349.525 Byte) abgelegt. Für die Ausführung durch die PAWN Script-Engine wird das PAWN-Binary falls nötig dekomprimiert und ins RAM kopiert. Die maximale Größe für das dekomprimierte PAWN-Binary im RAM beträgt 256kB .

### 7.2.4 Gerätekennzeichnung

Das Typenschild befindet sich auf der Oberseite des Moduls und beinhaltet folgende Angaben:

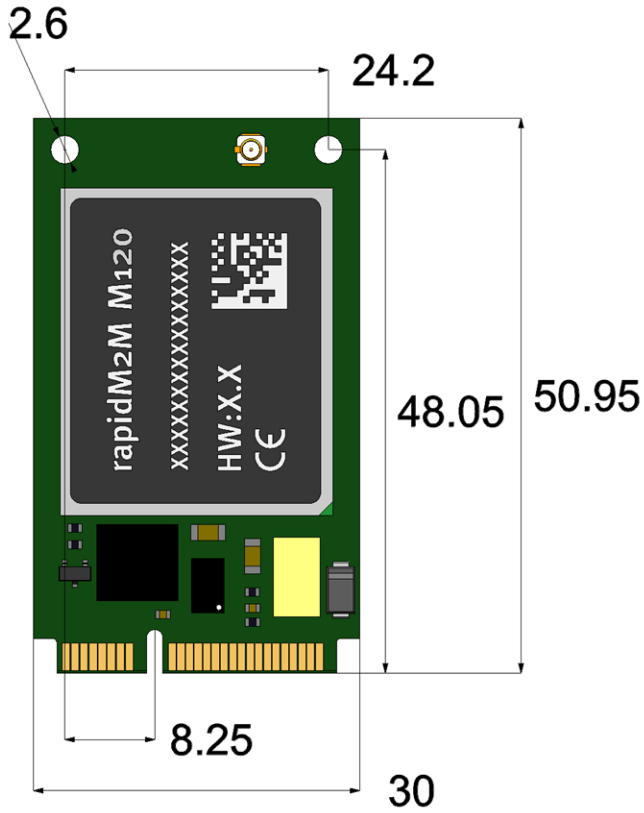
- Typenbezeichnung
- Seriennummer
- CE-Kennzeichnung
- Hardwarerevision



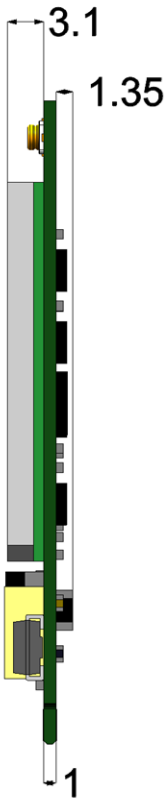

Typenschild des M120 Moduls

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

## 7.3 Abmessungen

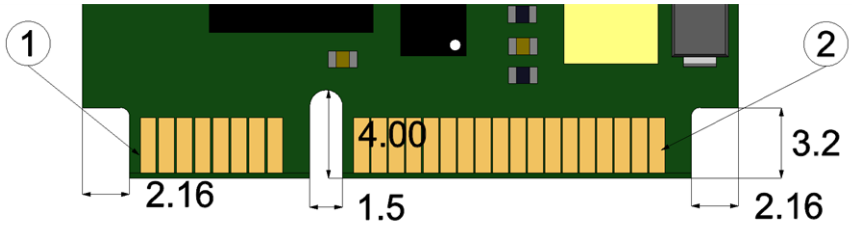



Abmessungen: Breite und Tiefe (Ansicht von oben)




Abmessungen: Höhe

### 7.3.1 Abmessungen der 52pin Steckerleiste (X2)




Steckerleiste Detailansicht (Ansicht von oben)

| | |
|---|---|
| 1 Pin 1 | 2 Pin 51 |

## 7.4 Technische Details

### 7.4.1 Technische Details zur Energieversorgung

Abhänig von der zur Verfügung stehenden Spannung gibt es zwei Möglichkeiten das M120 Modul zu versorgen:

- $V_{AUX}$: 3,2...4,5V
- $V_{IN}$: 4,5...32VDC

| Symbol | Parameter | Bedingung | min. | typ. | max. |
|---|---|---|---|---|---|
| $V_{IN}$ | Eingangsspannung | Versorgung über $V_{IN}$ | 4,5V | | 32V |
| | | Versorgung über$V_{AUX}$ | --- | --- | --- |
| $I_{IN}$ | Eingangsstrom | Versorgung über $V_{IN}$ | | 100mA @ 12V DC[1)] | |
| | | Versorgung über$V_{AUX}$ | --- | --- | --- |
| $\eta_{DC/DC\ VIN}$ | Wirkungsgrad des DC/DC-Wandlers | Versorgung über $V_{IN}$ | | 94% | |
| $I_{Q\ DC/DC\ VIN}$ | Ruhestrom des DC/DC-Wandlers | Versorgung über $V_{IN}$ | | 2,5µA | |
| $V_{AUX}$ | Ausgangsspannung | Versorgung über $V_{IN}$ | | 3,65V | |
| | Eingangsspannung | Versorgung über$V_{AUX}$ | 3,2V | | 4,5V |
| $I_{AUX}$ | Ausgangsstrom | Versorgung über $V_{IN}$ | | | 300mA |
| | Eingangsstrom | Versorgung über$V_{AUX}$ | | 294mA @ 3,8V DC[1)] | |

[1)] Die Werte wurden bei einer GSM-Signalstärke von -60dBm ermittelt. Während einer aktiven GSM Daten-Übertragung sind auch höhere Stromwerte möglich. Der Spitzenstrom hängt von der GSM Signalstärke, der Nutzdatenmenge und der Übertragungsgeschwindigkeit ab.

#### 7.4.1.1 Versorgung über $V_{IN}$

Die interne Versorgungsspannung (3,65V ) wird vom DC/DC-Wandler (94% Wirkungsgrad, 2,5µA Ruhestrom) des M120 Moduls (siehe "Blockschaltbild" auf Seite 102) erzeugt. Dieser muss dazu allerdings über den Pin "$EN_{Power}$" aktiviert werden. Um die Belastung der Energiequelle beim Einschalten gering zu halten, ist der DC/DC-Wandler mit einer Soft-Start-Funktion ausgestattet. Dadurch dauert es 50ms bis die interne Versorgungsspannung den Maximalwert erreicht. Wird das M120 Modul über $V_{IN}$ versorgt, fungiert der Pin "$V_{AUX}$" als Spannungsausgang für die interne 3,65V Versorgungsspannung.

#### 7.4.1.2 Versorgung über $V_{AUX}$

Der DC/DC-Wandler des M120 Moduls (siehe "Blockschaltbild" auf Seite 102) wird übergangen und die Versorgungsspannung direkt über den Pin "$V_{AUX}$" in das Modul eingespeist.

### 7.4.2 $EN_{Power}$-Signal

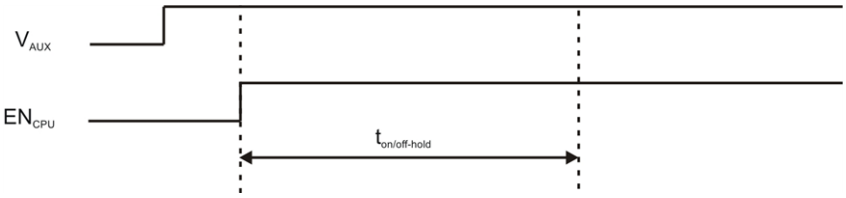
| Parameter | min. | typ. | max. |
|---|---|---|---|
| $V_{POWER\ th}$ | 0,94V | 1V | 1,06V |
| $V_{POWER\ hyst}$ | | 40mV | |
| $t_{soft\ start}$ | | 50ms | |

Das $EN_{Power}$-Signal ist nur erforderlich, wenn die Versorgung über $V_{IN}$ erfolgt. Um den DC/DC-Wandler zur Erzeugung der internen Versorgungsspannung und der Ausgangsspannung $V_{AUX}$ (3,65V ) zu aktiveren, muss die Spannung am Pin "$EN_{Power}$" über "$V_{POWER\ th}$" (1V ) liegen. Um die Belastung der Energiequelle beim Einschalten gering zu halten, ist der DC/DC-Wandler mit einer Soft-Start-Funktion ausgestattet. Dadurch dauert es 50ms bis die interne Versorgungsspannung den Maximalwert erreicht.

### 7.4.3 $EN_{CPU}$-Signal

| Parameter | min. | typ. | max. |
|---|---|---|---|
| $V_{IL}$ | | | $V_{AUX}$ * 0,2 |
| $V_{IH}$ | $V_{AUX}$ * 0,8 | | $V_{AUX}$ |
| $t_{on/off-hold}$ | 1,5sec. | | |
| $t_{rst}$ | | 40ms | |

Durch ein "High" für mindestens 1,5sec. ($t_{on/off-hold}$) am Pin "$EN_{CPU}$" wird der Wireless GSM Controller eingeschaltet. Der Pin "$EN_{CPU}$" sollte danach, solange das M120 Modul aktiv ist, auf dem "High" Pegel bleiben.



Power-ON Sequenz des Wireless GSM Controllers

Während der Startphase führt der Wireless GSM Controller einen internen Reset durch. Dieser dauert typischerweise 40ms ($t_{rst}$) und sollte nicht durch einen externen Reset (Pin "Reset") unterbrochen werden.

Ein Abschalten des Wireless GSM Controllers ist bei der aktuellen Firmware des M120 Moduls nicht vorgesehen.

## 7.4.4 Reset-Signal

| Parameter | min. | typ. | max. |
|---|---|---|---|
| $V_{IL}$ | 0V | | 0,57V |
| $V_{IH}$ | 1,33V | | |
| Reset time | 200µs | | |
| Int. pull-up | | 100kΩ | |

Um einen Reset auszulösen, muss der Pin "Reset" für min. 200µs mittels Open Collector oder Open Drain auf "Low" gesetzt werden. Der Pin "Reset" sollte nur als Notfall-Reset betrachtet werden, da eine Reset-Prozedur vom Wireless GSM Controller während des Boot-Vorgangs selbstständig durchgeführt wird.

Der Wireless GSM Controller bleibt, solange der Pin "Reset" auf "Low" ist, im Reset-Zustand.

Nach einem Reset ist der Wireless GSM Controller im Zustand "OFF", d.h. er muss mittels $EN_{CPU}$-Signal (siehe "ENCPU-Signal" auf Seite 108) wieder eingeschaltet werden. Der Pin "$EN_{CPU}$" kann aber während des Resets auf "High" verbleiben, wodurch der Wireless GSM Controller nachdem er das Signal am Pin "$EN_{CPU}$" erkannt hat (mind. 1,5sec. ) sofort wieder startet.

## 7.4.5 Status-LED

Das rapidM2M Modul M120 ist mit einem Open Drain Ausgang zum Anschluss einer Status-LEDs (Pin "$OD_{LED}$"), die während des Boot-Vorgangs ca. 1sec. leuchtet, ausgestattet.

Eine LED und ein Widerstand können direkt zwischen $V_{AUX}$ und dem Open Drain Ausgang $OD_{LED}$ geschaltet werden.

| Parameter | min. | typ. | max. |
|---|---|---|---|
| $V_{OL}$ | | | 0,4V |
| $I_{out}$ | | | 8mA |

## 7.4.6 Technische Details zu den Level Shiftern

*Hinweis: Die bidirektionalen Level Shifter mit automatischer Richtungserkennung, in "open drain"-Bauweise sind nicht dafür ausgelegt größere Lasten zu treiben.*

| Parameter | min. | typ. | max. |
|---|---|---|---|
| $V_{I/O\ Level}$ | 2,8V | | 5,5V |
| $I_{I/O\ Level}$ (bei $V_{I/O\ Level} = V_{AUX}$) | | 816µA | |
| $V_{IL}$ | | | 0,15V |
| $V_{IH}$ | $V_{I/O\ Level}$ - 0,4V | | |
| $V_{OL}$ (bei $I_O$=1mA ) | | | 0,4V |
| $V_{OH}$ (bei $I_O$=-20µA ) | 0,67*$V_{I/O\ Level}$ | | |

Um die Signalpegel des M120 Moduls an die umgebende Hardware anzupassen, ist das Modul mit Level Shifter für alle I/O-Signale ausgestattet. Für die Anpassung muss von der umgebenden Hardware eine zusätzliche Versorgungsspannung für die Level Shifter im Bereich von 2,8...5,5V bereit gestellt werden. Über den Pin "I/O Level" kann entweder eine Fremdspannung oder die vom M120 Modul am Pin "$V_{AUX}$" ausgegebene Spannung eingespeist werden (siehe "Blockschaltbild" auf Seite 102).

## 7.4.7 Serielle Schnittstellen

Die Pegel der seriellen Schnittstellen des M120 Moduls sind abhängig von der am Pin "I/O Level" eingespeisten Versorgungsspannung für die Level Shifter (siehe "Blockschaltbild" auf Seite 102).

### 7.4.7.1 UART

| | |
|---|---|
| Baudrate | 300-115200 |
| Stopbits | 1, 2 |
| Parität | N, E, O |
| Datenbits | 7, 8 |

Ein Script-Beispiel für die Verwendung der UART-Schnittstelle finden Sie im Kapitel t.b.d..

#### 7.4.7.1.1 Pin-Beschreibung

Am Anschluss COM1 (J400) des Developer Boards ist die UART1 des M120 Moduls zugängig. Am Anschluss COM2 (J401) des Developer Boards ist die UART2 des M120 Moduls zugängig. Die Umwandlung auf den RS232-Pegel erfolgt dabei jeweils durch einen Pegelwandler, der sich auf dem rapidM2M Developer Board befindet (siehe "Blockschaltbild" auf Seite 70).

| Pin | Typ | Bezeichnung | Beschreibung |
|---|---|---|---|
| 37 | O | RTS2 | RTS Leitung der UART2 |
| 39 | I | CTS2 | CTS-Leitung der UART2 |
| 41 | O | TxD2 | TxD-Leitung der UART2 |
| 43 | I | RxD2 | RxD-Leitung der UART2 |
| 45 | O | RTS1 | RTS-Leitung der UART1 |
| 47 | I | CTS1 | CTS-Leitung der UART1 |
| 49 | O | TxD1 | TxD-Leitung der UART1 |
| 51 | I | RxD1 | RxD-Leitung der UART1 |

#### 7.4.7.2 SPI

| | |
|---|---|
| Modus | SPI-Master |
| Clock-Polarität/Phase | einstellbar |
| Clock-Frequenz | 102kHz...13MHz |
| Interface | 4-wire |

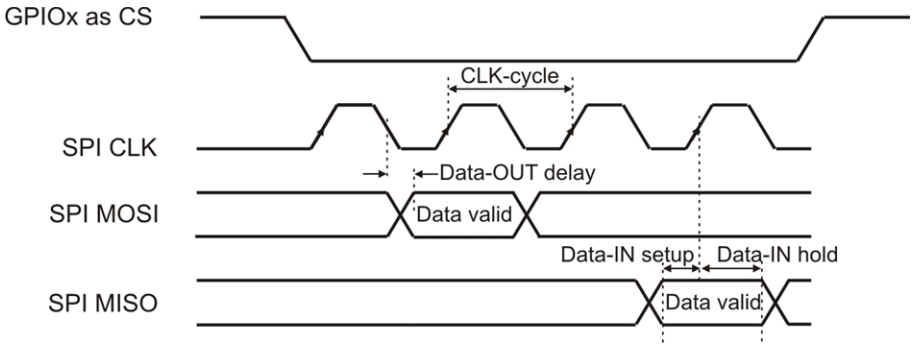
Für das CS-Signal müssen die GPIOs verwendet werden.

Ein Script-Beispiel für die Verwendung der SPI-Schnittstelle finden Sie im Kapitel t.b.d..

##### 7.4.7.2.1 Pin-Beschreibung

| Pin | Typ | Bezeichnung | Beschreibung |
|---|---|---|---|
| 17 | I/O | GPIO3 | GPIOs, die für das CS-Signal verwendet werden können |
| 19 | I/O | GPIO2 | |
| 20 | I/O | GPIO4 | |
| 22 | I/O | GPIO5 | |
| 23 | I/O | GPIO1 | |
| 42 | O | SPI MOSI | "Master out, Slave in"-Leitung der SPI-Schnittstelle |
| 44 | I | SPI MISO | "Master in, Slave out"-Leitung der SPI-Schnittstelle |
| 46 | O | SPI CLK | Clock-Leitung der SPI-Schnittstelle |
| 48 | I/O | GPIO6 | GPIO, der für das CS-Signal verwendet werden kann |

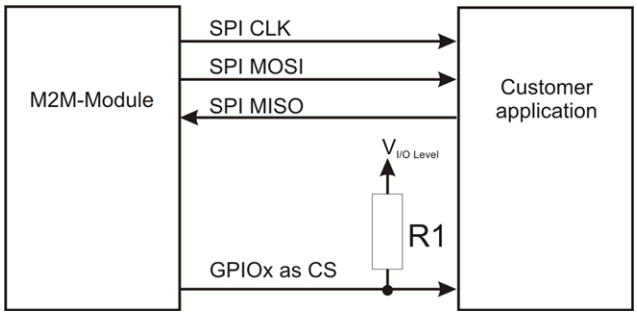
### 7.4.7.2.2 SPI Waveforms




SPI Timing-Diagramm (Mode 0, Master, 4-wire)

| Parameter | min. | typ. | max. |
|---|---|---|---|
| Data-Out delay | 3ns | | 15,5ns |
| Data-IN setup | 5ns | | |
| Data-IN hold | 5ns | | |
| CLK-cycle | 102kHz...13MHz | | |

### 7.4.7.2.3 Applikation




Anwendungsbeispiel

Es wird ein Pull-up-Widerstand R1 (min. 10kΩ , typ. 1MΩ ) benötigt, der während des Reset-Zustands des M120 Moduls die CS-Leitung auf "high" hält.

### 7.4.7.3 I²C

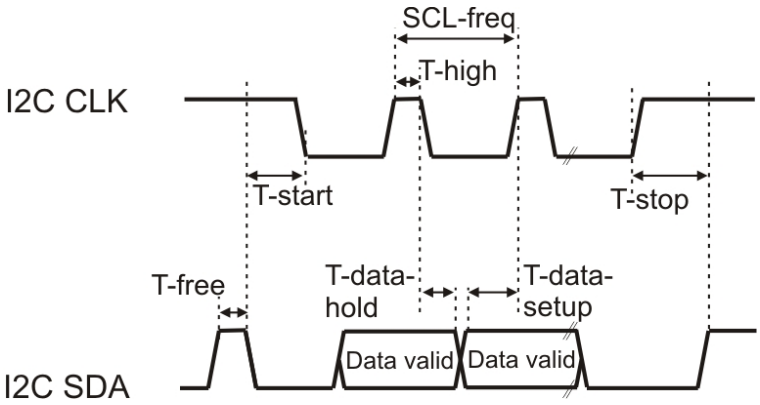
| | |
|---|---|
| Modus | Master |
| Clockfrequenz | 100kHz...400kHz |

Ein Script-Beispiel für die Verwendung der I²C-Schnittstelle finden Sie im Kapitel t.b.d..

#### 7.4.7.3.1 Pin-Beschreibung

| Pin | Typ | Bezeichnung | Beschreibung |
|---|---|---|---|
| 30 | O | I2C CLK | Clock-Leitung der I²C-Schnittstelle |
| 32 | I/O | I2C SDA | Daten-Leitung der I²C-Schnittstelle |

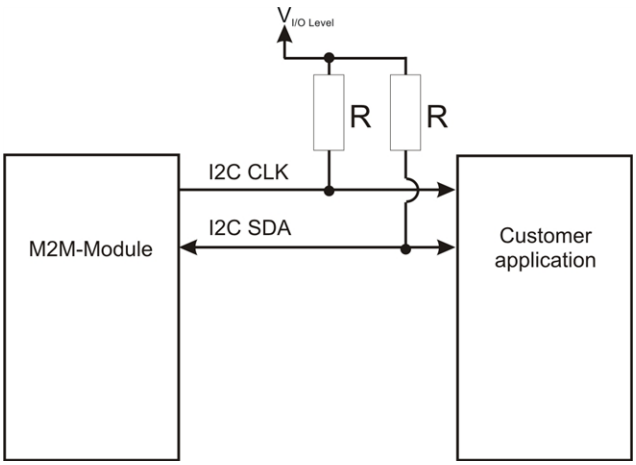
### 7.4.7.3.2 I²C Waveforms




I²C Timing Diagramm (Master)

| Parameter | min. | typ. | max. |
|---|---|---|---|
| T-start | 0,6µs | | |
| T-stop | 0,6µs | | |
| T-free | 1,3µs | | |
| T-high | 0,6µs | | |
| T-data-hold | | | 0,9µs |
| T-data-setup | 0,1µs | | |
| SCL-freq | 100kHz...400kHz | | |

### 7.4.7.3.3 Applikation




Anwendungsbeispiel

Sowohl die CLK- als auch die SDA-Leitung benötigen einen Pull-up. Dieser sollte nicht nicht kleiner als 10kΩ sein.

### 7.4.8 GPIO

Das M120 Modul verfügt über 6  GPIOs zum Austausch von Signalen mit der umgebenden Hardware. Die Pegel der GPIOs sind abhängig von der am Pin "I/O Level" eingespeisten Versorgungsspannung für die Level Shifter (siehe "Blockschaltbild" auf Seite 102).

Ein Script-Beispiel für die Verwendung der GPIO finden Sie im Kapitel t.b.d..

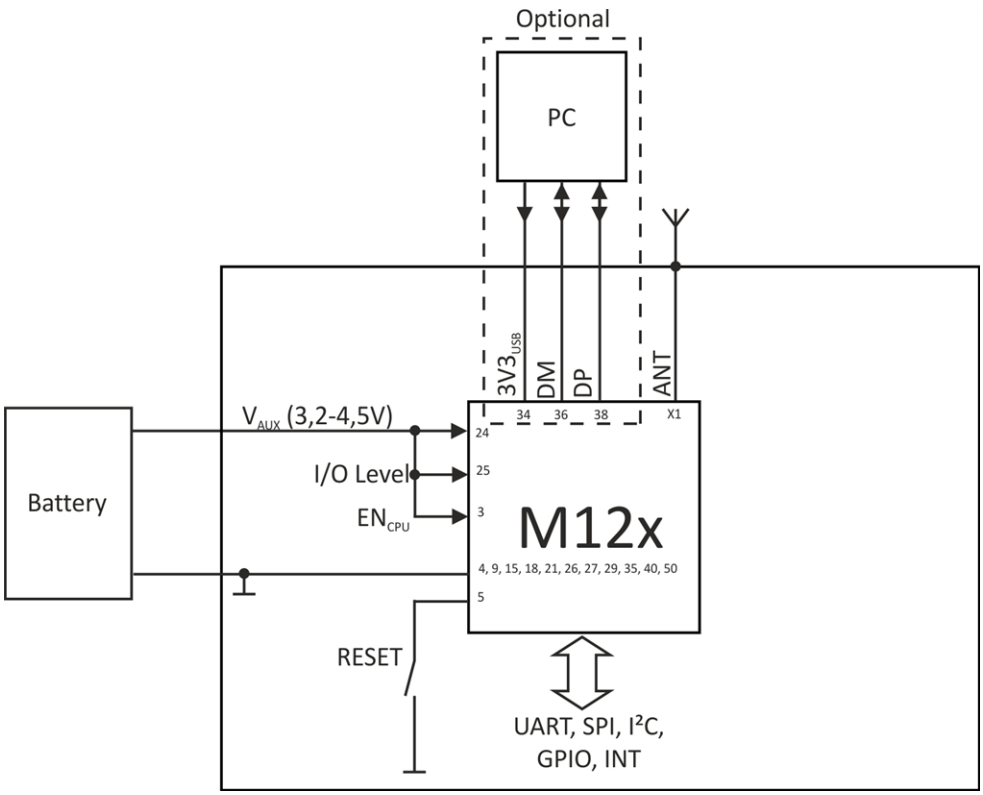
### 7.4.9 Interrupt Eingänge

Das M120 Modul verfügt über 2 Interrupt Eingänge für zeitkritische Anwendungen. Es handelt sich dabei um von den GPIOs völlig unabhängige Eingänge. Sie können ausschließlich dazu verwendet werden, um die Ausführung der mittels der Funktion "rM2M_IrqInit()" festgelegten Funktion auszulösen. Es ist nicht möglich, den Zustand ("low" oder "high") an diesen Eingängen per Script auszulesen. Die Pegel der Interrupt Eingänge sind abhängig von der am Pin "I/O Level" eingespeisten Versorgungsspannung für die Level Shifter (siehe "Blockschaltbild" auf Seite 102).

Ein Script-Beispiel für die Verwendung der Interrupt Eingänge finden Sie im Kapitel t.b.d..

## 7.5 Typische Anwendung

### 7.5.1 Versorgung über $V_{AUX}$

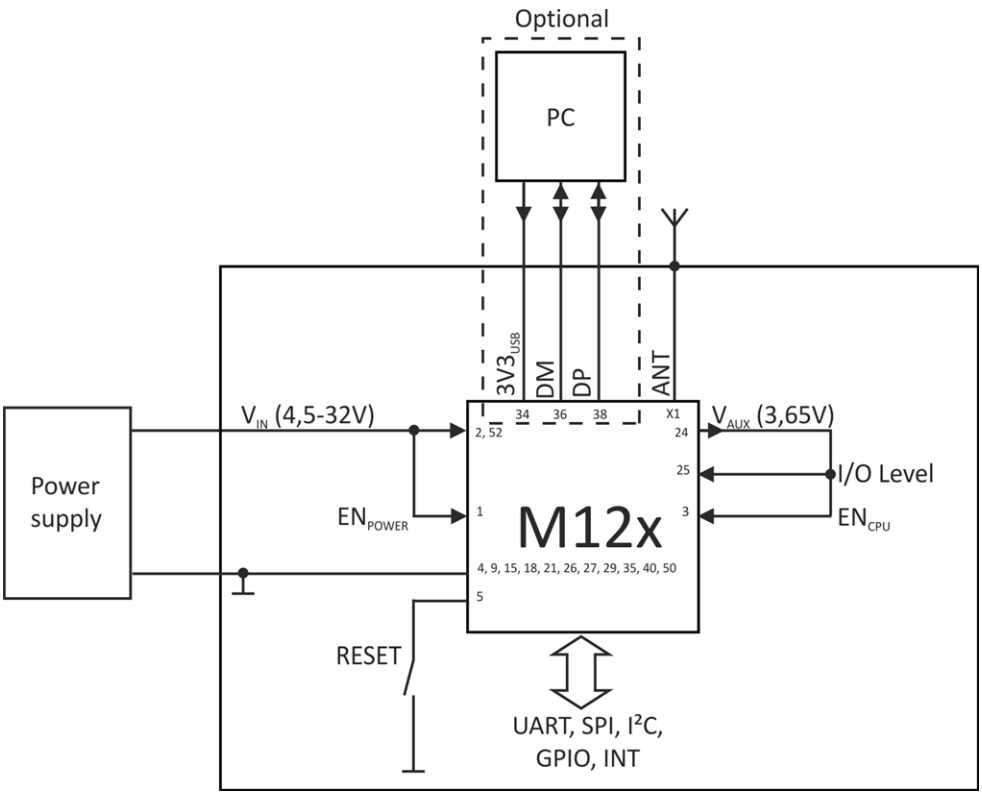



Versorgung über $V_{AUX}$

### Batterie

*Bei der ICR18650 handelt es sich um eine weit verbreitete Li-Ion Zelle, die für die Versorgung des M120 Moduls hervorragend geeignet ist. Bei einer direkten Anschaltung der Zelle an $V_{AUX}$ muss die Gesamtimpedanz beachtet werden (Batterie + Schutzschaltung + Leiterbahnen < 150mΩ ). Je nach Betriebstemperatur Anforderungen können zusätzliche Maßnahmen erforderlich sein (Voltage Delay bei Kälte). Beim Einsatz einer alternativen Zelle ist die Impulsstromfestigkeit zu beachten. Typischerweise kann jede Li-Ion Zelle mit ihrem "C-Rating" ohne Bedenken entladen (2Ah --› 2A) werden. Bei kleineren Zellen (200-500mAh) muss auf das C-Rating bzw. den maximalen Entladestrom geachtet werden.*

## 7.5.2 Versorgung über $V_{IN}$




Versorgung über $V_{IN}$

### Power supply

*Für die Versorgung können sowohl Batterien als auch Netzteile verwendet werden. Die Energiequelle sollte 6W liefern können. DC-Verteilnetze sollten nicht verwendet werden, da das rapidM2M Modul M120 über keine Schaltungen zum Schutz vor Überspannungen verfügt. Ein Verpolungsschutz ist hingegen vorhanden. Beim DC/DC-Wander zum Erzeugen der internen Versorgungsspannung handelt es sich um einen hoch effizienten Typ (94% Wirkungsgrad, 2,5µA Ruhestrom).*

### $EN_{Power}$

*Über dieses Signal wird der DC/DC-Wandler für die Versorgung des Controllers des M120 Moduls und die Erzeugung der 3,65 V Ausgangsspannung ($V_{AUX}$) aktiviert. Es gilt zu beachten, dass wenn das "$EN_{Power}$" Signal deaktiviert wird auch sämtliche anderen Signale, die ans rapidM2M Modul M120 angelegt sind, deaktiviert werden müssen.*

## 7.5.3 Gültig für beide Anwendungsfälle

### USB (opitonal)

*Über die USB-Slave-Schnittstelle wird die Verbindung zu einem PC hergestellt (siehe "USB-Schnittstelle" auf Seite 80).*

### Antenne

*Das rapidM2M Modul M120 ist mit einem GSM/GPRS Quad-Band Modem ausgestattet. Die Auswahl der verwendeten GSM-Antennen sollte entsprechend der am Einsatzort verfügbaren Mobilfunkfrequenzen getroffen werden. Die Optimierung der Antenne ist für energiesensible Anwendungen auf jeden Fall zu empfehlen.*

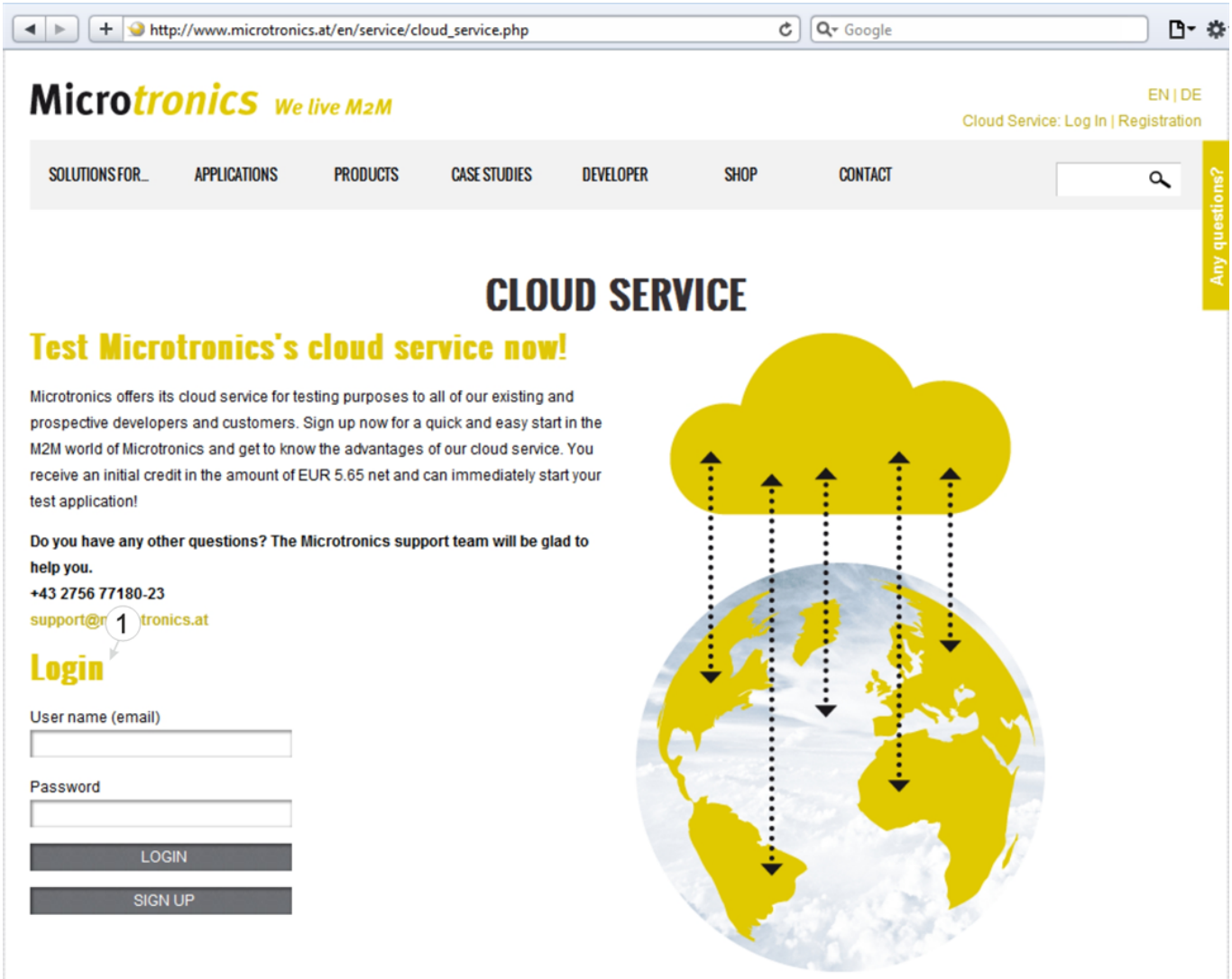
# Kapitel 8 Cloud-Server

***Hinweis:*** *Alle Screenshots zeigen den Cloud-Server in der Version 41.1. Bei neueren Versionen können geringfügige Änderungen am Erscheinungsbild des Servers vorgenommen worden sein.*

## 8.1 Neu registrieren und anmelden

Den Cloud-Server erreichen Sie über die Homepage der Firma Microtronics Engineering GmbH unter der Internetadresse:

cloud.microtronics.at




Übersicht der Homepage

| | |
|---|---|
| 1 Zugang zum Cloud-Server | |

### 8.1.1 Neuen Account anlegen

Sollten Sie noch nicht über einen Account am Cloud-Server verfügen, können Sie über den Button "SIGN UP" gratis einen Account anlegen. Dabei wird Ihnen auch ein Datenvolumen im Wert von 5,65 Euro (exkl. MwSt) gut geschrieben. Da unsere Geräte alle hinsichtlich der Reduzierung des Datenvolumens optimiert wurden, sollte das bereitgestellte Datenvolumen bei durchschnittlicher Nutzung für ca. ein Monat reichen. Wie Sie zusätzliches Datenvolumen erwerben, erfahren Sie im Kapitel t.b.d..

Nach dem Klicken auf den Button "SIGN UP" gelangen Sie zum Registrierungsformular.

Durch Klicken des Buttons "REGISTER" gelangen Sie zum nächsten Schritt. Wurden alle Pflichtfelder korrekt ausgefüllt, wird ein E-Mail mit einem Link zur Bestätigung der Anmeldung an die im Registrierungsformular angegebene E-Mailadresse versandt. Um die Registrierung abzuschließen, klicken Sie auf den im E-Mail enthaltenen Link. Anschließend können Sie sich über die Homepage der Firma Microtronics Engineering GmbH(siehe "Neu registrieren und anmelden" auf Seite 117) mit dem soeben erstellten Account am Cloud-Server anmelden. Als Benutzername wird dabei die im Registrierungsformular angegebene E-Mailadresse verwendet.

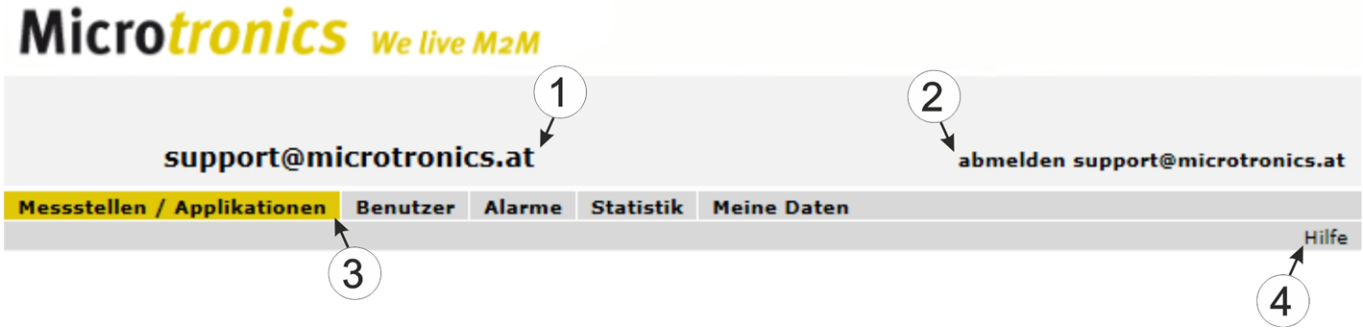
### 8.1.2 Anmelden

Geben Sie Ihren Benutzernamen (E-Mailadresse) und Ihr Kennwort in die entsprechenden Felder auf der Homepage der Firma Microtronics Engineering GmbH (siehe "Neu registrieren und anmelden" auf Seite 117) ein und klicken Sie auf "LOGIN". Dadurch gelangen Sie in den Bereich "Messstellen / Applikationen" des Cloud-Server.




Bereich "Messstellen / Applikationen" in dem alle zur Verfügung stehenden Messstellen angezeigt werden.

## 8.2 Übersicht




Übersicht Cloud-Server

| | |
|---|---|
| 1 Name des aktiven Benutzers | 3 Schaltflächen zum Wechseln zwischen den einzelnen Serverbereichen |
| 2 aktiven Benutzer ausloggen | 4 Aufruf des Hilfe-Menüs |

### 8.2.1 Erklärung der Symbole



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

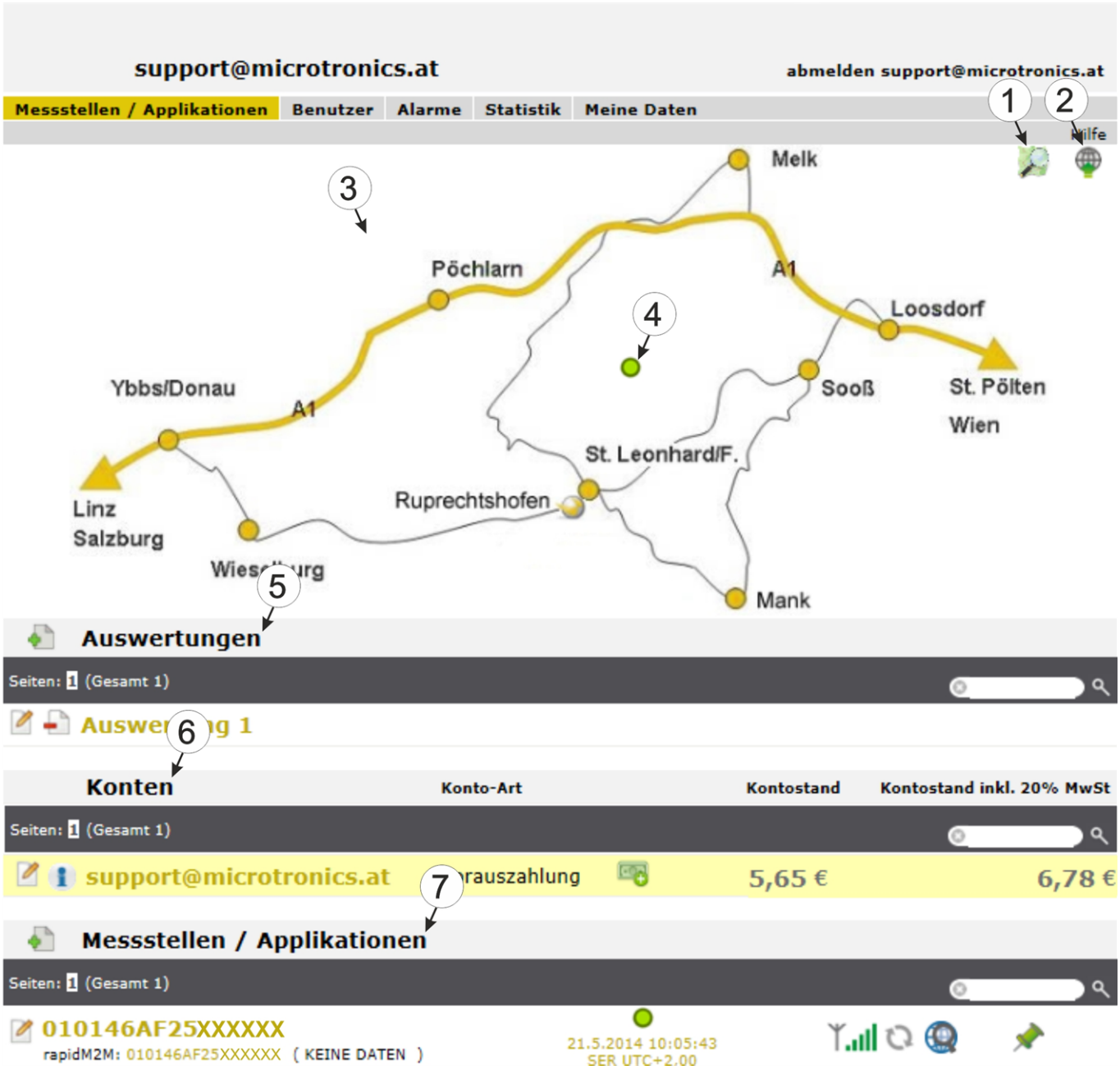


Löscht das nebenstehende Element (Auswertung, Messstelle, Benutzer, ...) aus der Liste



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

## 8.3 Bereich "Messstellen"

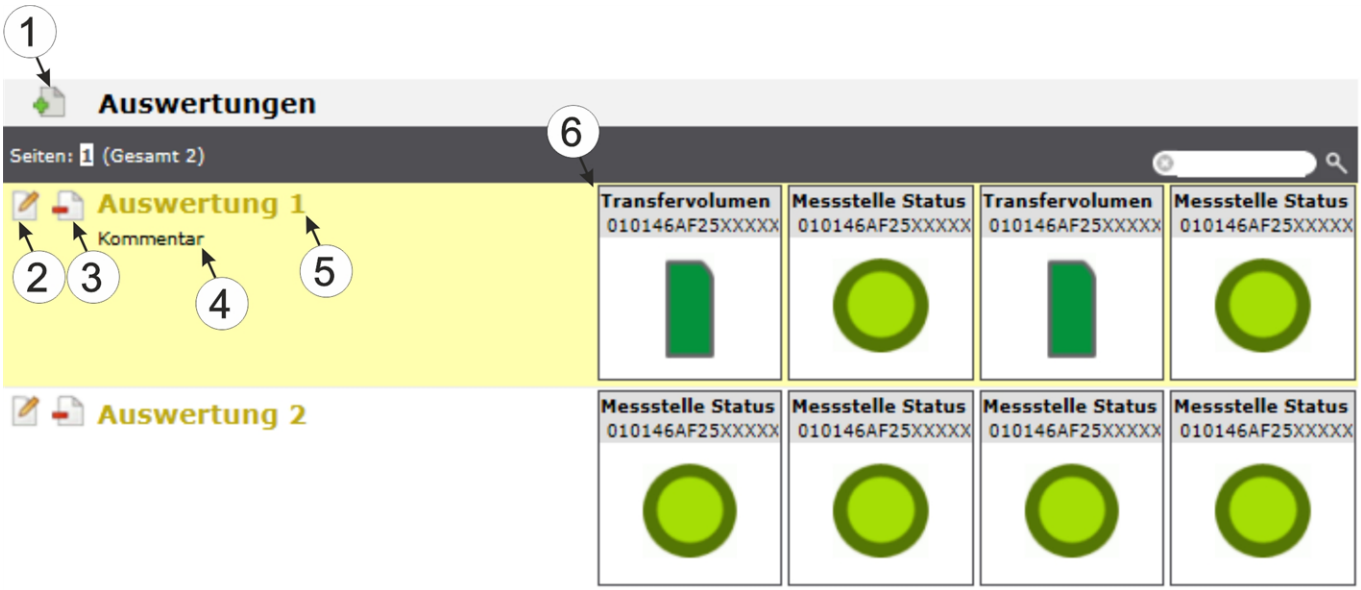


Übersicht des Bereichs "Messstellen"

| | |
|---|---|
| 1 | Symbol, über das sich eine OpenStreetMaps Karte laden lässt, auf der die Messstellen dargestellt werden. |
| 2 | Symbol, über das sich eine Bilddatei als "Karte" auf den Server laden lässt<br><br>Um die "Karte" wieder zu entfernen, öffnen Sie den Upload-Dialog erneut und klicken Sie auf "senden" ohne zuvor eine Biddatei auszuwählen. |
| 3 | aktuelle"Karte", auf der sich die Messstellen platzieren lassen |

| | |
|---|---|
| 4 | Symbol, das eine Messstelle auf der "Karte" repräsentiert |
| 5 | Liste der Auswertungen |
| 6 | Liste der Konten |
| 7 | Liste der Messstellen / Applikationen |

### 8.3.1 Auswertungen




Liste der Auswertungen

| | | | |
|---|---|---|---|
| 1 | fügt eine neue Auswertung hinzu(siehe "Grundkonfiguration" auf Seite 122) | 4 | Kommentar, der in der Grundkonfiguration der Auswertung eingegeben werden kann (siehe "Grundkonfiguration" auf Seite 122) |
| 2 | öffnet die Eingabemaske zur Grundkonfiguration der Auswertung | 5 | Durch Klicken auf den Namen der Auswertung wird die Eingabemaske zur Konfiguration und der Auswahl der Elemente geöffnet. |
| 3 | löscht die Auswertung | 6 | Anzeigeelemente für den letzten Messwert eines Messkanals (max. 4 pro Auswertung). Die Auswahl der Messstelle, des Kanals und des Typs des Anzeigeelements erfolgt in der Grundkonfiguration der Auswertung (siehe "Grundkonfiguration" auf Seite 122). |

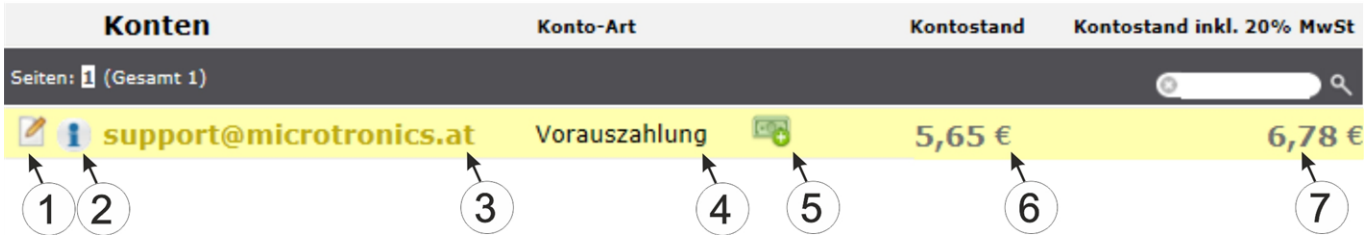
Ist eine der Auswertung gelb hinterlegt, wurde sie als Standardauswertung definiert. Dazu muss in der Grundkonfiguration der Auswertung der Typ auf "Standard" gesetzt werden (siehe t.b.d.). Das bedeutet, dass Sie direkt nach dem Einloggen über die Homepage der Firma Microtronics Engineering GmbH nicht zum Bereich "Messstellen" sondern zu dieser Auswertung gelangen. In diesem Fall gelangen Sie erst durch einen Klick auf "Menü" (siehe t.b.d.) in den Bereich "Messstellen".

***Hinweis:*** *Es ist zwar möglich bei mehreren Auswertungen den Typ auf "Standard" zu setzen. In diesem Fall ist jedoch nicht definiert, zu welcher der Auswertungen Sie nach dem Einloggen gelangen.*

#### 8.3.1.1 Grundkonfiguration

Diese Eingabemaske dient der grundlegenden Konfiguration der Auswertung. Eine detaillierte Anleitung hierzu finden Sie im Benutzerhandbuch für myDataweb-Server (206.886).

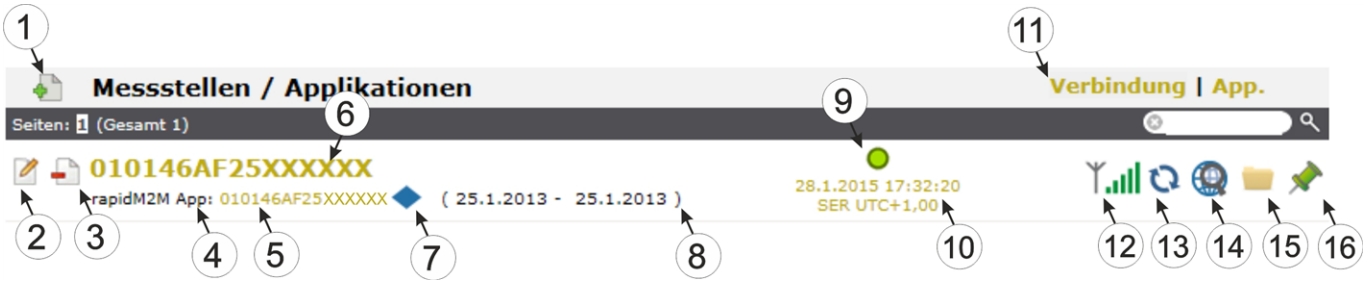
## 8.3.2 Konten




Liste der Konten

| | | | |
|---|---|---|---|
| 1 | öffnet die Eingabemaske zur Konfiguration der Alarmeinstellungen für das Guthaben. | 5 | öffnet das Eingabefenster zum Aufladen des Guthabens |
| 2 | zeigt Tarifinformationen als Tooltip an | 6 | aktueller Kontostand |
| 3 | öffnet die Anzeige der Buchungszeilen | 7 | aktueller Kontostand inkl. 20% Mehrwertsteuer |
| 4 | Art des Kontos | | |

## 8.3.3 Messstellen




Liste der Messstellen / Applikationen (verbindungsspezifische Steuerelemente sind eingeblendet)

| | |
|---|---|
| 1 | fügt eine neue Messstelle / Applikation hinzu |
| 2 | öffnet die Eingabemaske zur Konfiguration der Messstelle (siehe "Messstellenkonfiguration" auf Seite 193) |
| 3 | löscht die Messstelle / Applikation. |
| 4 | Typ der Messstelle |
| 5 | Seriennummer des mit der Messstelle verknüpften Geräts. Durch Klicken auf die Seriennummer wird die Eingabemaske zur Konfiguration des Geräts geöffnet (siehe "Gerätekonfiguration" auf Seite 195). |
| 6 | Name der Messstelle / Applikation. Durch Klicken auf den Namen wird die Eingabemaske zur Konfiguration der Messstelle / Applikation geöffnet (siehe "Messstellenkonfiguration" auf Seite 193). |
| 7 | Die Raute symbolisiert, dass ein Firmware-Update für das Gerät zur Verfügung steht. Die Farbe gibt dabei den Firmwaretyp an(blau für "Released", gelb für "Release Candidate" oder braun für "Beta Release"). |
| 8 | Anzeige des zur Verfügung stehenden Datenzeitraums |
| 9 | Status des Messgeräts. Neben dem Statussymbol werden auch, abhängig vom aktuellen Betriebszustand, weitere Symbole eingeblendet. |

Statussymbole

| | | | |
|---|---|---|---|
| | Normalzustand | | kein Messgerät zugeordnet (nur bei Messstellen) |
| | Alarmzustand | | Messgerät meldet sich nicht regelmäßig |
| | Warnungszustand | | |

weitere Symbole

| | | | |
|---|---|---|---|
| | nicht bestätigter Alarm in Liste | | Messgerät ist mit dem Server verbunden und überträgt Daten |
| update | Update läuft... | | |

| | |
|---|---|
| 10 | Zeitpunkt der letzten Verbindung. Durch einen Klick wird die Verbindungsliste geöffnet. |
| 11 | Schaltflächen zum Wechsel zwischen der Anzeige von verbindungsspezifischen und applikationsspezifischen Steuerelemente |
| 12 | Die Verbindung erfolgte via GSM/GPRS. Das Symbol gibt die Signalstärke an. Ein Klick auf das Symbol zeigt Informationen zur Funkzelle, die bei der letzten Verbindung verwendet wurde sowie zu den bei der letzten Positionsbestimmung berücksichtigten Funkzellen an. |
| 13 | Wurde beim Gerät die Verbindungsart „Intervall & Wakeup“ aktiviert, kann durch Klicken auf dieses Symbol eine Wakeup-SMS an das Gerät versendet werden. Beim Empfang einer solchen SMS baut das Gerät sofort eine Verbindung zum Server auf.<br><br>***Wichtiger Hinweis:*** *Durch den Versand einer Wakeup-SMS entstehen Kosten.* |
| 14 | zeigt die Position der Messstelle auf der OpenStreetMap Karte an (siehe "Messstelle Position" auf Seite 126). Ist dieses Symbol grün, wurden der Messstelle bereits manuell GPS-Koordinaten zugewiesen. Solange der Messstelle keine GPS Koordinaten zugewiesen wurden, ist dieses Symbol blau. |
| 15 | öffnet die Eingabemaske zur Verwaltung der Dateien, die mit dem Gerät synchronisiert werden (siehe "Dateiliste" auf Seite 127). |
| 16 | Öffnet die Eingabemaske zur Positionierung der Messstelle auf der als "Karte" verwendeten Bilddatei (siehe "Messstelle auf Karte positionieren" auf Seite 129). Dieses Symbol ist nur sichtbar, wenn eine entsprechende Bilddatei auf den Server geladen wurde |




Liste der Messstellen / Applikationen (Die applikationsspezifischen Steuerelemente sind eingeblendet:)

| | | |
|---|---|---|
| 1 | Schaltflächen zum Wechsel zwischen der Anzeige von verbindungsspezifischen und applikationsspezifischen Steuerelemente | |
| 2 | Versionsnummer der Applikation<br><br>Wurde die Applikation im Vergleich zu der der Versionsnummer entsprechenden Applikations-Vorlage verändert, wird die Versionsnummer rot und mit dem Zusatz "- verändert" angezeigt. | |
| 3 | Die Raute symbolisiert, ein verfügbares Update für die Applikation. Durch einen Klick auf die Raute kann die neue Version übernommen werden. | |
| 4 | gibt den Status der Synchronisation zwischen Server und Gerät an | |
| | ⌛ | Es wurden noch nicht alle Einstellungen (Konfiguration, Script, Dateien, ...) mit dem Gerät synchronisiert. |
| | ✓ | Alle Einstellungen (Konfiguration, Script, Dateien, ...) wurden erfolgreich mit dem Gerät synchronisiert. |
| | ⊗ | Der Messstelle / Applikation ist kein Gerät zugeordnet. |

### 8.3.3.1 Anlegen einer neuen Messstelle

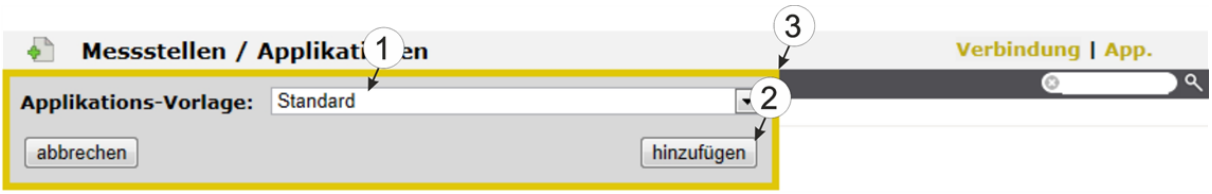



Anlegen der Messstelle

| | | | |
|---|---|---|---|
| 1 | Symbol "Neue Messstelle / Applikation hinzufügen" | 3 | Eingabefenster für das Anlegen einer neuen Messstelle |
| 2 | Feld zur Eingabe der Seriennummer | 4 | Button "hinzufügen" |

1. Klicken Sie auf das Symbol "Neue Messstelle hinzufügen". Daraufhin öffnet sich das Eingabefenster für das Anlegen einer neuen Messstelle.
2. Geben Sie die Seriennummer Ihres Geräts in das entsprechende Feld ein. Die Seriennummer finden Sie auf dem Typenschild des Geräts (siehe "Gerätekennzeichnung" auf Seite 106)
3. Klicken Sie auf den Button "hinzufügen".

4. Wählen Sie, falls gewünscht, eine Applikations-Vorlage aus der Liste. Dabei wird die neue Messstelle entsprechend der Vorlage konfiguriert (inkl. Script). Eine Beschreibung der zur Auswahl stehenden Applikations-Vorlagen finden Sie im Kapitel "Applikations-Vorlagen des Cloud-Servers" auf Seite 139. Bei der Auswahl "Standard" wird eine Messstelle mit Default-Konfiguration erstellt.




Auswahl der Applikations-Vorlage

| | |
|---|---|
| 1 Dropdown-Liste der verfügbaren Applikations-Vorlagen | 3 Eingabefenster zur Auswahl der Applikations-Vorlage |
| 2 Button "hinzufügen" | |

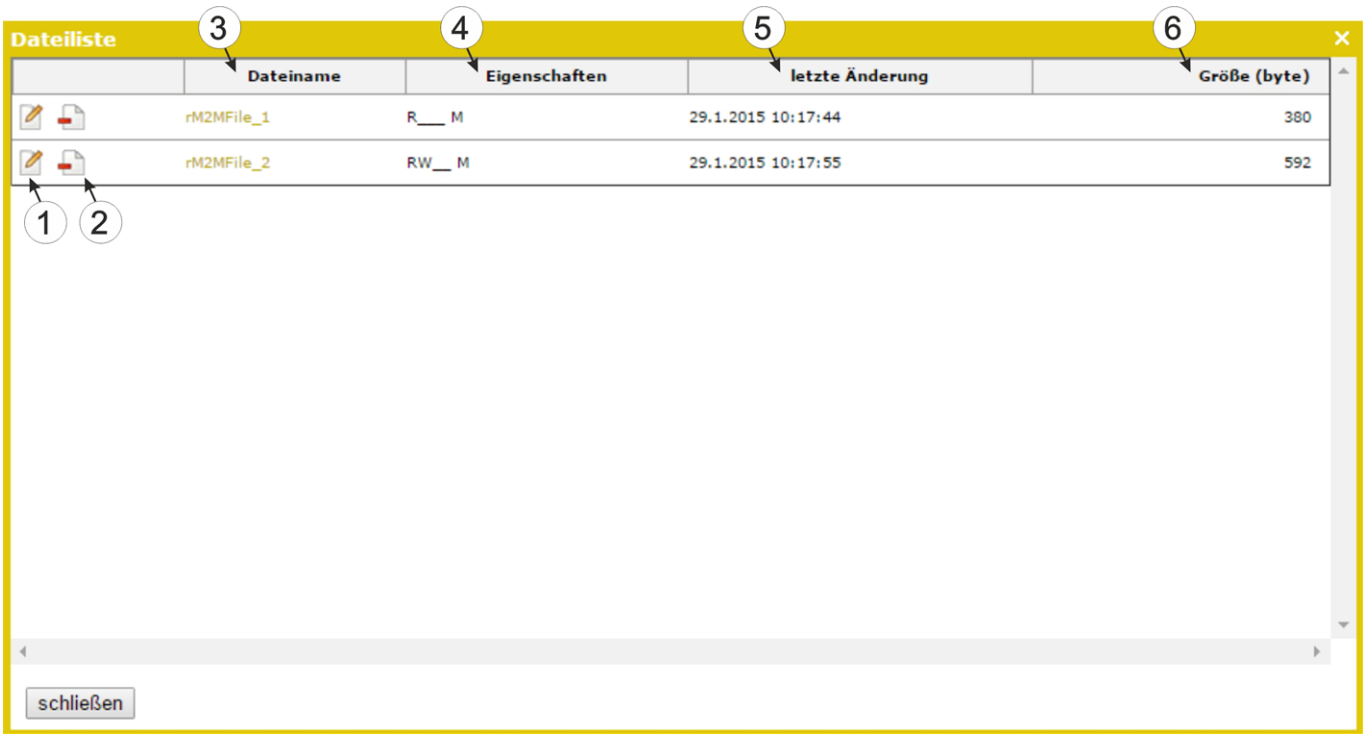
*Hinweis: Detaillierte Informationen zu den Applikations-Vorlagen finden Sie im Benutzerhandbuch für myDataweb-Server (206.886).*

5. Klicken Sie auf den Button "hinzufügen", um den Vorgang abzuschließen.

### 8.3.3.2 Messstelle Position

In diesem Fenster wird die Position des Messgeräts angezeigt. Wie oft das Messgerät seine Positionsdaten liefern soll, kann in der Messstellenkonfiguration mit dem Positionsintervall eingestellt werden. Eine detaillierte Anleitung hierzu finden Sie im Benutzerhandbuch für myDataweb-Server (206.886).

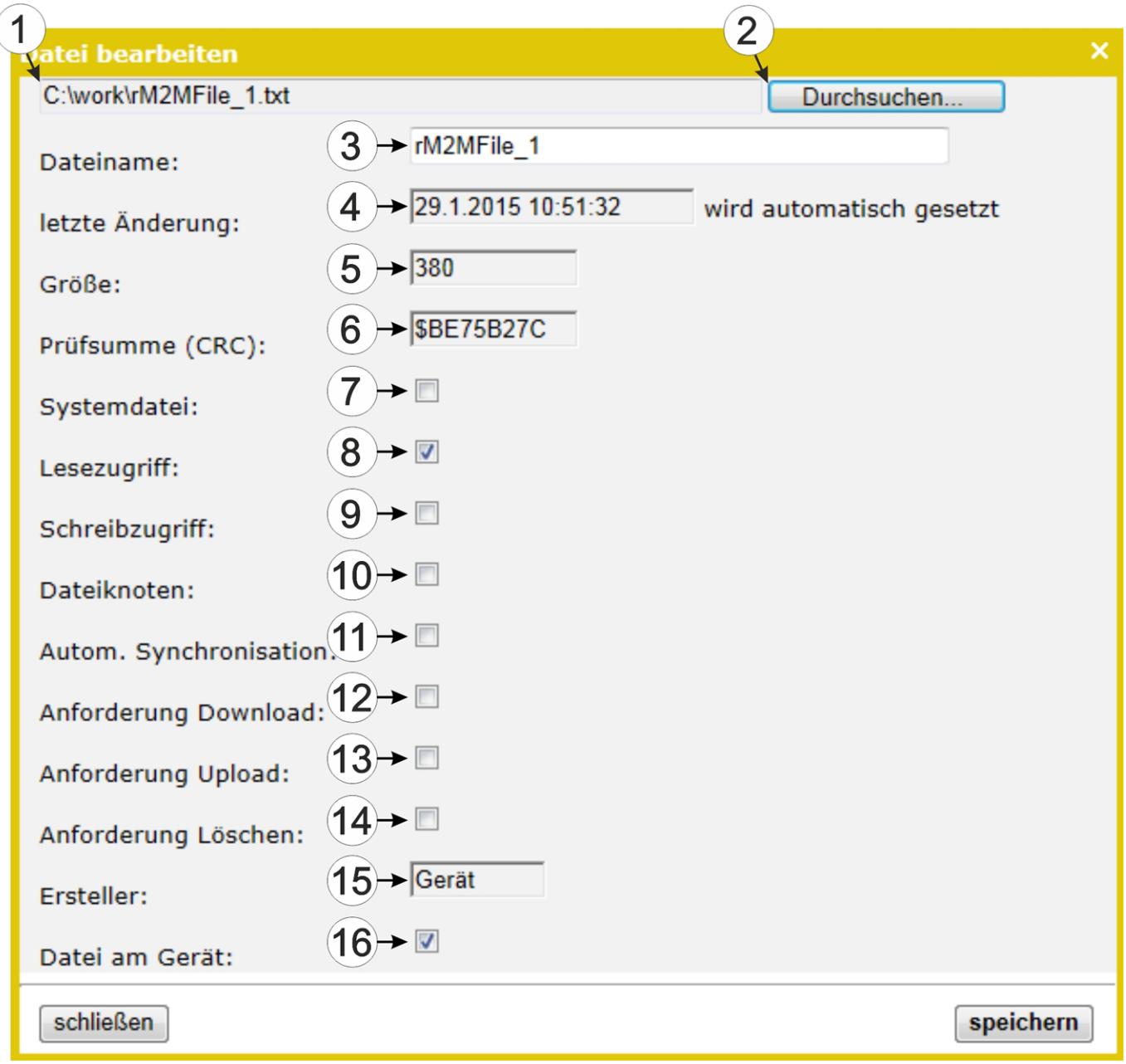
### 8.3.3.3 Dateiliste




Eingabemaske zur Verwaltung der Dateien

| | |
|---|---|
| 1 | öffnet die Eingabemaske zum Bearbeiten der Datei (siehe "Bearbeiten der Datei" auf Seite 128) |
| 2 | löscht die Datei vom Server. Dies hat keine Auswirkungen auf die Datei im Gerät. Abhängig von der Einstellung der Synchronisation (automatisch/manuell) wird die Datei gegebenenfalls erneut vom Gerät geladen. Soll die Datei vom Gerät gelöscht werden, muss die Checkbox "Anforderung Löschen" in der Eingabemaske zum Bearbeiten der Datei (siehe "Bearbeiten der Datei" auf Seite 128) gesetzt werden. |
| 3 | Name der Datei. Enthält die Datei bereits Daten (Größe ungleich 0), kann durch einen Klick auf den Namen die Datei vom Server geladen werden. |
| 4 | Eigenschaften der Datei. Diese können über die Eingabemaske zum Bearbeiten der Datei (siehe "Bearbeiten der Datei" auf Seite 128) bearbeitet werden.<br><br>R ... Read - Datei kann vom Gerät gelesen werden.<br>W ... Write - Datei kann ins Gerät geschrieben werden.<br>N ... Node - Dateiknoten (wird benötigt, um eine Datei ans Gerät senden zu können)<br><br>A/M ... automatische / manuelle Synchronisation |
| 5 | Datum / Zeit der letzten Änderung |
| 6 | Größe der Datei in Byte |

### 8.3.3.3.1 Bearbeiten der Datei




Eingabemaske zum Bearbeiten der Datei

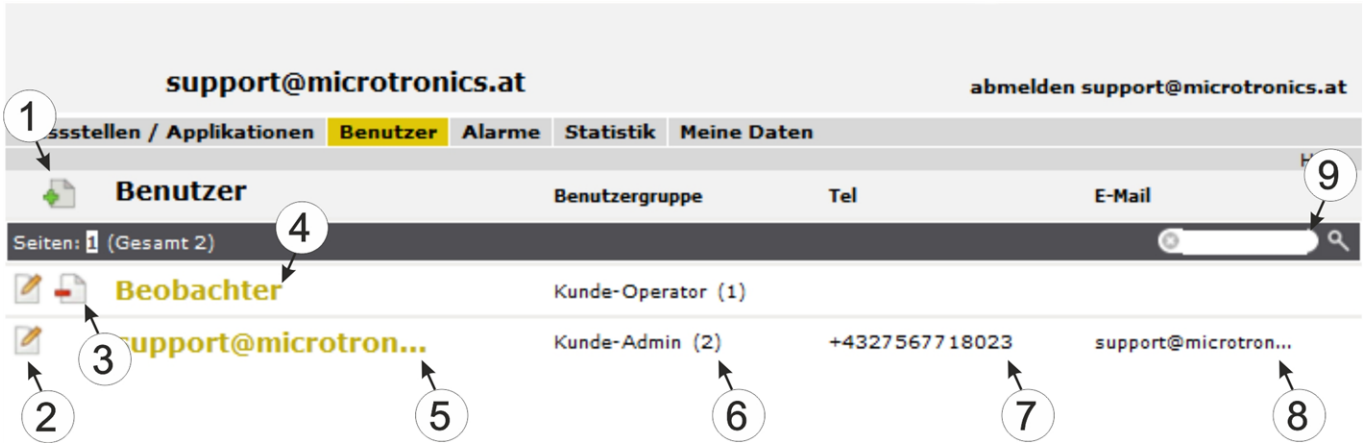
| | |
|---|---|
| 1 | Dateipfad der hochzuladenden Datei (ist nur sichtbar solange die Eingabemaske zum Bearbeiten der Datei geöffnet ist) |
| 2 | öffnet den Dialog zur Auswahl der hochzuladenden Datei |
| 3 | Dateiname, der für den File Transfer verwendet wird |
| 4 | Datum / Zeit der letzten Änderung |
| 5 | Größe der Datei in Byte |
| 6 | Prüfsumme (CRC), die vom System automatisch berechnet wird |

| | |
|---|---|
| 7 | zeigt an, ob e sich um eine Systemdatei handelt (Read only) |
| 8 | Auswahl, ob die Datei aus dem Gerät gelesen werden darf (selektiert: Lesezugriff gestattet) |
| 9 | Auswahl, ob die Datei ins Gerät geschrieben werden darf (selektiert: Schreibzugriff gestattet) |
| 10 | Auswahl, ob es sich um eine Datei oder einen Dateiknoten handelt (selektiert: Dateiknoten) |
| 11 | Auswahl, ob die Datei bei einer Änderung automatisch zwischen Server und Gerät synchronisiert werden soll (selektiert: automatische Synchronisation) |
| 12 | Die Datei wird bei der nächsten Verbindung ins Gerät geladen. Diese Option steht nur zur Verfügung, wenn die automatische Synchronisation nicht aktiv ist |
| 13 | Die Datei wird bei der nächsten Verbindung vom Gerät zum Server übertragen. Diese Option steht nur zur Verfügung, wenn die automatische Synchronisation nicht aktiv ist |
| 14 | Die Datei wird bei der nächsten Verbindung vom Gerät gelöscht |
| 15 | gibt an, ob die Datei durch das Gerät oder den Server erstellt wurde |
| 16 | reserviert für Erweiterungen |

#### 8.3.3.4 Messstelle auf Karte positionieren

Über diese Eingabemaske kann die Messstelle auf der als "Karte" verwendeten Bilddatei positioniert werden. Eine detaillierte Anleitung hierzu finden Sie im Benutzerhandbuch für myDataweb-Server (206.886).

## 8.4 Bereich "Benutzer"




Übersicht des Bereichs "Benutzer"

| | | | |
|---|---|---|---|
| 1 | hinzufügen eines neuen Benutzers (siehe "Benutzer anlegen" auf Seite 131) | 6 | gibt die dem Benutzer zugeordnete Benutzergruppe an |
| 2 | öffnet die Eingabemaske zur Bearbeitung der Benutzerdaten | 7 | Telefonnummer des Benutzers. Diese wird für Alarm-SMS und Anrufe verwendet. |
| 3 | löschen eines Benutzers<br><br>Der Hauptbenutzer des Cloud-Server Accounts kann nicht gelöscht werden. | 8 | E-Mail-Adresse des Benutzers. An diese Adresse werden die Alarm-E-Mails geschickt. |
| 4 | Benutzer mit niedrigeren Benutzerrechten, der durch den Hauptbenutzer des Cloud-Server Accounts erstellt wurde. Erkennbar an der Benutzergruppe "Kunde-Operator (1)".<br><br>Durch Klicken auf den Benutzernamen wird die Eingabemaske zur Bearbeitung der Benutzerdaten geöffnet. | 9 | Suchfeld zum Filtern der Benutzerliste. Es wird sowohl die Suche nach den Benutzernamen, Telefonnummern als auch nach E-Mail-Adressen unterstützt. |
| 5 | Hauptbenutzer des Cloud-Server Accounts, erkennbar an der Benutzergruppe "Kunde-Admin (2)"<br><br>Durch Klicken auf den Benutzernamen wird die Eingabemaske zur Bearbeitung der Benutzerdaten geöffnet. | | |

Der Hauptbenutzer des Cloud-Servers kann weitere Benutzer mit niedrigeren Benutzerrechten anlegen, die auf den Großteil der vom Cloud-Server bereitgestellten Funktionen nur Lesezugriff besitzen. Eine Aufstellung der Rechte der einzelnen Benutzergruppen finden Sie unter t.b.d..

### 8.4.1 Benutzer anlegen

Über diese Eingabemaske erfolgt die Eingabe der Benutzerdaten. Eine detaillierte Anleitung hierzu finden Sie im Benutzerhandbuch für myDataweb-Server (206.886).

## 8.5 Bereich "Alarme"




Übersicht des Bereichs "Alarme"

| | |
|---|---|
| 1 | Schaltflächen zum Wechseln zwischen den einzelnen Unterbereichen des Bereichs "Alarme" |

Der Bereich "Alarme" ist in folgende Unterbereiche aufgegliedert:

**Alarme**
*unquittierte Alarme/Warnungen*

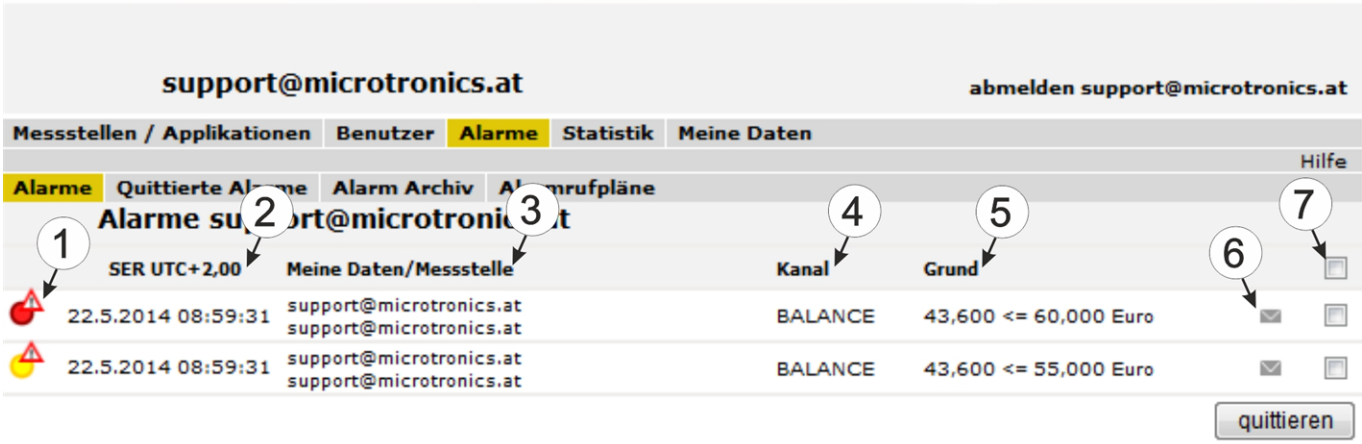
**Quittierte Alarme**
*Alarme/Warnungen, die bereits quittiert wurden, aber noch aktiv sind. Je nach Konfiguration erfolgt die Quittierung von Alarmen/Warnungen automatisch nach dem Versenden aller Benachrichtigungen oder muss durch den Benutzer manuell durchgeführt werden.*

**Alarm Archive**
*Alarme/Warnungen, die bereits quittiert wurden und nicht mehr aktiv sind. Es ist auch ersichtlich, von wem der/die Alarm/Warnung quittiert wurde/n.*

**Alarmrufpläne**
*dient zur Konfiguration der Quellen, des Zeitplans, der Empfänger und der Art (E-Mail, SMS, Telefonanruf) der Benachrichtigungen (siehe t.b.d.).*

> ***Wichtiger Hinweis:*** *Es gilt zu beachten, dass durch den Versand einer SMS oder einen Telefonanruf zur Alarmierung zusätzliche Kosten entstehen.*

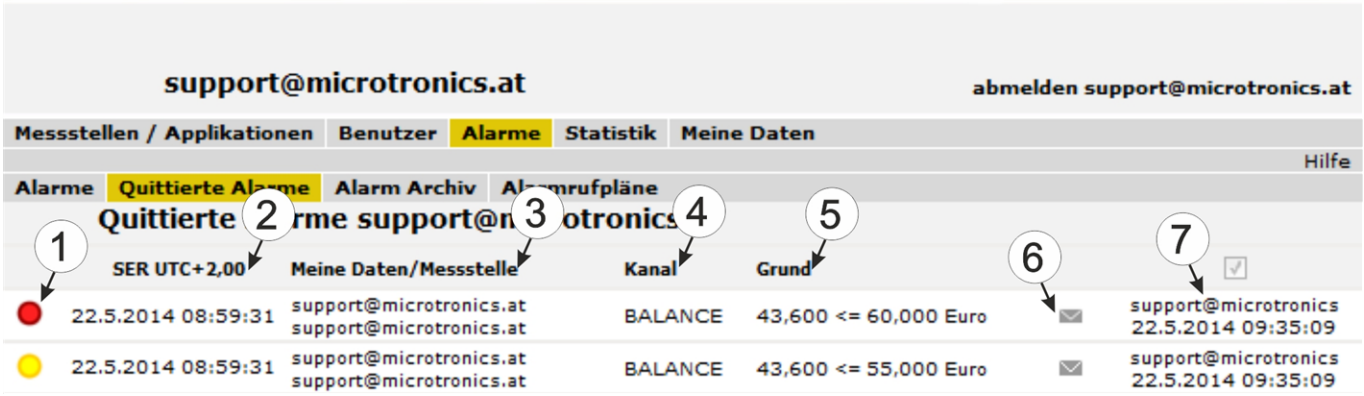
## 8.5.1 Unterbereich "Alarme"




Unterbereich "Alarme"

| | | | |
|---|---|---|---|
| 1 | Alarme werden durch ein rotes Statussymbol, Warnungen durch ein gelbes Statussymbol angezeigt. Ein überlagertes Dreieck mit Rufzeichen weist auf einen nicht bestätigten Alarm hin ⚠. | 5 | Grund für das Auftreten des Alarms / der Warnung ("auslösender Wert" >= oder <= "Schwelle" ) |
| 2 | Zeitpunkt, zu dem der Alarm / die Warnung aufgetreten ist (erste Zeile) bzw. zurückgenommen wurde (zweite Zeile) | 6 | Details zu den abgesandten Benachrichtigungen (siehe "Benachrichtigung" auf Seite 136) |
| 3 | Benutzernamen des Server Accounts (erste Zeile) und Messstelle bzw. Konto (zweite Zeile) bei der der Alarm aufgetreten ist | 7 | Checkbox zur Auswahl der Alarme/Warnungen die durch Klicken des Buttons "quittieren" quittiert werden |
| 4 | Kanal, der den Alarm / die Warnung ausgelöst hat | | |

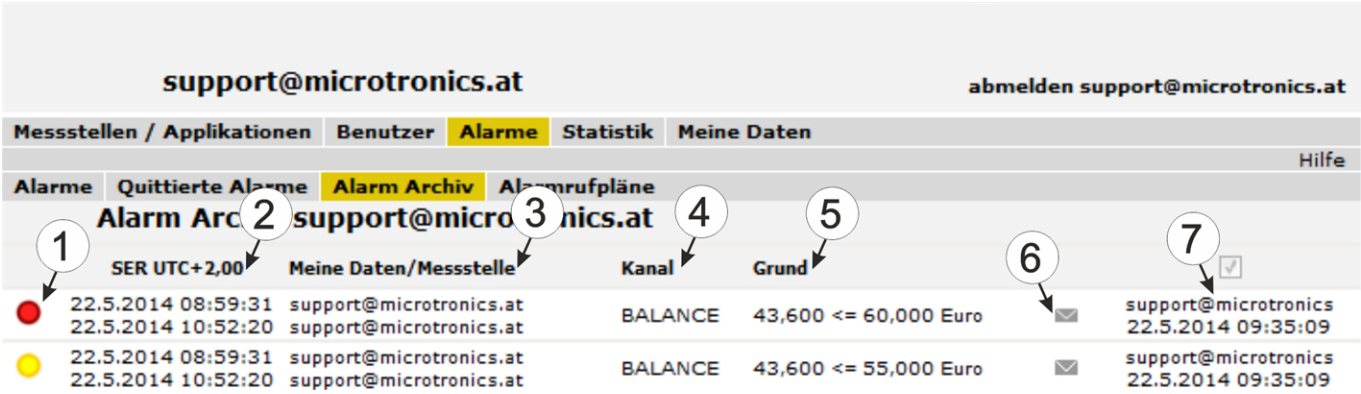
## 8.5.2 Unterbereich "Quittierte Alarme"




Unterbereich "Quittierte Alarme"

| | | | |
|---|---|---|---|
| 1 | Alarme werden durch ein rotes Statussymbol, Warnungen durch ein gelbes Statussymbol angezeigt. | 5 | Grund für das Auftreten des Alarms / der Warnung ("auslösender Wert" >= oder <= "Schwelle" ) |
| 2 | Zeitpunkt, zu dem der Alarm / die Warnung aufgetreten ist | 6 | Details zu den abgesandten Benachrichtigungen (siehe "Benachrichtigung" auf Seite 136) |
| 3 | Benutzernamen des Server Accounts (erste Zeile) und Messstelle bzw. Konto (zweite Zeile) bei der der Alarm aufgetreten ist | 7 | Name des Benutzers, durch den die Quittierung erfolgt ist (erste Zeile) und Zeitpunkt der Quittierung (zweite Zeile) (nur bei manueller Quittierung) |
| 4 | Kanal, der den Alarm / die Warnung ausgelöst hat | | |

## 8.5.3 Unterbereich "Alarm Archiv"



Unterbereich "Alarm Archiv"

| | | | |
|---|---|---|---|
| 1 | Alarme werden durch ein rotes Statussymbol, Warnungen durch ein gelbes Statussymbol angezeigt. | 5 | Grund für das Auftreten des Alarms / der Warnung ("auslösender Wert" >= oder <= "Schwelle" ) |
| 2 | Zeitpunkt, zu dem der Alarm / die Warnung aufgetreten ist (erste Zeile) bzw. zurückgenommen wurde (zweite Zeile) | 6 | Details zu den abgesandten Benachrichtigungen (siehe "Benachrichtigung" auf Seite 136) |
| 3 | Benutzernamen des Server Accounts (erste Zeile) und Messstelle bzw. Konto (zweite Zeile) bei der der Alarm aufgetreten ist | 7 | Name des Benutzers, durch den die Quittierung erfolgt ist (erste Zeile) und Zeitpunkt der Quittierung (zweite Zeile) (nur bei manueller Quittierung) |
| 4 | Kanal, der den Alarm / die Warnung ausgelöst hat | | |

## 8.5.4 Unterbereich "Alarmrufpläne"




Unterbereich "Alarmrufpläne"

| | | | |
|---|---|---|---|
| 1 | fügt einen neuen Alarmrufplan hinzu | 5 | Durch Klicken auf den Namen des Alarmrufplans wird die Eingabemaske zur Konfiguration des Alarmrufplans geöffnet (siehe"Alarmrufplan und Kommentar" auf Seite 135 ) |
| 2 | öffnet die Eingabemaske zur Konfiguration des Alarmrufplans (siehe "Alarmrufplan und Kommentar" auf Seite 135) | 6 | Kommentar, der über die Eingabemaske zur Konfiguration des Alarmrufplans (siehe "Alarmrufplan und Kommentar" auf Seite 135) eingegeben werden kann |
| 3 | löscht den Alarmrufplan | 7 | Statussymbol, das anzeigt, ob der Alarmrufplan aktiv ist (grün: aktiv, grau: inaktiv) |
| 4 | öffnet die Eingabemaske zur Konfiguration der Empfängerliste für den Alarmrufplan (siehe "Einrichten einer telefonischen Alarmierung" auf Seite 135) | | |

### 8.5.4.1 Alarmrufplan und Kommentar

Über diese Eingabemaske erfolgt die Konfiguration des Alarmrufplans und die Eingabe eines Kommentars zum Alarmrufplan. Eine detaillierte Anleitung hierzu finden Sie im Benutzerhandbuch für myDataweb-Server (206.886).

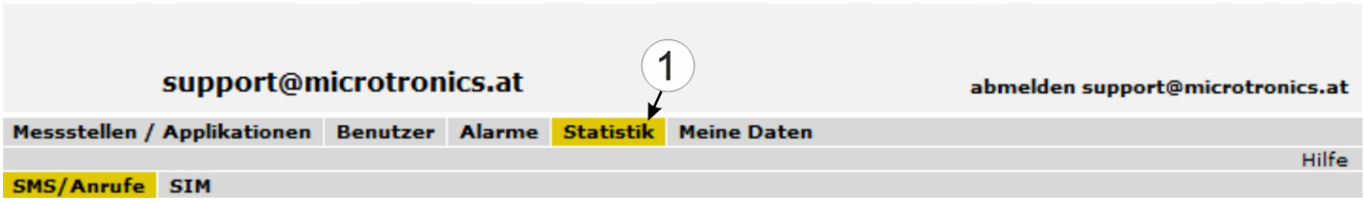
### 8.5.4.2 Einrichten einer telefonischen Alarmierung

Über diese Eingabemaske erfolgt die Konfiguration der telefonischen Alarmierung. Eine detaillierte Anleitung hierzu finden Sie im Benutzerhandbuch für myDataweb-Server (206.886).

### 8.5.5 Benachrichtigung

Diese Eingabemaske enthält Informationen zu den versandten Benachrichtigungen. Eine detaillierte Anleitung hierzu finden Sie im Benutzerhandbuch für myDataweb-Server (206.886).

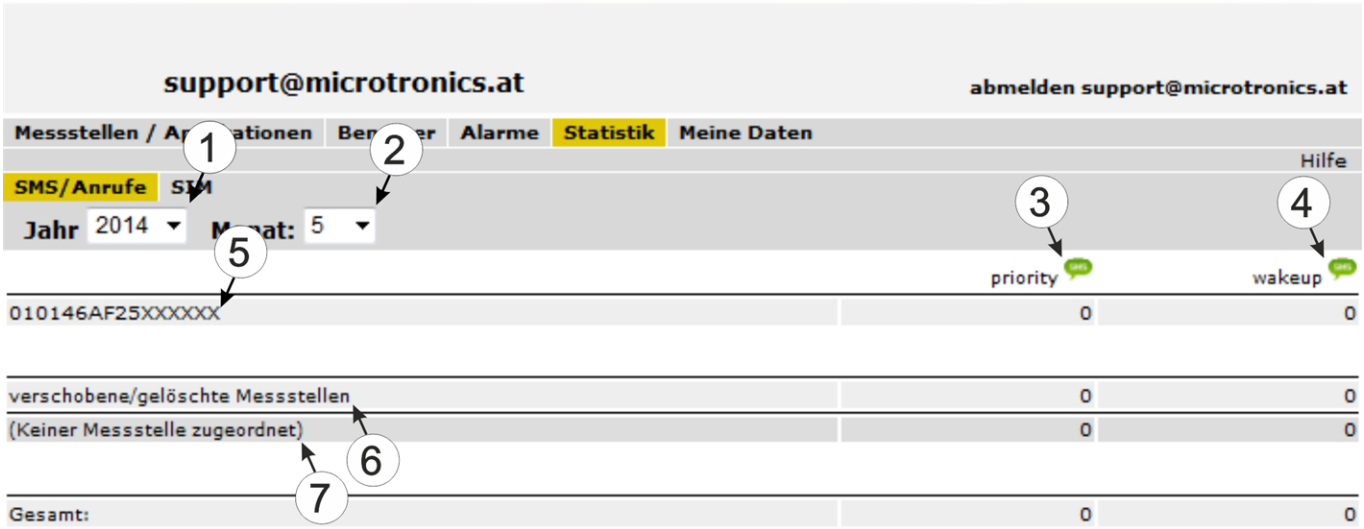
## 8.6 Bereich "Statistik"




Übersicht des Bereichs "Statistik"

| | |
|---|---|
| 1 | Schaltflächen zum Wechseln zwischen den einzelnen Unterbereichen des Bereichs "Statistik" |

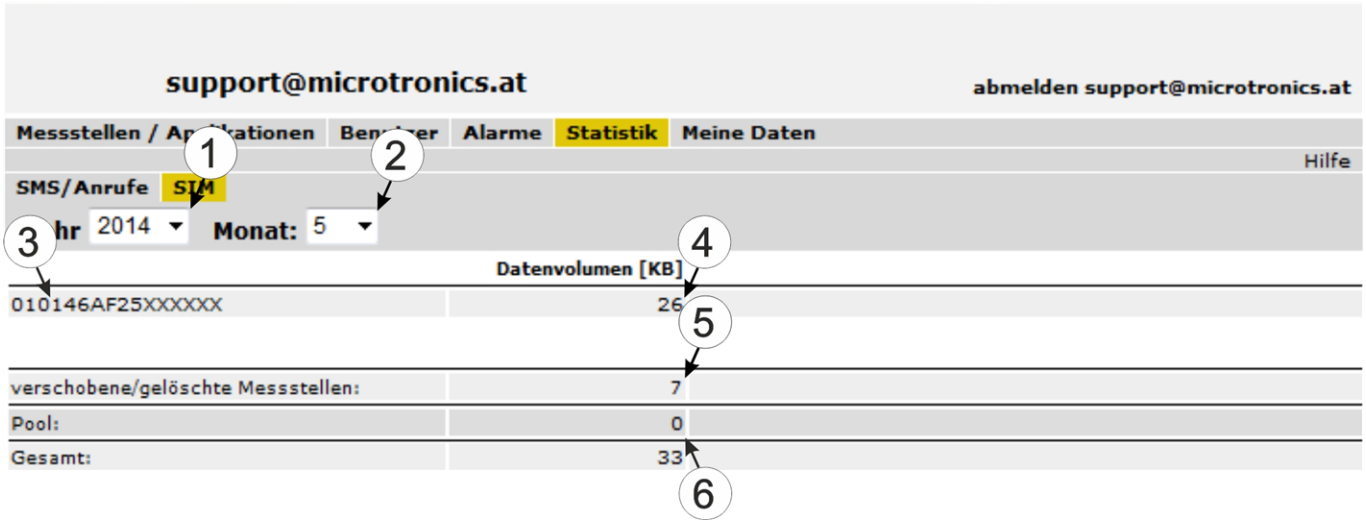
## 8.6.1 Unterbereich "SMS/Anrufe"




Unterbereich "SMS/Anrufe"

| | |
|---|---|
| 1 | Die Statisik wird immer für einen Monat erstellt. Über diese Dropdown-Liste wird zunächst das Jahr ausgewählt. Die Auswahl des Monats erfolgt mit Hilfe der Dropdown-Liste "Monat". |
| 2 | Auswahl des Monats, für das die Statistik erstellt werden soll |
| 3 | Anzahl der hochprioritären SMS, die versendet wurden. Die Zustellung dieser SMS erfolgt mit erhöhter Priorität. Eine erfolgreiche Zustellung wird zudem durch eine Rückmeldung bestätigt. Ob die SMS auch tatsächlich gelesen wurde, kann anhand dieser Rückmeldung jedoch nicht ermittelt werden. Die Anzahl der versendeten SMS werden für jede Messstellle aufgelistet. |
| 4 | Anzahl der versendeten Wakeup-SMS. Wenn sich ein Gerät im Wakeup-Modus befindet (Verbindungsart „Intervall & Wakeup“), kann es über den Server in den ALOHA-Übertragungsmodus versetzt werden. Hierfür wird eine SMS vom Server an das Gerät gesendet. Die Anzahl der versendeten SMS werden für jede Messstellle aufgelistet. |
| 5 | Auflistung der aktiven Messstellen |
| 6 | Auflistung der SMS, die durch verschobene oder gelöschte Messstellen verursacht wurden. Es handelt sich dabei um die Anzahl der SMS, die bis zum Verschieben/Löschen der Messstellen verursacht wurden. |
| 7 | Auflistung der SMS, die nicht unmittelbar einer Messstelle zugeordnet werden können, z.B. verursacht durch das Auslösen eines Testalarms |

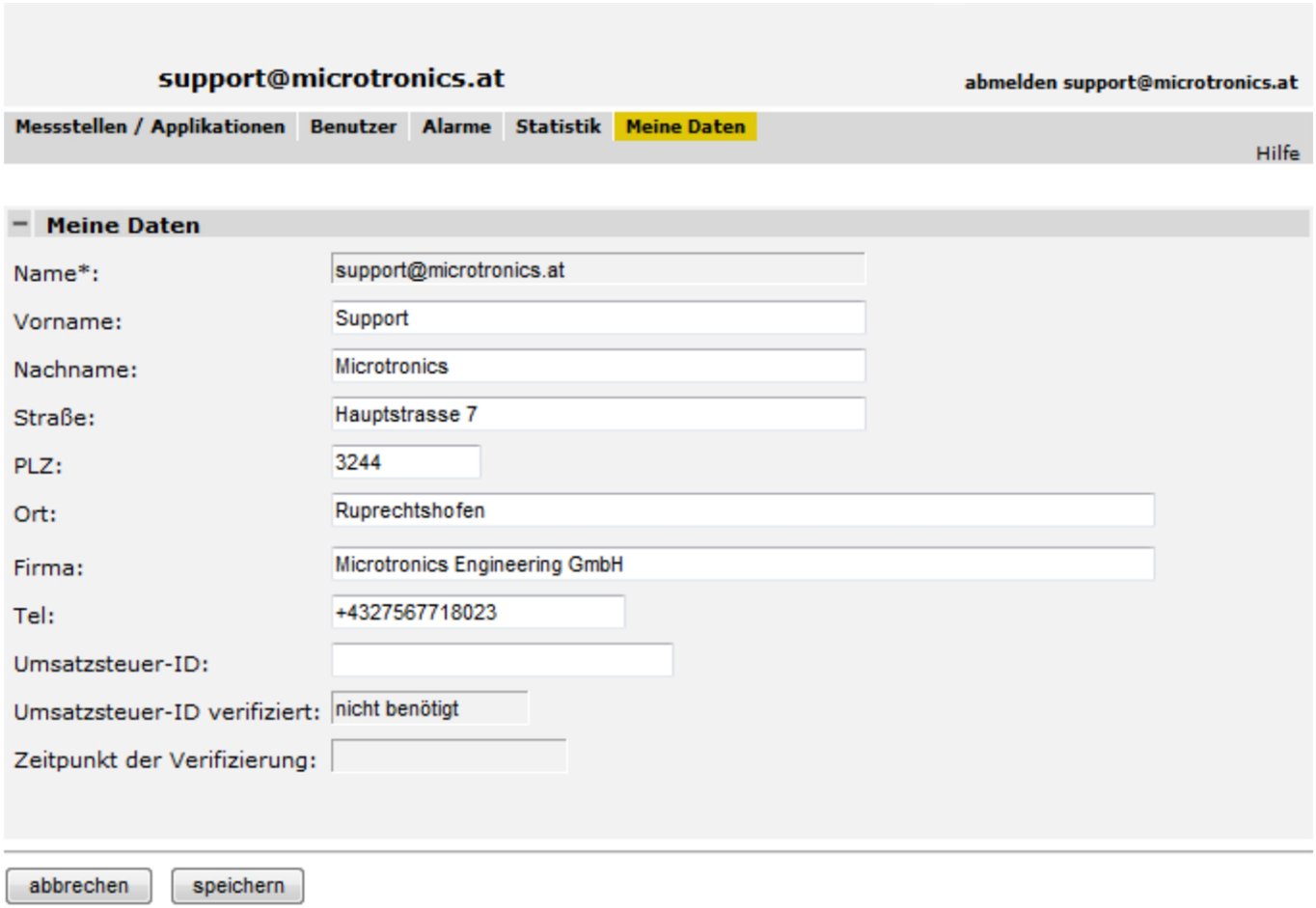
## 8.6.2 Unterbereich "SIM"




Unterbereich "SIM"

| | |
|---|---|
| 1 | Die Statisik wird immer für einen Monat erstellt. Über diese Dropdown-Liste wird zunächst das Jahr ausgewählt. Die Auswahl des Monats erfolgt mit Hilfe der Dropdown-Liste "Monat". |
| 2 | Auswahl des Monats, für das die Statistik erstellt werden soll |
| 3 | Auflistung der aktiven Messstellen |
| 4 | verbrauchtes Datenvolumen, aufgelistet nach den einzelnen Messstellen |
| 5 | Datenvolumen, das von verschobenen oder gelöschten Messstellen verbraucht wurde. Es handelt sich dabei um das Datenvolumen, das bis zum Verschieben/Löschen der Messstellen verbraucht wurde. |
| 6 | Datenvolumen, das von Geräten verbraucht wurde, die sich im Pool befinden. Es handelt sich dabei um das Datenvolumen, das ab dem Zeitpunkt an dem das Gerät in den Pool verschoben wurde, verbraucht wurde. |

## 8.7 Bereich "Meine Daten"




Übersicht des Bereichs "Meine Daten"

In diesem Bereich können mit Ausnahme des Benutzernamens des Cloud-Server Accounts alle bei der Registrierung eingegebenen Daten editiert werden.

## 8.8 Applikations-Vorlagen des Cloud-Servers

t.b.d.

6872
6872
6872

# Kapitel 9 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.*

## 9.1 Allgemein

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

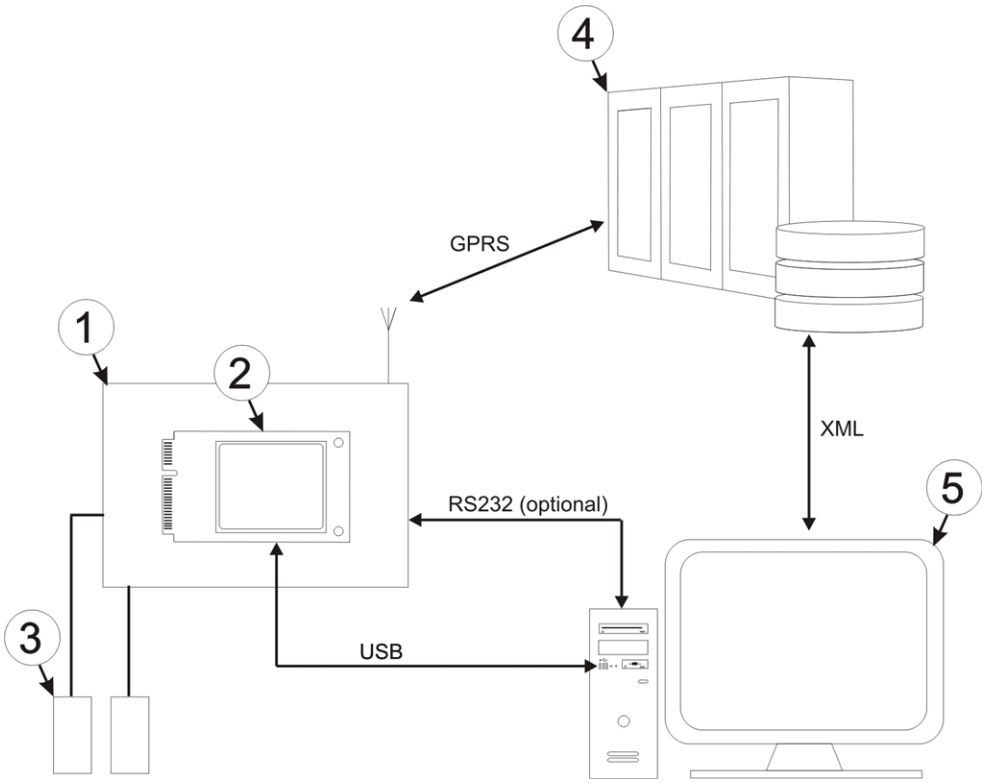
www.microtronics.at/Toolset

Es handelt sich um eine IDE, die dem Kunden bei der Erstellung von Applikationen für M120 Module unterstützen soll. Diese umfasst den kompletten Entwicklungsprozess - angefangen vom Editieren des PAWN-Scripts (siehe "Pawn Script" auf Seite 197), über das Kompilieren des Scripts in Byte-Code und Laden des Byte-Codes in das M120 Modul bis hin zum Auslesen der vom M120 Modul an den Cloud-Server übertragenen Daten über die XML-Schnittstelle des Cloud-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 M120 Modul. Die 10 Konfigurationsspeicherblöcke können ebenfalls mittels rapidM2M Toolset aus dem M120 Modul gelesen bzw. in das M120 Modul geschrieben werden. Für Testzwecke bietet das rapidM2M Toolset die Möglichkeit einen einfachen seriellen Sensor zu simulieren und enthält eine Sammlung von Beispiel-Scripts. Schlussendlich kann das rapidM2M Toolset auch dazu verwendet werden, Firmwareupdates in das M120 Modul einzuspielen.

## 9.2 Voraussetzungen

| | |
|---|---|
| Schnittstellen | 1 x USB<br>1 x RS232 (optional) |
| Betriebssystem | Win XP<br>Windows Vista<br>Windows 7 |
| Internetverbindung | empfohlen |
| Benötigter Speicherplatz | ca. 75MB |

# 9.3 Funktionsprinzip




Funktionsprinzip

| | | | |
|---|---|---|---|
| 1 | Gerät (kundenspezifische Hardware) mit eingesetztem M120 Modul. z.B.rapidM2M Developer Board | 4 | Cloud-Sever |
| 2 | M120 Modul | 5 | PC, auf dem das rapidM2M Toolset installiert ist |
| 3 | Sensor, der mit dem Gerät (kundenspezifische Hardware) verbunden ist (optional) | | |

Das rapidM2M Toolset kommuniziert über die USB-Schnittstelle direkt mit dem M120 Modul. Dazu gehören:

- der Script-Download (siehe "Kateireiter "Script"" auf Seite 148)
- der Up/Download der Konfigurationsspeicherblöcke (siehe "Karteireiter "Config"" auf Seite 150)
- die Anzeige der Konsolenausgaben (siehe "Karteireiter "Console"" auf Seite 151)
- das Auslesen der Log-Einträge (siehe "Karteireiter "Log"" auf Seite 154)
- das Firmwareupdate (siehe "Karteireiter "FW"" auf Seite 155)

Die RS232-Verbindung ist optional und wird nur benötigt, wenn die Möglichkeit zum Simulieren eines einfachen seriellen Sensors (siehe "Karteireiter "COM Data"" auf Seite 151) des rapidM2M Toolset genutzt werden soll.

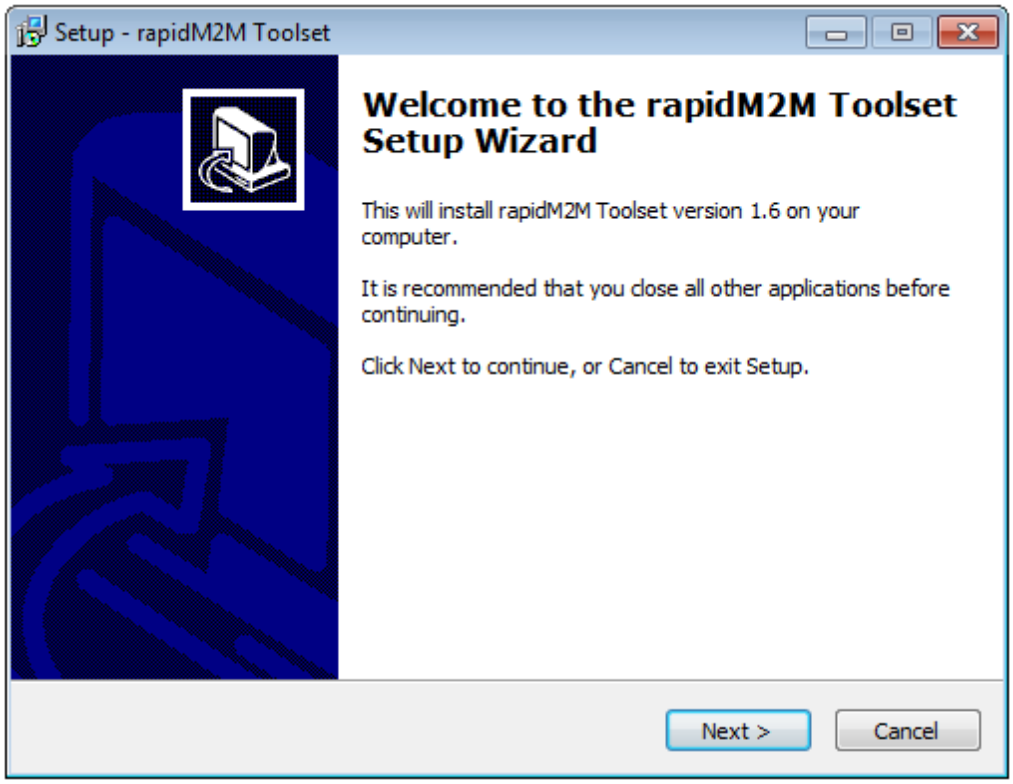
Das rapidM2M Toolset  bietet auch die Möglichkeit die Daten über die XML-Schnittstelle des Cloud-Severs auszulesen (siehe "Karteireiter "Server"" auf Seite 153). Dadurch kann sehr einfach und schnell überprüft werden, ob die komplette Prozesskette - angefangen beim Speichern der Daten mittels Script über die Übertragung per GPRS bis hin zur Ablage der Daten am Server - korrekt funktioniert.

## 9.4 Installation

Das folgende Kapitel beschreibt den Installationsprozess unter Windows 7.

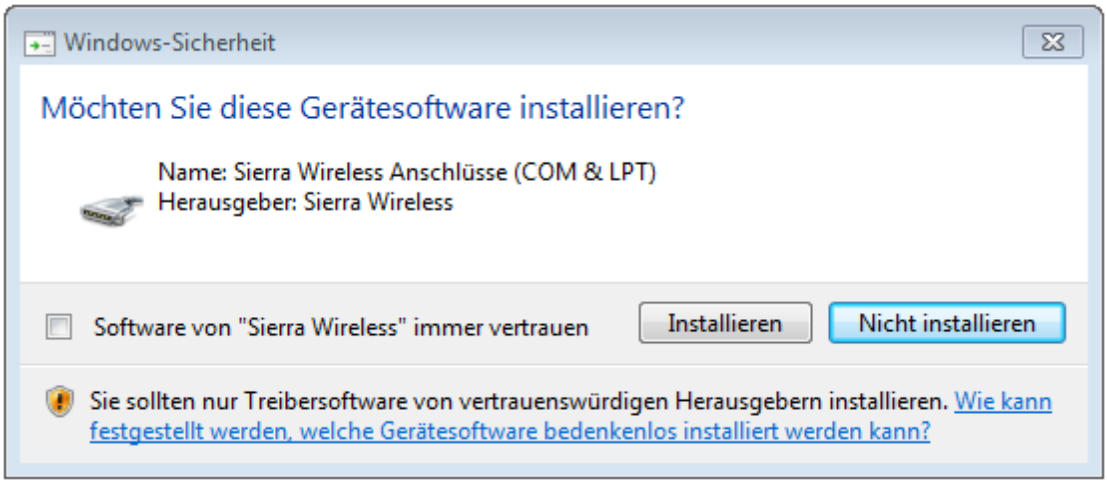
1. Führen Sie die Datei *"InstRapidM2MToolset.exe"* aus, um den Installationsprozess zu starten.

   ***Hinweis:*** *Verbinden Sie das rapidM2M Developer Paket M12x erst nach Abschluss des Installationsprozesses mit Ihrem PC da die benötigten USB-Treiber erst während dieses Vorgangs installiert werden.*

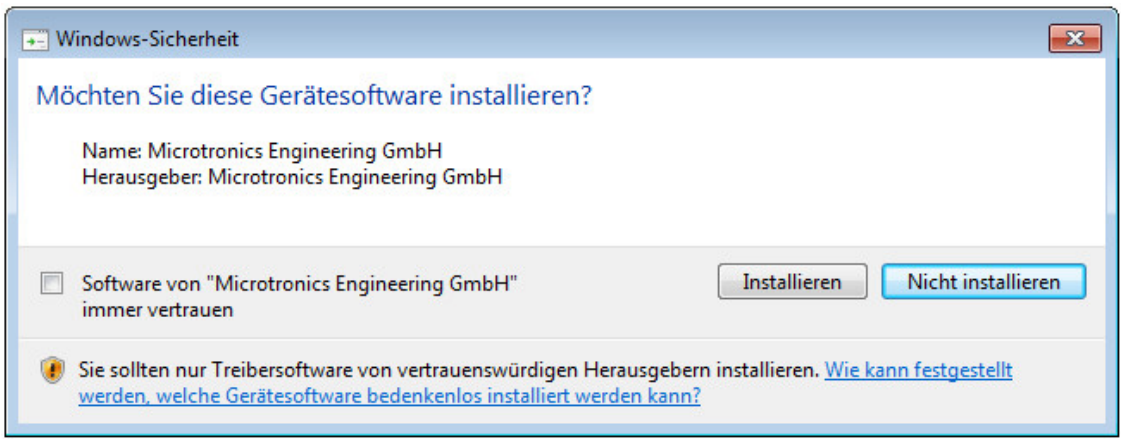



rapidM2M Toolset Setup Wizard

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.

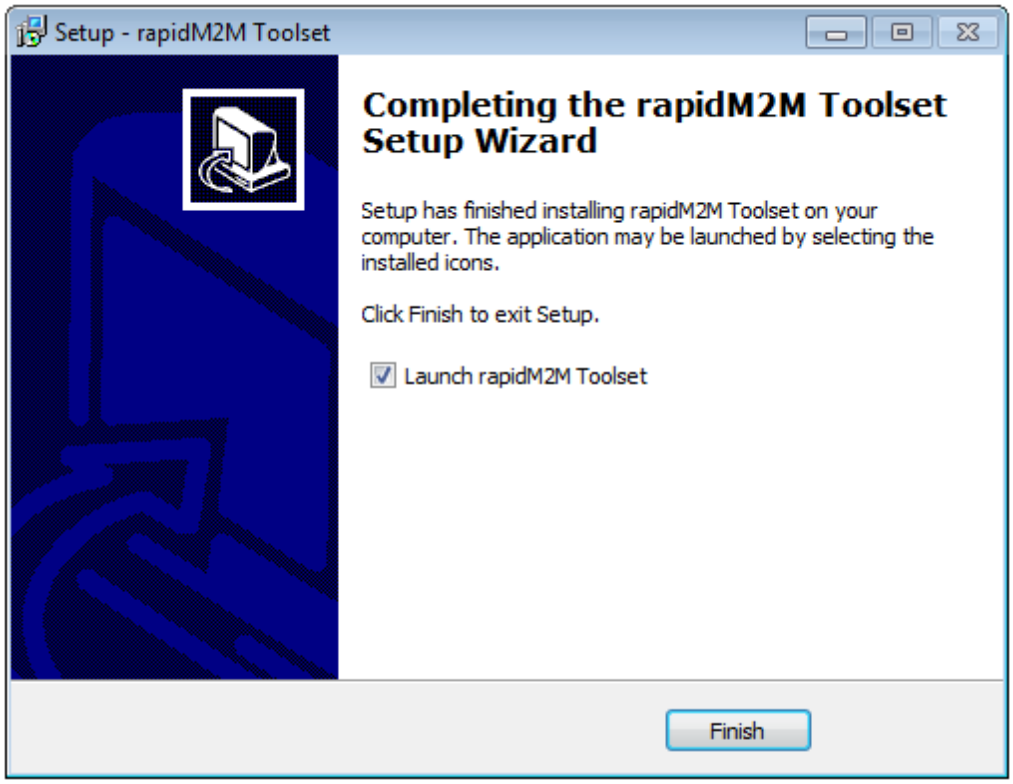



Installation der USB-Treiber für M120 Module




Installation der USB-Treiber für M22x Module

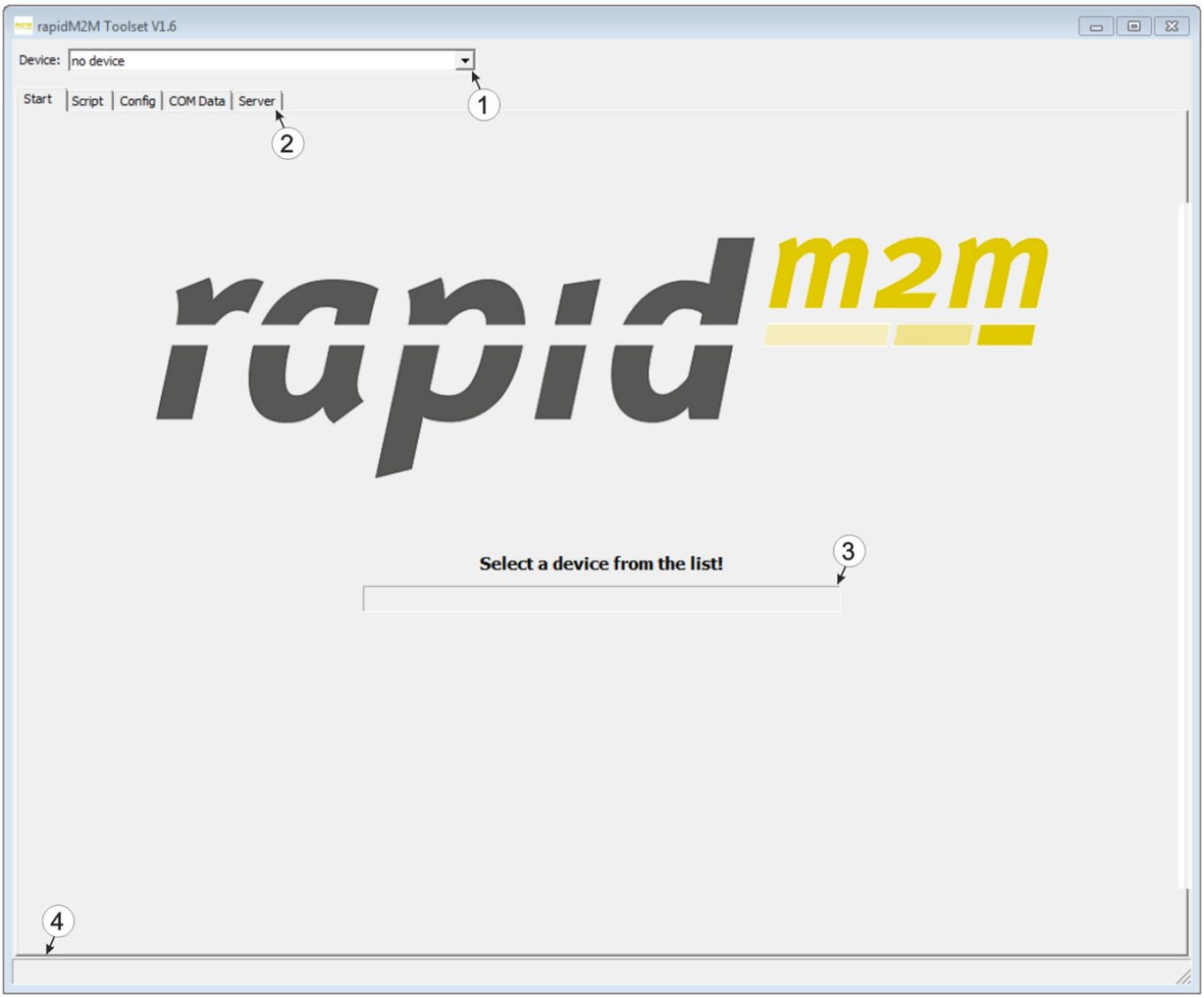
3. Wenn Sie schließlich zur folgenden Ansicht gelangen, schließen Sie den Installationsvorgang durch Klicken auf den Button *"Finish"* ab.




Setup abschließen

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

# 9.5 Übersicht




Karteireiter "Start" (kein M120 Modul ausgewählt)

| | | | |
|---|---|---|---|
| 1 | Auswahl des M120 Moduls | 3 | Fortschrittsbalken für den Aufbau der USB-Verbindung zum M120 Modul |
| 2 | Auswahl der Karteireiter für die verschiedenen Funktionen des rapidM2M Toolset , die auch ohne verbundenem Modul verfügbar sind | 4 | Anzeige des Kommunikationsstatus auf der USB-Schnittstelle zum M120 Modul |

Sobald die USB-Verbindung zu einem M120 Modul hergestellt wurde, werden die Karteireiter der Funktionen, die eine bestehende USB-Verbindung zum Modul erfordern, eingeblendet und der Karteireiter "Start" ausgeblendet. Zusätzlich werden im rechten oberen Bereich des rapidM2M Toolset weitere Informationen zum Modul eingeblendet.

***Wichtiger Hinweis:*** *Verknüpfen Sie das M120 Modul mit einer Messstelle (siehe "Anlegen einer neuen Messstelle" auf Seite 125) 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 M120 Modul verknüpft ist.*




Verbindung zum M120 Modul hergestellt

| | |
|---|---|
| 1 Auswahl des M120 Moduls | 4 Anzeige der GSM-Signalstärke (siehe "GSM-Signalstärke") |
| 2 Auswahl der Karteireiter für die verschiedenen Funktionen des rapidM2M Toolset | 5 Seriennummer des verbundenen Moduls |
| 3 Anzeige des GSM-Verbindungsstatus (siehe "GSM-Verbindungsstatus") | |

### GSM-Verbindungsstatus

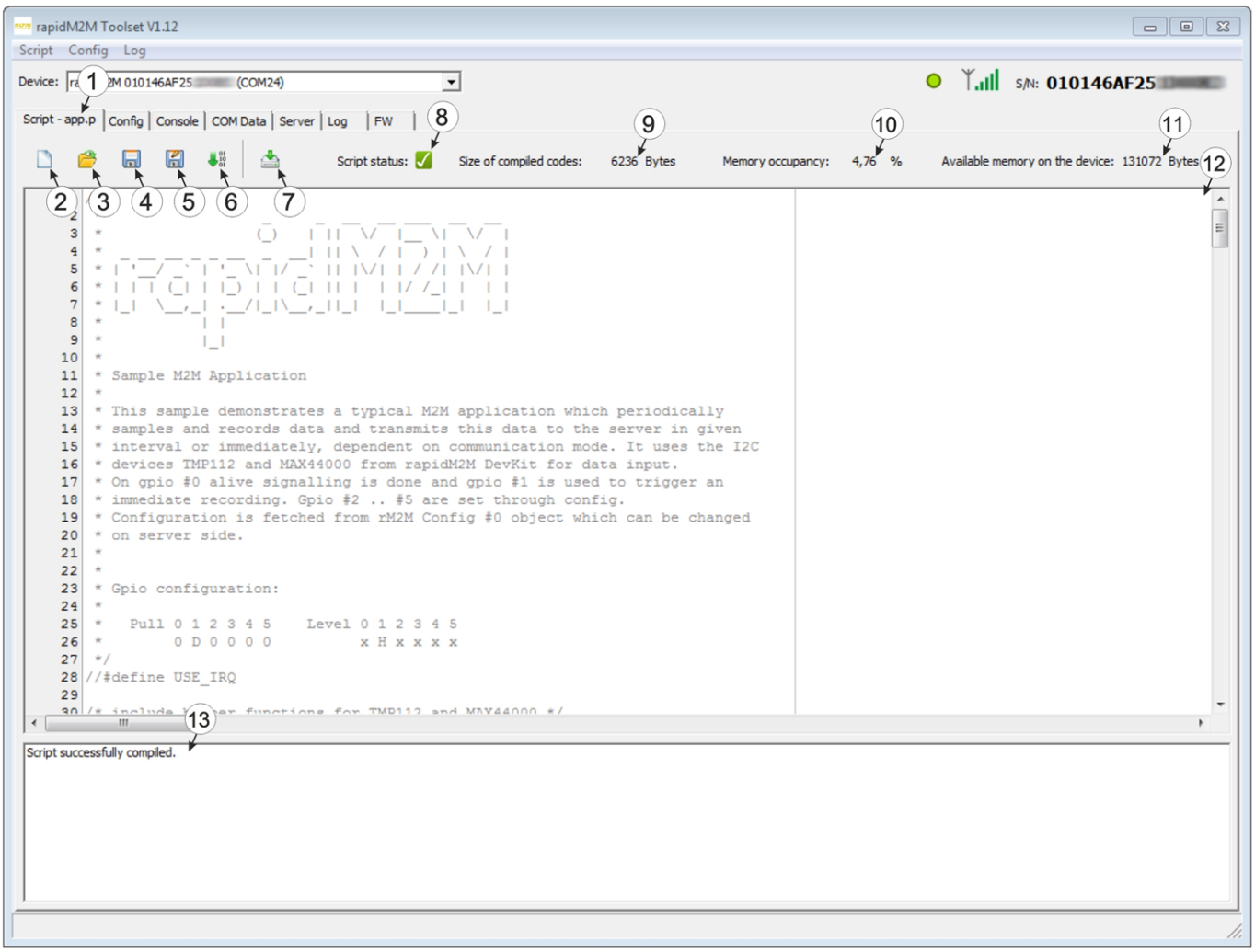
| Statussymbol | |
|---|---|
| grau | offline |
| grün blinkend | Verbindung wird hergestellt |
| grün | verbunden |
| rot | letzter Verbindungsversuch ist fehlgeschlagen |

### GSM-Signalstärke

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

# 9.6 Kateireiter "Script"

Dieser Bereich dient zum Erstellen des Scripts. Der Karteireiter ist auch verfügbar, wenn keine USB-Verbindung zu einem M120  Modul 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 |

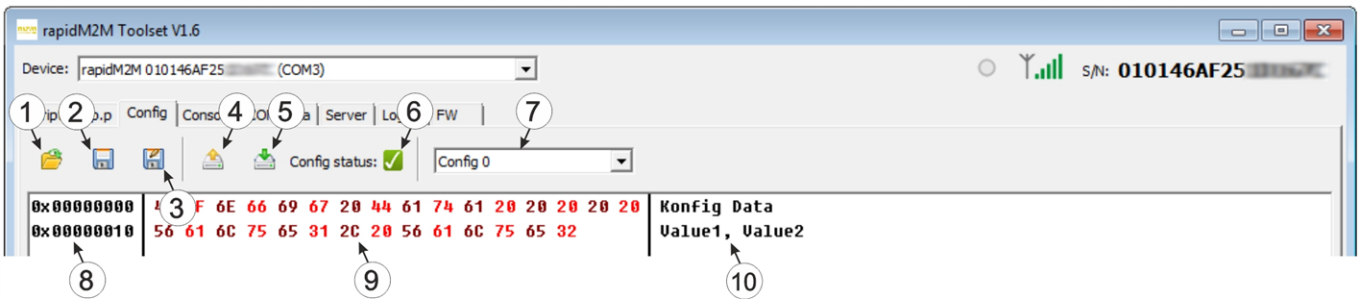
| | |
|---|---|
| 5 | öffnet den Dialog zum Speichern des Scripts unter einem anderen Dateinamen |
| 6 | kompiliert das geöffnete Script. Dabei wird es auch automatisch gespeichert. |
| 7 | überträgt das geöffnete Script in das Modul. Dabei wird das Script zuerst gespeichert und dann kompiliert. Das 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 M120 Modul 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 M120 Modul geladen. |
|  | Script wird gerade zum M120 Modul übertragen. |
|  | Der Zeitstempel des im M120 Modul gespeicherten kompilierten Scripts stimmt nicht mit dem des kompilierten Scripts des rapidM2M Toolset überein. |

# 9.7 Karteireiter "Config"

Dieser Karteireiter ermöglicht den Zugriff auf die 10 Konfigurationsspeicherblöcke. Diese können editiert, gespeichert und vom und zum M120 Modul übertragen werden. Der Karteireiter ist auch verfügbar, wenn keine USB-Verbindung zu einem M120 Modul besteht. In diesem Fall ist es allerdings nicht möglich, die Konfigurationsblöcke vom und zum M120 Modul 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 M120 Modul | 9 | Hex-Ansicht des Konfigurationsspeicherblocks |
| 5 | überträgt den ausgewählten Konfigurationsspeicherblock ans M120 Modul | 10 | ASCII-Ansicht des Konfigurationsspeicherblocks |

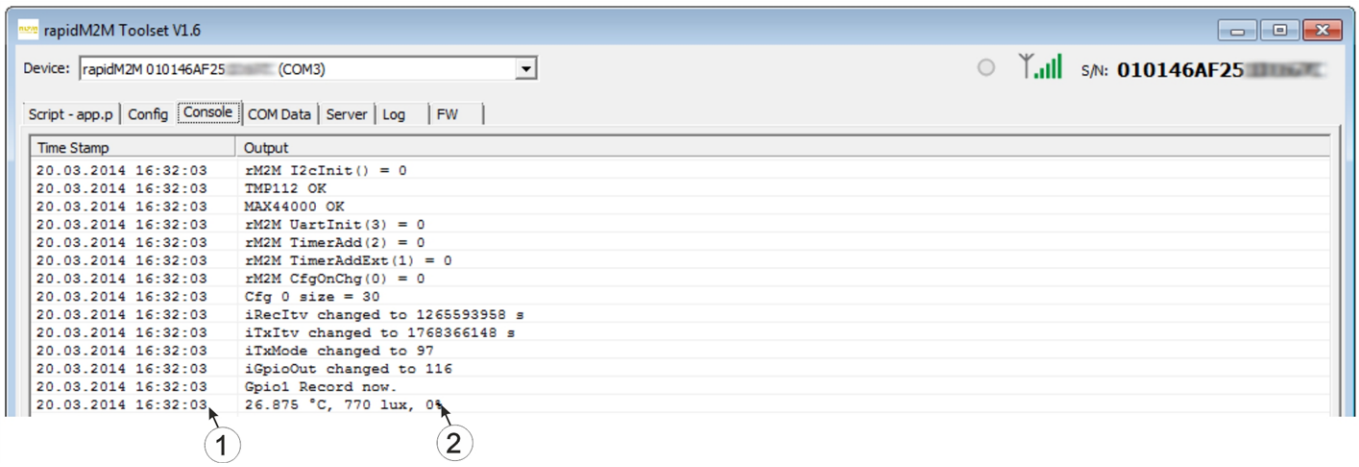
[1)] Die Operationen "öffnen", "speichern" und "übertragen" beziehen sich immer nur auf den ausgewählten Konfigurationsspeicherblock.

### Config Status

| Symbol | Beschreibung |
|---|---|
| ✅ | Der ausgewählte Konfigurationsspeicherblock wurde erfolgreich in das M120 Modul geladen. |
| ⚙ | Der ausgewählte Konfigurationsspeicherblock wird gerade übertragen. |
| ❗ | Der Zeitstempel des im M120 Modul gespeicherten Konfigurationsspeicherblock stimmt nicht mit dem des rapidM2M Toolset überein. Die Anzeige gilt für den aktuell ausgewählten Konfigurationsspeicherblock. |

## 9.8 Karteireiter "Console"

Dieser Bereich dient der Anzeige von Debug-Informationen, die innerhalb des PAWN-Scripts mittels der Funktionen "print" und "printf" (siehe "Consolen Funktionen" auf Seite 201) auf die Standardausgabe gedruckt wurden. Der Karteireiter "Console" ist nur verfügbar, wenn eine USB-Verbindung zu einem M120 Modul 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.

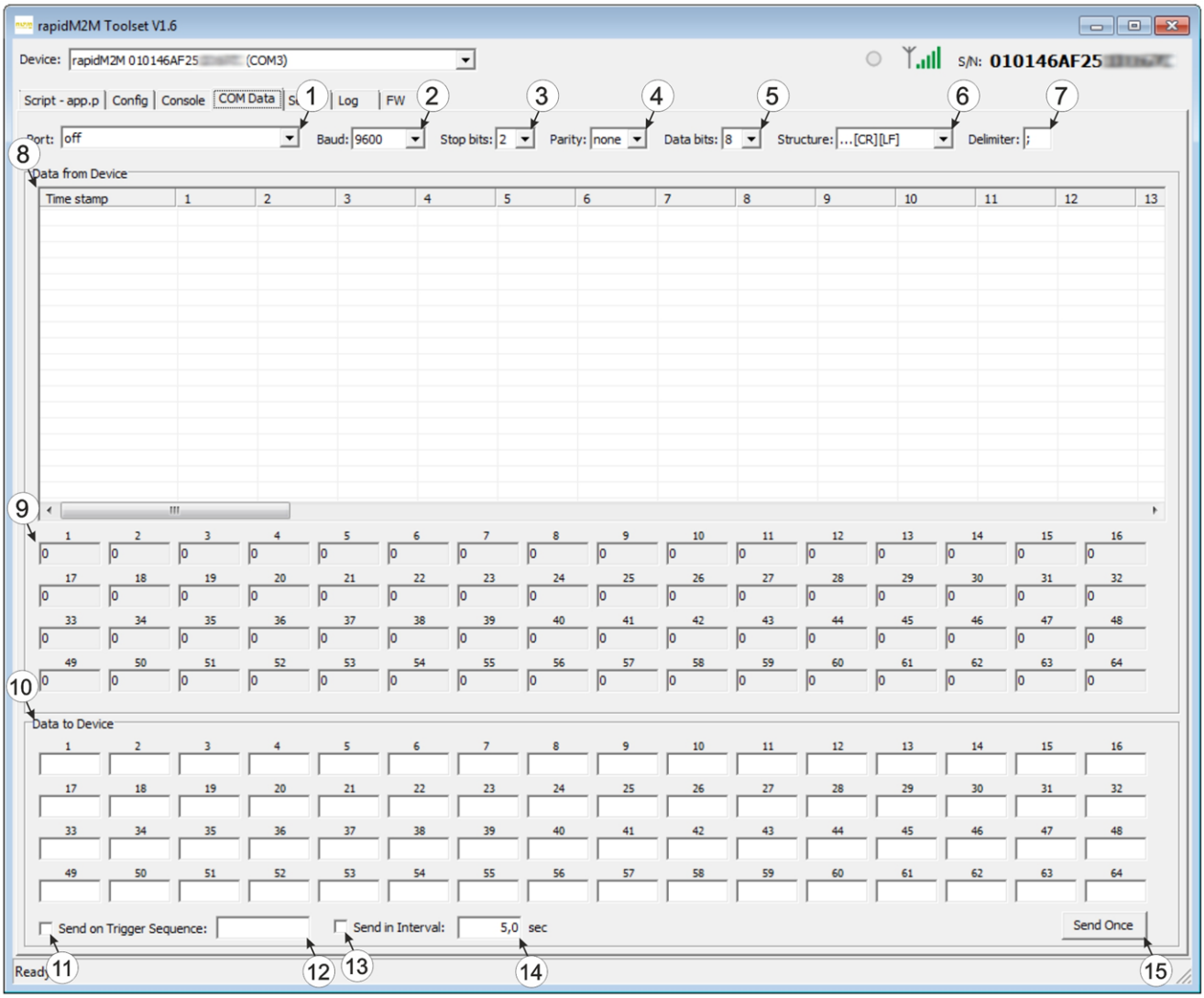
## 9.9 Karteireiter "COM Data"

Dieser Karteireiter dient zur Simulation eines einfachen seriellen Sensors. Dabei wird eine ASCII-Datenmessage erzeugt, die durch ein einstellbares Trennzeichen in einzelne Felder unterteilt ist. Die Frame-Struktur (Start- und Endekennung) ist ebenfalls auswählbar. Es gibt 3 Möglichkeiten den so konfigurierten Frame über die ausgewählte COM-Schnittstelle des PCs auszugeben:

- Einmalig mittels *"Send Once"* Button
- Im eingestellten Intervall (*Send in Interval*)
- Beim Empfang einer bestimmten Sequenz (*Send on Trigger Sequence*)

Wenn diese Möglichkeit genutzt werden soll, muss das rapidM2M Developer Paket M12x mit dem im Lieferumfang enthaltenen Nullmodemkabel mit der COM-Schnittstelle des PCs verbunden werden.

Der Karteireiter "COM Data" ist auch verfügbar, wenn keine USB-Verbindung zu einem M120 Modul besteht.




Karteireiter "COM Data"

| | | | |
|---|---|---|---|
| 1 | Auswahl der COM-Schnittstelle über die der Frame gesendet werden soll | 9 | Anzeige der Werte des zuletzt empfangenen Frames |
| 2 | zu verwendende Baudrate | 10 | Maske zur Eingabe der zu sendenden Werte |
| 3 | Anzahl der zu verwendenden Stoppbits | 11 | Der Frame wird vom PC nach dem Empfang der Trigger Sequenz versendet. |
| 4 | Auswahl der zu verwendenden Parität | 12 | Sequenz nach deren Empfang der Frame versendet werden soll |
| 5 | Anzahl der zu verwendenden Datenbits | 13 | der Frame wird zyklisch versendet |
| 6 | Konfiguration der Frame-Struktur (Start- und Endekennung), siehe "Frame-Struktur" | 14 | Intervall, in dem der Frame zyklisch versendet werden soll (Minimalwert 0,01 sec.) |
| 7 | zu verwendendes Trennzeichen | 15 | löst das einmalige Senden des Frames aus |
| 8 | Anzeige der empfangenen Frames[1)] | | |

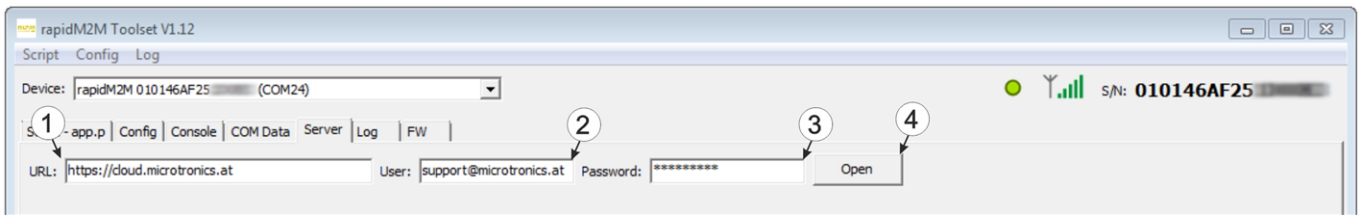
[1)] Durch einen Klick mit der rechten Maustaste auf einen der Einträge der Anzeige der empfangenen Frames wird ein Kontextmenü eingeblendet, dass das Löschen sowie das Speichern der Liste ermöglicht.

Frame-Struktur

| Structure | Beschreibung |
|---|---|
| ...[CR][LF] | Der Frame beginnt mit einer Pause und endet mit der Zeichenfolge [CR][LF] (oxoD)(oxoA). |
| ...[ETX] | Der Frame beginnt mit einer Pause und endet mit [ETX] (ox03). |
| [STX]...[CR][LF] | Der Frame beginnt mit [STX] (ox02) und endet mit der Zeichenfolge [CR][LF] (oxoD)(oxoA). |
| [STX]...[ETX] | Der Frame beginnt mit [STX] (ox02) und endet mit [ETX] (ox03). |

# 9.10 Karteireiter "Server"

Über diesen Karteireiter erreichen Sie die Testseite der REST-API des Cloud-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 M120 Modul 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 M120 Modul" auf Seite 153).




Karteireiter "Server" mit verbundenem M120 Modul

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

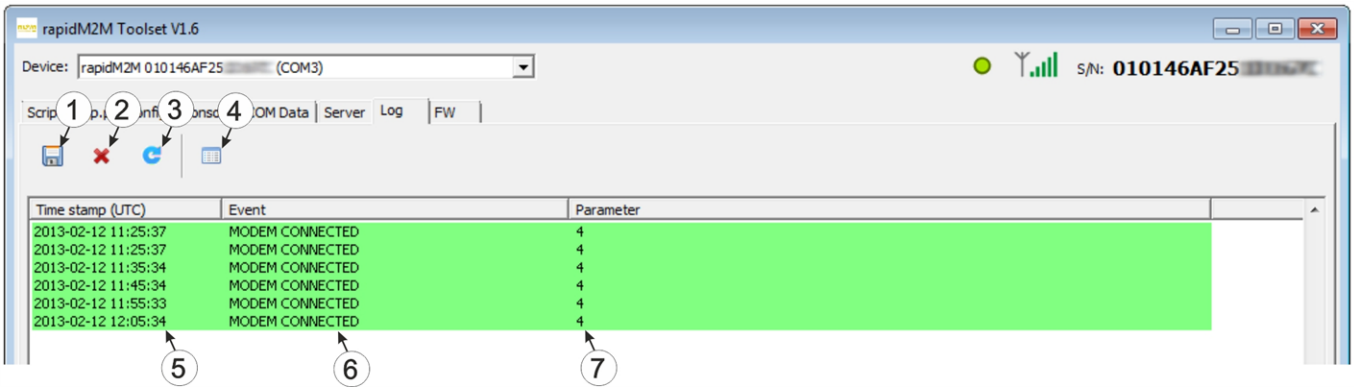



Karteireiter "Server" ohne verbundenem M120 Modul

| | |
|---|---|
| 1 | Seriennummer des M120 Moduls auf dessen Daten per REST-API zugegriffen werden sollen |

# 9.11 Karteireiter "Log"

Dieser Bereich dient der Verwaltung der Log-Einträge. Er ermöglicht das Laden der Einträge vom M120 Modul, das Speichern als *.tsv-Datei und das Löschen der Einträge aus dem Speicher des M120 Moduls.

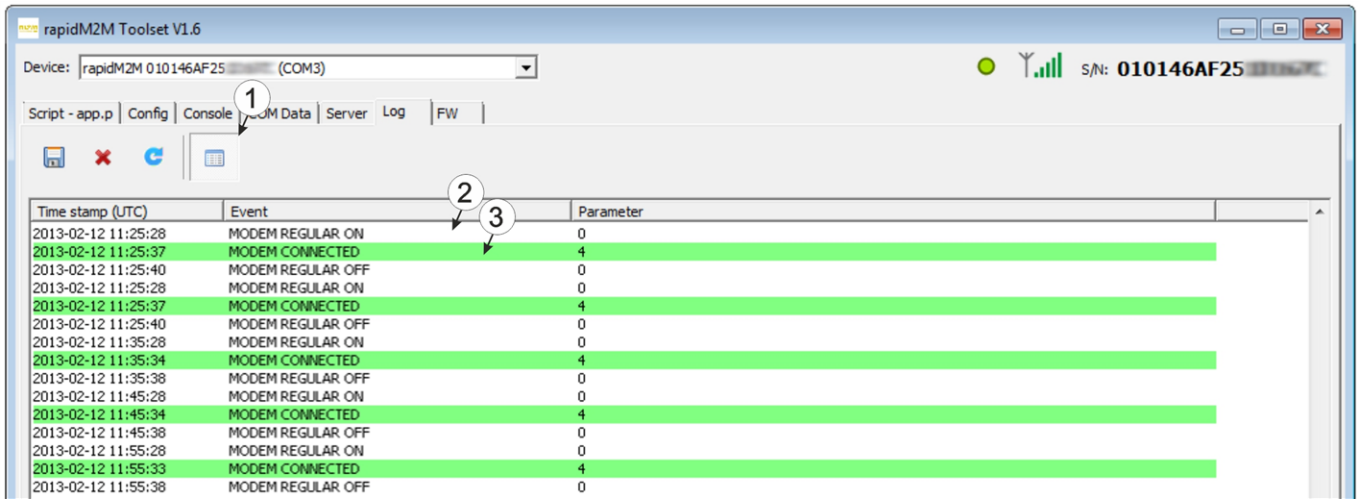



Karteireiter "Log"

| | | | |
|---|---|---|---|
| 1 | speichert die angezeigten Log-Einträge als *.tsv-Datei | 5 | Zeitstempel des Log-Eintrags |
| 2 | löscht die Log-Einträge aus dem Speicher des M120 Moduls | 6 | Log-Eintrag |
| 3 | lädt die Log-Einträge vom M120 Modul | 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 155).

| 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 | | |

## 9.12 Karteireiter "FW"

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



Karteireiter "FW"

| | | | |
|---|---|---|---|
| 1 | aktuell installierte Firmware | 3 | OS-Version des Modems |
| 2 | öffnet den Dialog zur Auswahl der Firmware-Datei | | |

## 9.13 Hot-Keys

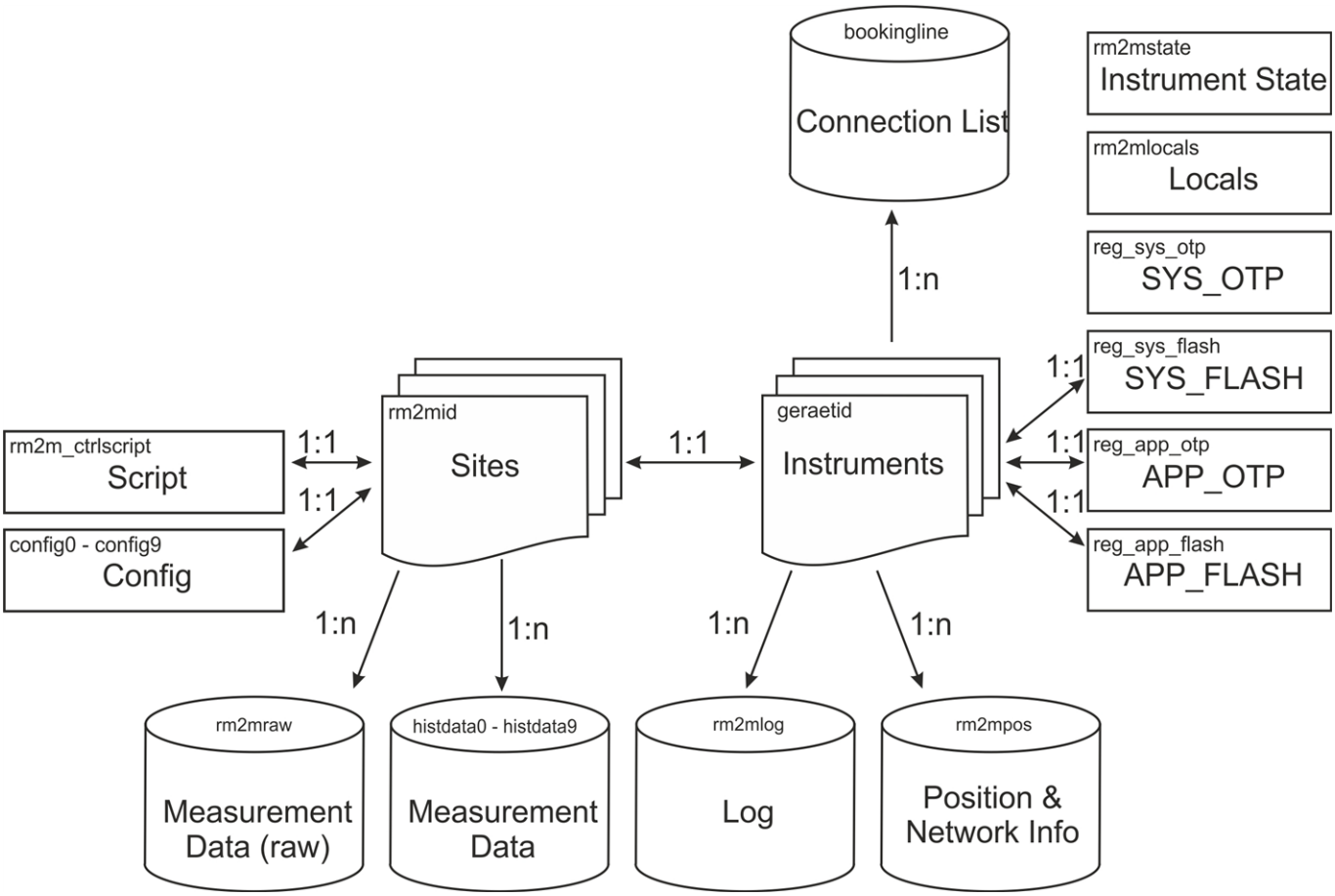
| Befehl | Tastenkombination | Erklärung |
|---|---|---|
| New Script | Ctrl+N | erstellt ein neues leeres Script |
| Open Script | Ctrl+O | öffnet ein gespeichertes Script |
| Save Script | Ctrl+S | speichert das aktuell geöffnete Script |
| Compile Script | Shift+F9 | kompiliert das geöffnete Script.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

# Kapitel 10 XML-Schnittstelle

## 10.1 Allgemein

Die XML-Schnittstelle dient dazu Daten aus Cloud-Server zu exportieren sowie Daten in den Cloud-Server zu importieren. Dies beschränkt sich jedoch nicht nur auf die reinen Messdaten sondern auf alle durch den Cloud-Server bereitgestellten Daten (z.B. Messdaten, Konfigurationen, Log-Einträge, Alarmlisten,...). Dadurch ist es dem Kunden möglich, komplett auf die Oberfläche des Cloud-Servers zu verzichten und seine eigene Benutzerschnittstelle zu erstellen. Dies kann zum Beispiel durch ein eigens entwickeltes PC-Programm oder ein Web-Interface erfolgen.

## 10.2 Aufbau der Datenbank




Aufbau der Datenbank

### Allgemein gültige Tabellenbezeichner

| Bezeichner | Beschreibung |
|---|---|
| rm2mid | Messstellenkonfiguration<br>(siehe "rm2mid" auf Seite 160) |
| geraetid | Geräte<br>(siehe "geraetid" auf Seite 164) |
| bookingline | Verbindungsliste<br>(siehe "bookingline (Read Only)" auf Seite 165) |
| | |

### Geräteabhängige Tabellenbezeichner

| Bezeichner | Zuordnung | Beschreibung |
|---|---|---|
| histdata0 | Messstelle | Messdaten 0 (strukturiert)<br>(siehe "histdata0 - histdata9" auf Seite 166 |
| histdata1 | Messstelle | Messdaten 1 (strukturiert) |
| histdata2 | Messstelle | Messdaten 2 (strukturiert) |
| histdata3 | Messstelle | Messdaten 3 (strukturiert) |
| histdata4 | Messstelle | Messdaten 4 (strukturiert) |
| histdata5 | Messstelle | Messdaten 5 (strukturiert) |
| histdata6 | Messstelle | Messdaten 6 (strukturiert) |
| histdata7 | Messstelle | Messdaten 7 (strukturiert) |
| histdata8 | Messstelle | Messdaten 8 (strukturiert) |
| rm2mlog | Gerät | Gerätelog<br>(siehe "rm2mlog" auf Seite 166) |
| rm2mstate | Gerät | Gerätestatus<br>(siehe "rm2mstate" auf Seite 166) |
| rm2mlocals | Gerät | Lokalisierungsinformationen<br>(siehe "rm2mlocals" auf Seite 167) |
| rm2mpos | Gerät | Positions- und Netzinformationen<br>(siehe "rm2mpos" auf Seite 167) |
| rm2m_ctrlscript | Messstelle | Byte-Code des Scripts<br>(siehe "rm2m_ctrlscript" auf Seite 168) |

| Bezeichner | Zuordnung | Beschreibung |
|---|---|---|
| reg_sys_otp | Gerät | Systeminformationen, die einmalig im Zuge der Produktion geschrieben werden<br>(siehe "reg_sys_otp" auf Seite 169) |
| reg_sys_flash | Gerät | Systeminformationen, die im Betrieb veränderbar sein müssen<br>(siehe "reg_sys_flash" auf Seite 169) |
| reg_app_otp | Gerät | applikationsspezifische Informationen, die einmalig im Zuge der Produktion geschrieben werden<br>(siehe "reg_app_otp" auf Seite 169) |
| reg_app_flash | Gerät | applikationsspezifische Informationen, die im Betrieb veränderbar sein müssen<br>(siehe "reg_app_flash" auf Seite 170) |
| config0 | Messstelle | Konfigurationsspeicherblock 0<br>(siehe "config0 - config9" auf Seite 170) |
| config1 | Messstelle | Konfigurationsspeicherblock 1 |
| config2 | Messstelle | Konfigurationsspeicherblock 2 |
| config3 | Messstelle | Konfigurationsspeicherblock 3 |
| config4 | Messstelle | Konfigurationsspeicherblock 4 |
| config5 | Messstelle | Konfigurationsspeicherblock 5 |
| config6 | Messstelle | Konfigurationsspeicherblock 6 |
| config7 | Messstelle | Konfigurationsspeicherblock 7 |
| config8 | Messstelle | Konfigurationsspeicherblock 8 |
| config9 | Messstelle | Konfigurationsspeicherblock 9 |
| rm2mraw | Messstelle | Messdaten (Rohformat)<br>(siehe "rm2mraw" auf Seite 170) |

Bei den geräteabhängigen Tabellenbezeichnern wird zwischen jenen unterschieden, die dem Gerät (geraetid) und jenen, die der Messstelle (rm2mid) zugeordnet sind. Wird die Zuordnung von Gerät und Messstelle aufgehoben, bleiben die dem Gerät zugeordneten Tabellen (z.B. das Gerätelog) fest mit dem Gerät verbunden und die der Messstelle zugeordneten Tabellen (z.B. die Messdaten) fest mit der Messstelle verbunden. D.h. wird einer Messstelle ein neues Gerät zugeordnet, können (wenn als Ausgangspunkt die Tabelle "geraetid" des neuen Geräts gewählt wird) auch die Messdaten ausgelesen werden, die erzeugt wurden als noch das zuletzt zugewiesene Gerät mit der Messstelle verbunden war.

Für den Aufbau der xPath Query (siehe "Syntax der xPath Query" auf Seite 171) ist die Kenntnis des Aufbaus der Datenbank erforderlich. Sollen z.B. die Messdaten zu einer bestimmten Seriennummer ausgelesen werden, führt der Weg zunächst über die Tabelle "geraetid", in der die Seriennummer enthalten ist, über die Tabelle "rm2mid", die die Verknüpfung zw. Gerät und Messstelle enthält und zur Tabelle "rm2mraw", die die gewünschten Messdaten enthält. Ein Beispiel dazu finden Sie unter t.b.d.

## 10.2.1 Allgemein gültige Tabellenbezeichner

### 10.2.1.1 rm2mid

Diese Tabelle enthält alle messstellenspezifischen Konfigurationen.

**name**

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

**timezone**

*Regionseinstellungen (nicht relevant für Rohmessdaten, da diese in UTC gespeichert werden)*

| *Wert* | *Erklärung* |
|---|---|
| *99,00* | *Die Regionseinstellung für die Messstelle wird von der globalen Servereinstellung übernommen.* |
| *-12,00* | *UTC -12 Stunden* |
| *:* | *:* |
| *+14,00* | *UTC +14 Stunden (Kiritimati)* |

**summertime**

*Konfiguration für automatische Zeitumstellung*

| *Wert* | *Erklärung* |
|---|---|
| **DEFAULT** | *Die Konfiguration für die Zeitumstellung wird von der globalen Servereinstellung übernommen* |
| **OFF** | *automatische Zeitumstellung deaktiviert* |
| *USA;99CC96152B000000* | *vordefinierte Einstellung für den amerikanischen Raum* |
| *EU;99CC95A627080884* | *vordefinierte Einstellung für den europäischen Raum* |

alert_mode

*Einstellung zur Quittierung der Alarme*

| *Wert* | *Erklärung* |
|---|---|
| *0* | *Für die Entscheidung, ob die Alarme automatisch oder manuell quittiert werden müssen, wird die globale Servereinstellung herangezogen.* |
| *1* | *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.* |
| *2* | *Alarme müssen durch den Anwender quittiert werden.* |

max_transfer_volume

*Einstellung für den Transfervolumenalarm*

| *Wert* | *Erklärung* |
|---|---|
| *-1* | *Die Einstellung für den Transfervolumenalarm wird von der globalen Servereinstellung übernommen.* |
| *0* | *Der Transfervolumenalarm ist deaktiviert.* |
| *› 0* | *Schwelle, bei der der Transfervolumenalarm ausgelöst werden soll [KB]* |

max_transfer_volume_kb

*Es handelt sich um ein "Ghost"-Feld, dessen Inhalt immer jenem des Feldes "max_transfer_volume" entspricht.*

comment

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

loc_site_lat

*Angabe der geographischen Breite, an der sich die Messstelle befindet [°]*

loc_site_long

*Angabe der geographischen Länge, an der sich die Messstelle befindet [°]*

loc_alarm_rad_mode

*Für eine Messstelle lässt sich ein Radius definieren. Sollte das Gerät diesen Radius verlassen, wird abhängig von der Einstellung dieses Parameters ein Alarm ausgelöst.*

| *Wert* | *Erklärung* |
|---|---|
| *0* | *Überwachung des Radius deaktiviert (kein Alarm)* |
| *› 0* | *Angabe des Alarm Radius [km]. Verlässt das Gerät den Radius, wird ein Alarm ausgelöst.* |

loc_alarm_rad

*Es handelt sich um ein "Ghost"-Feld, dessen Inhalt immer jenem des Feldes "loc_alarm_rad_modee" entspricht.*

view

*Auswahl der Auswertung, die durch einen Klick auf den Gerätelink in den Karten geladen wird*

| *Wert* | *Erklärung* |
|---|---|
| **OFF** | *Die Standardgrafik wird geladen.* |
| *"viewid"* | *Die ausgewählte Auswertung wird geladen. Um die "viewid" zu erhalten öffnen, Sie die gewünschte Auswertung (siehe "Bereich "Messstellen"" auf Seite 120) und entnehmen Sie die id aus der Adresszeile des Browsers. (z.B.....&viewid=**A1A5AD6CD299EF96**&....).* |

ftpexportprofile

*Auswahl des FTP Export Profils*

| *Wert* | *Erklärung* |
|---|---|
| *0* | *FTP Export deaktiviert* |
| *"id"* | *Um die "id" zu erhalten, öffnen Sie die Konfiguration des gewünschten FTP Export Profils (siehe Benutzerhandbuch für myDataweb-Server 206.886) und entnehmen Sie die id aus der Adresszeile des Browsers. (z.B.....&id=**A0B66D1CF7ED87DD**&....).* |

ftpexportdir

*ermöglicht es, das Standardverzeichnis des ausgewählten FTP-Export-Profils zu überschreiben [0-100 Zeichen]*

mappincaption

*gibt an, ob die Bezeichnung der Messstelle in der "Karte" angezeigt werden soll*

mapovpincaption

*gibt an, ob die Bezeichnung der Messstelle in der "Übersichts-Karte" angezeigt werden soll*

dbdesc

*Konfiguration des myDatanet Connector (siehe "myDatanet Connector " auf Seite 283) (max. 40kB)*

nameassitetype

*gibt an, ob der Name der Appliaktions-Vorlage aus der die Messstelle erstellt wurde als Messstellentyp in der Messstellenübersicht angezeigt werden soll*

| *Wert* | *Erklärung* |
|---|---|
| *0* | *Typ der Messstelle verwenden* |
| *1* | *Name der Appliaktions-Vorlage verwenden* |

app_version

*Aktuelle Version der Applikations-Vorlage. Bei jedem Speichern von neuen Einstellungen wird die Nummer automatisch erhöht. (Read Only)*

app_min_fw_modem
*Es kann eine geforderte Mindest-Modem-Version für die Applikations-Vorlage angegeben werden.*

*4 Zeichen Hex kodiert. Die ersten beiden Zeichen geben die Hauptversionsnummer und die letzten beiden die Nebenversionsnummer an (z.B. 010B für 01v011).*

app_min_fw_ctrl
*Es kann eine geforderte Mindest-Controller-Version für die Applikations-Vorlage angegeben werden.*

*4 Zeichen Hex kodiert. Die ersten beiden Zeichen geben die Hauptversionsnummer und die letzten beiden die Nebenversionsnummer an (z.B. 010B für 01v011).*

app_public
*gibt an, ob die Appliaktions-Vorlage für alle Kunden verfügbar ist.*

| *Wert* | *Erklärung* |
|---|---|
| *0* | *Verfügbarkeit auf einzelne Kunden beschränkt* |
| *1* | *für alle verfügbar (default)* |

mapview_showvaluesinmap
*gibt an, ob in der Kartendarstellung die Messwerte und die Bezeichnung der Messstelle angezeigt werden sollen.*

| *Wert* | *Erklärung* |
|---|---|
| *0* | *aus (default)* |
| *1* | *Messwerte und Messstellenbezeichnung anzeigen* |

mapview_clicktarget
*reserviert für Erweiterungen*

mapview_clicktarget_view
*reserviert für Erweiterungen*

cho - ch4
*reserviert für Erweiterungen*

binary_src
*gibt an, ob das Script zuletzt direkt über die Serveroberfläche eingegeben oder als File hochgeladen wurde*

| *Wert* | *Erklärung* |
|---|---|
| *0* | *Das Script wurde über die Oberfläche des Servers eingegeben.* |
| *1* | *Das Script wurde als Binary-File (*.amx) hochgeladen.* |

kunde
*eindeutige Datensatznummer des Kunden, dem die Messstelle zugewiesen ist (Read Only)*

geraet
*eindeutige Datensatznummer des Geräts, das der Messstelle zugewiesen ist (Read Only)*

template
*eindeutige Datensatznummer der Appliaktions-Vorlage aus der die Messstelle erstellt wurde (Read Only)*

#### 10.2.1.2 geraetid

Diese Tabelle enthält allgemeine Information und Konfigurationen, die für alle Geräte der myDatanet Familie gleich sind.

kunde
*eindeutige Datensatznummer des Kunden, dem das Gerät zugewiesen ist (Read Only)*

conn_path
*gibt den Kommunikationskanal an, über den die letzte Verbindung erfolgt ist*

| *Wert* | *Erklärung* |
|---|---|
| *0* | *GSM/GPRS* |
| *1* | *anderer Kommunikationskanal (z.B. LAN)* |

devsn
*Seriennummer des Geräts*

devclass
*Geräteklasse. Damit ein Gerät mit einer Messstelle verbunden werden kann, müssen die Geräteklasse der Messstelle und die des Geräts übereinstimmen.*

telnr
*Telefonnummer der SIM-Karte. An diese Nummer werden die Steuer-SMS (z.B. Wakeup) gesendet. Format: +43555837465[0-20 Zeichen]*

devflags
*zusätzliche Information zur Geräteklasse (für interne Verwendung) (Read Only)*

swversion
*aktuell installierte Softwareversion des Messcontrollers (Read Only)*

modemversion
*aktuell installierte Softwareversion des Modemcontrollers (Read Only)*

osversion
*OS-Version des Modems (Read Only)*

lastcon
*Zeitstempel des letzten Verbindungsaufbaus (Read Only)*

lastwakeup
*Zeitstempel des letzten Wakeups (Read Only)*

lastdisc
*Zeitstempel des letzten Verbindungsabbaus (Read Only)*

lasterr
*Zeitstempel des letzten Übertragungsfehlers (Read Only)*

lastaloha
*Zeitstempel der letzten Aloha Verbindung (Read Only)*

fw_update
*Einstellung für das Firmwareupdate*

| *Wert* | *Erklärung* |
|---|---|
| *0* | *aus: Firmware Update ist deaktiviert.* |
| *1* | *ein: Sobald eine neue Version des ausgewählten Firmware-Typs vorhanden ist, wird diese sofort installiert.* |
| *2* | *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!).* |
| *3* | *Downgrade erlauben: ermöglicht es, eine ältere Firmwareversion als die im Gerät vorhandene zu installieren (NICHT EMPFOHLEN!)* |
| *4* | *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.* |
| *5* | *ignorieren: Das Firmware Update ist deaktiviert und auf verfügbare Firmware Updates wird nicht hingewiesen.* |

fw_release_type
*Einstellung für den Firmwaretype*

| *Wert* | *Erklärung* |
|---|---|
| *0* | *Released: Nur Firmwareversionen bei denen sowohl interner Test als auch Feldtest erfolgreich waren, werden installiert (Fehlfunktionen nahezu ausgeschlossen).* |
| *1* | *Release Candidate: Nur Firmwareversionen bei denen der interne Test erfolgreich war, werden installiert (Fehlfunktionen nicht ausgeschlossen).* |
| *2* | *Beta Release: Auch Firmwareversionen bei denen noch nicht alle internen Tests erfolgreich abgeschlossen sind, werden installiert (Fehlfunktionen durchaus möglich).* |

### 10.2.1.3 bookingline (Read Only)

Diese Tabelle enthält Informationen zu den Verbindungen zwischen Geräten und dem Cloud-Server

kunde
*eindeutige Datensatznummer des Kunden, dem das Gerät zugewiesen ist*

geraet
*eindeutige Datensatznummer des Geräts, dem die Verbindungsliste zugewiesen ist*

text
*Dauer der Verbindung*

context
*Bezeichnung der Messstelle, mit der das Gerät verknüpft ist bzw. Fehler, der während der Verbindung aufgetreten ist*

source
*Seriennummer und IP-Adresse*

cmh_quantity
*Menge der eingehenden Daten in Byte (Gerät zum Server)*

cmh_code
*zeigt an ob das Gerät eine Verbindung zum Cloud-Server geöffnet hat (41=geschlossen, 942=verbunden)*

> ***Wichtiger Hinweis:*** *Tabelleneinträge mit anderen als den oben genannten Codes enthalten Informationen, die nur für die interne Verwendung bestimmt sind und die vom Kunden herausgefiltert werden sollten.*

## 10.2.2 Geräteabhängige Tabellenbezeichner

### 10.2.2.1 histdatao - histdata9

Diese Tabellen enthalten die Messdaten in strukturierter Form. Mit Hilfe des myDatanet Connector (siehe "myDatanet Connector " auf Seite 283) wird dem Cloud-Server die Struktur des Messdatenblocks der Tabelle "rm2mraw" bekanntgegeben und die Messwerte z.B. in die Tabelle "histdatao" kopiert. Aufgrund dessen können die Daten für Auswertungen, Ansichten, Grafiken, etc. am Cloud-Server verwendet werden.

flags
*reserviert für Erweiterungen*

cho - ch999
*Messwerte, die mittels myDatanet Connector (siehe "myDatanet Connector " auf Seite 283) definiert und aus der Tabelle "rm2mraw" kopiert wurden*

### 10.2.2.2 rm2mlog

Diese Tabelle beinhaltet die Log-Einträge und Fehlercodes (siehe "Log-Einträge und Fehlercodes " auf Seite 295) des Geräts.

log
*Log-Eintrag*

par
*Parameter zum Log-Eintrag*

### 10.2.2.3 rm2mstate

Diese Tabelle beinhaltet den Gerätestatus zum Zeitpunkt der letzten Übertragung. Alle Felder dieser Tabelle können nur gelesen und nicht verändert werden (Read Only).

**gsm_level**
*GSM-Feldstärke [dBm]*

**com_mode**
*Verbindungsart zum Zeitpunkt der letzten Verbindung*

**sim_id**
*IMSI des SIM-Chips*

**sim_apn**
*APN, der für die letzte Verbindung verwendet wurde*

**net_mcc**
*MCC (Mobile Country Code) des Netzes, das für die letzte Verbindung verwendet wurde*

**net_mnc**
*MNC (Mobile Network Code) des Netzes, das für die letzte Verbindung verwendet wurde*

**imei**
*IMEI des Modems*

**sim_icc_id**
*ICC-ID des SIM-Chips*

**idstring**
*rapidM2M Modulidentifikation (z.B. "rapidM2M M1 HW1.1")*

#### 10.2.2.4 rm2mlocals

Diese Tabelle beinhaltet die Lokalisierungsinformationen.

**time_offset**
*Differenz (in Sekunden) zwischen Systemzeit (UTC) und der für die Messstelle am Cloud-Server konfigurierten lokalen Zeit (inkl. Berücksichtigung von Sommer-/Winterzeit).*

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

#### 10.2.2.5 rm2mpos

Diese Tabelle enthält die Positions- und Netzinformationen. Das Erzeugen dieser Daten kann mittels Script-Funktion "rM2M_TxStart() ", "rM2M_SetPos()" oder "rM2M_SetPosNMEA()" ausgelöst werden.

**gps_lat**
*die ermittelte geographische Breite, an der sich das Gerät befindet [°]*

**gps_long**
*die ermittelte geographische Länge, an der sich das Gerät befindet [°]*

**gps_elev**
*die ermittelte Höhe über dem Meeresspiegel [m]*

gps_quality

*gibt die Qualität der ermittelten Positionsdaten an. Für Geräte ohne GPS-Empfänger wird die Position anhand der Informationen der GSM-Zellen im Empfangsbereich berechnet.*

| *Wert* | *Erklärung* |
|---|---|
| *0, 1* | *Position konnte nicht ermittelt werden.* |
| *10* | *Es konnte nur das Land, in dem sich das Gerät befindet, ermittelt werden.* |
| *20, 30* | *Die Position konnte auf ein größeres Gebiet innerhalb eines Landes eingegrenzt werden.* |
| *›=100* | *Position konnte aus den Informationen der GSM-Zellen im Empfangsbereich berechnet werden.* |
| *›31000* | *Position durch GPS-Empfänger bestimmt*[1)] |

[1)] *Das Gerät verfügt über keinen GPS-Empfänger.*

state

*reserviert für interne Verwendung*

Informationen über die 5 10 GSM-Zellen im Empfangsbereich, die für die Bestimmung der Position herangezogen werden:

gsm_mcc_0 - gsm_mcc_4 9

*MCC (Mobile Country Code) der GSM-Zelle*

gsm_rssi_0 - gsm_rssi_4 9

*empfangene GSM-Feldstärke [dBm] für die GSM-Zelle*

gsm_mnc_0 - gsm_mnc_4 9

*MNC (Mobile Network Code) der GSM-Zelle*

gsm_ta_0 - gsm_ta_4 9

*TA (Timing Advance) der GSM-Zelle*

gsm_lac_0 - gsm_lac_4 9

*LAC (Location Area Code) der GSM-Zelle*

gsm_cellid_0 - gsm_cellid_4 9

*Cell ID der GSM-Zelle*

#### 10.2.2.6 rm2m_ctrlscript

Diese Tabelle enthält die Auswahl des Script Typs (siehe "Steuerung" auf Seite 193) sowie das auf den Server geladene Binary des Scripts in Hex-Notation (2 Zeichen pro Byte), die zum rapidM2M Modul M120 übertragen werden.

script_error

*zeigt ein fehlerhaftes Script an (darf per XML nicht verändert werden)*

| *Wert* | *Erklärung* |
|---|---|
| *0* | *Das Script wird ordnungsgemäß ausgeführt.* |
| *1* | *Das Script wurde vom Gerät aufgrund eines Laufzeitfehlers deaktiviert. Nähere Informationen finden Sie im Gerätelog (siehe "Pawn Script Fehlercodes" auf Seite 255).* |

script_type

*Auswahl des Script Typs (darf per XML nicht verändert werden)*

| *Wert* | *Erklärung* |
|---|---|
| *0* | *Script deaktiviert* |
| *2* | *Aktiviert die Scriptabarbeitung und teilt dem rapidM2M Modul M120 mit, dass es sich bei dem Script um ein „Pawn Script“ handelt (siehe "Pawn Script" auf Seite 197).* |

#### 10.2.2.7 reg_sys_otp

Diese Tabelle beinhaltet die Systeminformationen, die einmalig im Zuge der Produktion geschrieben werden (Read Only).

json

*Inhalt des Registrierungsspeicherblocks "REG_SYS_OTP" als String unter Verwendung der JSON Notation (siehe "Registrierungsspeicherblöcke" auf Seite 63)*

#### 10.2.2.8 reg_sys_flash

Diese Tabelle beinhaltet die Systeminformationen, die im Betrieb veränderbar sein müssen (Read Only).

json

*Inhalt des Registrierungsspeicherblocks "REG_SYS_FLASH" als String unter Verwendung der JSON Notation (siehe "Registrierungsspeicherblöcke" auf Seite 63)*

#### 10.2.2.9 reg_app_otp

Diese Tabelle beinhaltet die applikationsspezifischen Informationen, die einmalig im Zuge der Produktion geschrieben werden(Read Only).

json

*Inhalt des Registrierungsspeicherblocks "REG_APP_OTP" als String unter Verwendung der JSON Notation (siehe "Registrierungsspeicherblöcke" auf Seite 63)*

#### 10.2.2.10 reg_app_flash

Diese Tabelle beinhaltet die applikationsspezifischen Informationen, die im Betrieb veränderbar sein müssen (Read Only).

**json**
: *Inhalt des Registrierungsspeicherblocks "REG_APP_FLASH" als String unter Verwendung der JSON Notation (siehe "Registrierungsspeicherblöcke" auf Seite 63)*

#### 10.2.2.11 configo - config9

Diese Tabelle enthält die für den Kunden frei definierbaren, voneinander unabhängigen Speicherblöcke für Konfigurationsdaten mit je 10kB . Um die Konfigurationsparameter über die Oberfläche des Cloud-Servers verändern bzw. anzeigen zu können, muss dem Cloud-Server der logische Aufbau der Konfiguratiosdatenblöcke bekannt sein. Dies erfolgt mit Hilfe des myDatanet Connector (siehe "myDatanet Connector " auf Seite 283).

**Fieldo - Field999**
: *Konfigurationsparameter, der mittels myDatanet Connector (siehe "myDatanet Connector " auf Seite 283) definiert wurde.*

#### 10.2.2.12 rm2mraw

Diese Tabelle enthält die Messdaten im Rohformat. Dem Cloud-Server ist der logische Aufbau des Messdatenblocks nicht bekannt. Um die Daten für Auswertungen, Ansichten, Grafiken, etc. am Cloud-Server verwenden zu können, müssen sie mit Hilfe des myDatanet Connector (siehe "myDatanet Connector " auf Seite 283) z.B. in die Tabelle "histdatao" kopiert werden.

**flags**
: *reserviert für Erweiterungen*

**data**
: *Messdatenblock (Rohformat)*

## 10.3 XML-Export

### 10.3.1 Aufbau des HTTP-GET-Kommandos

Eine Abfrage wird durch ein HTTP-GET Kommando durchgeführt, welches einer einfachen Eingabe des Befehls in die Adresszeile eines herkömmlichen Internet-Browsers entspricht. Die Befehlszeile (URL) setzt sich dabei wie folgt zusammen und liefert als Ergebnis ein XML-Dokument:

```
https://{URL}:{Port}/{XPath Query}?{login}{settings}{update}
```

| Parameter | Erklärung |
|---|---|
| {URL} | Adresse des Servers, auf dem das xPath Modul läuft (Cloud-Server) |
| {Port} | Port, auf dem der xPath Server läuft Die Standardeinstellung ist 8080. |
| {xPath Query} | xPath der gewünschten XML-Tabelle (z.B. "Aufbau der Datenbank" auf Seite 157). Eine Beschreibung der Syntax finden Sie unter "Syntax der xPath Query" auf Seite 171. |
| {login} | Benutzername und Passwort eines am Cloud-Server angelegten Benutzerkontos um die Abfrage zu authentifizieren (z.B. `user=u1&password=p1` ). |
| {settings} | OPTIONAL<br><br>Einstellungen, die das Ausgabeformat der Abfrage betreffen. Es gibt hier nur einen möglichen Parameter. Dieser deaktiviert die Anzeige der Feld-Attribute: `attributes=off`<br><br>Einstellungen, die die Ausgabe der Daten betreffen. Es gibt hier nur einen möglichen Parameter. Dieser gibt die Periode für die Datenverdichtung an: `period=...` (siehe "Datensatzverdichtung" auf Seite 176) |
| {update} | OPTIONAL<br><br>Update von Datenfeldern im Abfrageergebnis (z.B `&$field01=33`) (siehe "Update von Datenfeldern" auf Seite 178) |
| {data_type} | OPTIONAL<br><br>Auswahl des Ausgabeformats:<br>**0**<br>*XML-Dokument (siehe "Format des XML-Ergebnisses" auf Seite 181)*<br>**1**<br>*JSON-Dokument (siehe "Format des JSON-Ergebnisses" auf Seite 182)*<br><br>Ohne Angabe dieses Parameters wird als Ausgabeformat das XML-Dokument verwendet. |
| { verbose_time } | OPTIONAL<br><br>**1**<br>*Die Zeitstempel werden in erhöter Genauigkeit (ms) ausgegeben ( z.B.* `stamp =" 23.5.2012 15:21:09.348 "`*).* |

### 10.3.1.1 Syntax der xPath Query

#### 10.3.1.1.1 Auflistung der Felder einer Tabelle

Eine Auflistung der Felder einer Tabelle kann durch ein `@` vor der Tabellenbezeichnung ausgelesen werden.

`https://cloud.microtronics.at:8080/@geraetid?user=u1&password=p1`

Das oben aufgeführte Beispiel liefert eine Auflistung der Felder der Tabelle "Geräte" (*geraetid*):

```
<Result>
  <Field title="Kunde">kunde</Field>
  <Field title="Seriennummer">devsn</Field>
  <Field title="Geräteklasse">devclass</Field>
  <Field title="Firmware Version">swversion</Field>
  <Field title="Modem Version">modemversion</Field>
  <Field title="OS version">osversion</Field>
  <Field title="Letzter Verbindungsaufbau" units="<%data locals.time_code%> 
    <%data locals.offset_record.lastcon%>">lastcon</Field>
  <Field title="Letzter Wakeup" units="<%data locals.time_code%> 
    <%data locals.offset_record.lastwakeup%>">lastwakeup</Field>
  <Field title="Letzter Verbindungsabbau" units="<%data locals.time_code%> 
    <%data locals.offset_record.lastdisc%>">lastdisc</Field>
  <Field title="Letzter Übertragungsfehler" units="<%data locals.time_code%> 
    <%data locals.offset_record.lasterr%>">lasterr</Field>
  <Field title="Letzte Aloha Verbindung" units="<%data locals.time_code%> 
    <%data locals.offset_record.lastaloha%>">lastaloha</Field>
  <Field title="Firmware Update">fw_update</Field>
  <Field title="Firmware Typ">fw_release_type</Field>
</Result>
```

Eine Liste der möglichen Tabellenbezeichnungen finden Sie unter "Aufbau der Datenbank" auf Seite 157.

#### 10.3.1.1.2 Liste der Messstellen-Typen

Eine Liste aller vom Cloud-Server unterstützten Messstellen-Typen erhalten Sie bei Angabe der xPath Query *SiteTypes* :

```
https://cloud.microtronics.at:8080/SiteTypes?user=u1&password=p1
```

Ergebnis:

```
<Result>
  :
  <SiteType class="6700" title="rapidM2M"> FFFF6700 </SiteType>
  :
</Result>
```

#### 10.3.1.1.3 Filterbedingungen

Eine Filterbedingung ist immer von eckigen Klammern `[]` umschlossen und besteht aus Ausdrücken die durch die Schlüsselwörter *and* und *or* verknüpft werden können. Die Ausdrücke können mit runden Klammern `()` priorisiert werden. In der höchsten Ebene sind nur *and* Verknüpfungen zulässig. Ein Filterausdruck besteht aus einem Feldnamen, einem Operator und einer Konstante. Es können alle Felder der Tabelle verwendet werden.

> ***Hinweis:*** *Beim Namen der Messstelle werden Groß- und Kleinschreibung berücksichtigt. Beim Filter* `id` *und* `ref` *darf die Bedingung nicht mit Hochkomma ' angegeben werden.*

Zusätzlich zu den Feldern aus der Tabelle sind folgende Attribute verfügbar:

| Attribut | Erklärung |
|---|---|
| id | eindeutige Datensatznummer |
| stamp | Zeitstempel des Datensatzes (nur Operator >= und <= zulässig) |
| ref | Eindeutige Datensatznummer des Referenz-Datensatzes |

Es werden folgende Operatoren unterstützt: **=, <>, >, <, >=, <=**

Als Konstanten können folgende Typen verwendet werden:

| Datentyp | Erklärung |
|---|---|
| Integer | ganze Zahlen |
| Float | Fließkommazahlen |
| Strings | Texte, gekennzeichnet durch einfache Hochkomma ' |
| Date | Zeitstempel in einfachen Hochkomma ' |

Formate (Komma, Datumsformat, ...) werden von den Benutzer- bzw. von den Servereinstellungen übernommen.

#### 10.3.1.1.3.1 Beispiel

Ohne Filterbedingung wird die Konfiguration aller Messstellen des Typs rapidM2M geliefert.

```
https://cloud.microtronics.at:8080/rm2mid?user=u1&password=p1
```

Ergebnis:

```
<Result>
  <rm2mid id="A1526911CF9CCAA0" stamp="8.2.2013 11:48:16">
    <tag>FFFF6700</tag>
    :
    <name title="Name"> Messstelle1 </name>
    :
  </rm2mid>
  <rm2mid  id="A1617F03900A44F3"  stamp="8.2.2013 11:54:52" >
    <tag>FFFF6700</tag>
    :
    <name title="Name"> Messstelle2 </name>
    :
  </rm2mid>
</Result >
```

Soll nur die Konfiguration der Messstelle "Messstelle1" zurückgegeben werden, muss der Filter wie folgt gesetzt werden:

```
https://cloud.microtronics.at:8080/rm2mid[name='Messstelle1']?user=u1&password=p1
```

Ergebnis:

```
<Result>
  <rm2mid id="A1526911CF9CCAA0" stamp="8.2.2013 11:48:16">
    <tag>FFFF6700</tag>
    :
    <name title="Name"> Messstelle1 </name>
    :
  </rm2mid>
</Result >
```

#### 10.3.1.1.4 Felder

Wird mehr als ein Feld ausgelesen, so ist die Liste der Felder in runden Klammern `()` zu setzen und durch `|` oder `Blank` zu trennen. Wird kein Feld angegeben, so werden alle verfügbaren Felder ausgegeben, soweit es die Berechtigungsstufe des Benutzers zulässt.

> ***Hinweis:*** *Das Spezialfeld* `tag` *kann in jeder Tabelle abgefragt werden, es gibt den tag der abgefragten Tabelle zurück (z.B. FFFF0005 für den Tabellenbezeichner* `geraetid` *).*

##### 10.3.1.1.4.1 Beispiel

Das folgende Beispiel liefert die Namen aller Messstellen des Typs rapidM2M

```
https://cloud.microtronics.at:8080/rm2mid/(name)?user=u1&password=p1
```

Ergebnis:

```
<Result>
  <rm2mid id="A1526911CF9CCAA0" stamp="8.2.2013 11:48:16">
    <name title="Name">Messstelle1</name>
  </rm2mid>
  <rm2mid id="A1617F03900A44F3" stamp="8.2.2013 11:54:52">
    <name title="Name">Messstelle2</name>
  </rm2mid>
</Result>
```

#### 10.3.1.1.5 Datenabfrage mit Index-Angaben

Es werden nur die folgenden Index-Angaben unterstützt:

| Index | Erklärung |
|---|---|
| [0] | erster Datensatz (mit dem ältesten Zeitstempel) |
| [last()-1] | letzter Datensatz (mit dem jüngsten Zeitstempel) |

Andere Index-Angaben wie z.B. `[5]` oder `[last()-2]` sind ungültig.

#### 10.3.1.1.5.1 Beispiele

Das folgende Beispiel liefert den ersten Datensatz einer Messstelle des Typs rapidM2M deren Namen "Messstelle1" ist.

```
https://cloud.microtronics.at:8080/rm2mid[name='Messstelle1']/rm2mraw[0]
?user=u1&password=p1
```

Ergebnis:

```
<Result>
  <rm2mraw id="F000002D00000000" stamp="11.2.2013 14:39:31">
    <tag>000067FF</tag>
    <flags title="Flags">0000000000000000</flags>
    <data title="data">0000000018AD029241D88000443D000000000
       00100000000000000000000</data>
  </rm2mraw>
</Result>
```

Erklärung:

**rm2mid**
*gibt an, dass die Daten einer Messstelle des Typs rapidM2M abgefragt werden sollen*

**[name='Messstelle1']**
*gibt an, dass der Name der Messstelle, deren Daten abgefragt werden soll, "Messstelle1" ist.*

**rm2mraw**
*gibt an, dass die Daten aus der Tabelle Messdaten abgefragt werden sollen.*

**[0]**
*gibt an, dass der erste Datensatz (mit dem ältesten Zeitstempel) abgefragt werden soll.*

Das folgende Beispiel liefert den letzten Datensatz einer Messstelle des Typs rapidM2M , die mit dem Gerät mit der Seriennummer "010146AF25XXXXXX" verbunden ist.

```
https://cloud.microtronics.at:8080/geraetid[devsn='010146AF25XXXXXX']/
rm2mid/rm2mraw[last()-1]?user=u1&password=p1
```

Ergebnis:

```
<Result>
  <rm2mraw id="F000002D00000018" stamp="11.2.2013 15:02:42">
    <tag>000067FF</tag>
    <flags title="Flags">0000000000000000</flags>
    <data title="data">0000000318AD080141D9800044354000000000
      00000000000000000000000</data></rm2mraw>
</Result>
```

Erklärung:

**geraetid**
*gibt an, dass der Einstiegspunkt für die Abfrage die Gerätetabelle ist*

[devsn='010146AF25XXXXXX']
*gibt an, dass die Seriennummer des Geräts, für das die Daten abgefragt werden sollen, "010146AF25XXXXXX" ist.*

rm2mid
*gibt an, dass die Daten einer Messstelle des Typs rapidM2M abgefragt werden sollen.*

rm2mraw
*gibt an, dass die Daten aus der Tabelle "Messdaten" abgefragt werden sollen.*

[last()-1]
*gibt an, dass der letzte Datensatz (mit dem jüngsten Zeitstempel) abgefragt werden soll.*

#### 10.3.1.2 Datensatzverdichtung

> ***Hinweis:*** *Bei aktivierter Datensatzverdichtung ist die Ausgabe als JSON-Dokument nicht möglich.*

Durch das optionale Setting `period=...` kann die Datensatzverdichtung aktiviert werden. Dadurch werden alle Datensätze innerhalb der angegebenen Periode mit der pro Feld einstellbaren Berechnungsmethode verdichtet.

In der Feldliste (siehe "Felder" auf Seite 174) des xPath Query muss hinter jedem Feld in Klammern die Berechnungsmethode angegeben werden:

```
https://{IP-Adresse}:{Port}/{xPath Query Tabellenauswahl}/
(ch0(method=1)|ch0(method=2)|ch0(method=16;sum_mode=4))?{login}&period=15
```

Das oben aufgeführte Beispiel liefert 3 unterschiedliche Ergebnisse für die über 15min. verdichteten Werte des Kanals 0:

- cho(method=1): Minimum innerhalb der Verdichtungsperiode
- cho(method=2): Maximum innerhalb der Verdichtungsperiode
- cho(method=16;sum_mode=4): Tagessumme über die Verdichtungsperiode

Annahme: Der Wert des Kanals 0 steigt pro Minute um den Wert 1.

```
<Result>
  <record stamp="2013-01-25 11:30:00">
    <ch0_min>0,0</ch0_min>
    <ch0_max>14,0</ch0_max>
    <ch0_sum>105,0</ch0_sum>
  </record>
  <record stamp="2013-01-25 11:45:00">
    <ch0_min>15,0</ch0_min>
    <ch0_max>29,0</ch0_max>
    <ch0_sum>330,0</ch0_sum>
  </record><record stamp="2013-01-25 12:00:00">
    <ch0_min>30,0</ch0_min>
    <ch0_max>44,0</ch0_max>
    <ch0_sum>555,0</ch0_sum>
  </record>
</Result>
```

`period=...`

*Perioden auf die verdichtet werden soll in Minuten:*

*Beispiele:*

| | |
|---|---|
| *60* | *Stundenwerte* |
| *1440* | *Tageswerte* |
| *10080* | *Wochenwerte* |
| *44640* | *Monatswerte* |
| *133920* | *Quartalswerte* |

`method=…`

*Pro Kanal können die Verdichtungsmethoden angegeben werden:*

| | |
|---|---|
| *1* | *Minimum innerhalb der Verdichtungsperiode* |
| *2* | *Maximum innerhalb der Verdichtungsperiode* |
| *4* | *Durchschnitt über die Verdichtungsperiode* |
| *8* | *Median über die Verdichtungsperiode* |
| *16* | *Summe über die Verdichtungsperiode* |

*Es können mehrere Methoden gleichzeitig eingestellt werden. Dazu müssen die einzelnen Werte addiert werden. D.h.* `(ch0(method=1) | ch0(method=2))` *ist gleichbedeutend mit* `(ch0(method=3))` *.*

*Wird die Verdichtungsmethode "Summe"(16) aktiviert, so muss zusätzlich zur Verdichtungsmethode* `method=…` *auch noch die Methode zur Summenbildung* `sum_mode=…` *angegeben werden.*

`sum_mode=…`

| | |
|---|---|
| *1* | *1/sec (z.B. Ein Kanal wird in $m^3$/s aufgezeichnet; die Periodensumme ergibt $m^3$.)* |
| *2* | *1/min (z.B. Ein Kanal wird in $m^3$/min aufgezeichnet; die Periodensumme ergibt $m^3$.)* |
| *3* | *1/h (z.B. Ein Kanal wird in $m^3$/h aufgezeichnet; die Periodensumme ergibt $m^3$.)* |
| *5* | *1/Tag (z.B. Ein Kanal wird in $m^3$/Tag aufgezeichnet; die Periodensumme ergibt $m^3$.)* |
| *4* | *Summe eines Tageszählers (Tageszähler bedeutet, dass der Zählkanal einmal pro Tag zurückgesetzt wird. ) Über die Verdichtungsperiode wird jeweils die Differenz zwischen den Werten ($Wert_n$ - $Wert_{n-1}$) aufaddiert. Dabei wird auch der Rücksetzzeitpunkt des Tageszähles berücksichtigt.* |
| *6* | *Summe eines Intervallzählers (Intervallzähler bedeutet, dass der Zählkanal nach jeder Messung zurückgesetzt wird.) Die Werte werden über die Verdichtungsperiode aufaddiert.* |

#### 10.3.1.3 Update von Datenfeldern

Ein HTTP-GET Kommando kann auch dazu verwendet werden, um Felder eines Datensatzes upzudaten. Dazu muss das HTTP-GET Kommando um die jeweilige Feldzuweisung, beginnend mit einem Dollar-Zeichen `$` , erweitert werden. Es können beliebig viele Felder auf einmal überschrieben werden.

> ***Hinweis:*** *Für die Modifikation der Daten muss der Benutzer die entsprechenden Rechte besitzen (siehe t.b.d.).*

##### 10.3.1.3.1 Beispiele

Das folgende Beispiel schaltet für alle Geräte, die mit dem Messstellentyp rapidM2M kompatibel sind, das "Firmware Update" auf "ein" und setzt den "Firmware Typ" auf "Release Candidate".

```
https://cloud.microtronics.at:8080/geraetid[devclass=6700]?user=u1&password=p1
&$fw_update=1&$fw_release_type=1
```

Ergebnis:

```
<Result>
  <updated>2</updated>
</Result>
```

Erklärung:

**geraetid**
: *gibt an, dass der Einstiegspunkt für die Abfrage die Gerätetabelle ist.*

**[devclass=6700]**
: *gibt an, dass sich die Aktion auf alle Geräte, die mit dem Messstellentyp rapidM2M kompatibel sind, beziehen soll.*

**$fw_update=1**
: *gibt an, dass das "Firmware Update" auf "ein" gesetzt werden soll.*

**$fw_release_type=1**
: *gibt an, dass der "Firmware Typ" auf "Release Candidate" gesetzt werden soll.*

Das folgende Beispiel schaltet für das Gerät mit der Seriennummer "010146AF25XXXXXX" das "Firmware Update" auf "aus".

```
https://cloud.microtronics.at:8080/geraetid[devsn='010146AF25XXXXXX']
?user=u1&password=p1&$fw_update=0
```

Ergebnis:

```
<Result>
  <updated>1</updated>
</Result>
```

Erklärung:

**geraetid**
*gibt an, dass der Einstiegspunkt für die Abfrage die Gerätetabelle ist.*

**[devsn='010146AF25XXXXXX']**
*gibt an, dass sich die Aktion auf das Gerät mit der Seriennummer "010146AF25XXXXXX" beziehen soll.*

**$fw_update=0**
*gibt an, dass das "Firmware Update" auf "aus" gesetzt werden soll.*

#### 10.3.1.4 Datenabfrage mittels Streamingtechnologie

Neben dem aktiven Abfragen von einzelnen Datensätzen gibt es auch die Möglichkeit einen Stream zu öffnen. Dieser sendet solange Daten, bis er wieder aktiv beendet wird.

Für die Streamingtechnologie wird Ihnen eine Bibliothek zur Verfügung gestellt, die sich um das eigentliche Streaming und die damit verbundenen Eigenheiten der diversen Browser kümmert.

Diese Bibliothek steht unter der folgenden Adresse zum Download bereit:

**http://support.rapidm2m.com/lib/streaming.js**

**streamer_ = new rM2M . Streamer ( settings );**
*initalisieren des Streams mit den zuvor erstellten Settings*

| *Parameter* | *Erklärung* |
|---|---|
| *settings* | *Um den Stream starten zu können, muss der Methode "Streamer" ein Objekt mit den folgenden Eigenschaften übergeben werden.*<br><br>*Authentifizierung ist über user und password oder über userid möglich.D.h. das Objekt muss entweder die Eigenschaften "user" und "passwort" oder die Eigenschaften "userid" enthalten.*<br><br>**protocol**<br>*Protokoll, das verwendet werden soll (http oder https)*<br>**address**<br>*Adresse des Servers, von dem die Daten gestreamt werden sollen*<br>**port**<br>*Port, über den auf den Server zugegriffen werden soll*<br>**user**<br>*Benutzername des am Server angelegten Benutzers*<br>**password**<br>*Passwort des am Server angelegten Benutzers*<br>**userid**<br>*ID des am Server angelegten Users.* |

**streamer_. start (idSN, idContent);**

*starten des Streams*

| *Parameter* | *Erklärung* |
|---|---|
| *idSN* | *Seriennummer des Geräts (als String), von dem die Daten gestreamt werden sollen* |
| *idContent* | *Auswahl der Daten die gestreamt werden sollen. An dieser Stelle sind nur die hier vordefinierten Strings zulässig:*<br>**histdata0 - histdata9**<br>*Messdaten 0 - 9 (strukturiert)*<br>**config0 - config9**<br>*Konfigurationsspeicherblock 0 - 9*<br>**rm2mraw**<br>*Messdaten (Rohformat)* |

**streamer_ . stop ();**

*Stream beenden*

**streamer_.addEventListener ('data', function (name, param1) {//code here} );**

*Das Stream-Objekt unterstützt Events, die mit der Methode* `addEventListener` *registriert werden können.*

| *Parameter* | *Erklärung* |
|---|---|
| *data* | *Name des Events (siehe "Events" auf Seite 180)* |
| *function (name, param1)* | *Wenn ein Event auftritt, dann wird die entsprechend registrierte Funktion aufgerufen und ihr die folgenden beiden Parameter übergeben:*<br>**name**<br>*Der Name des Events („trying", „start", etc.). Dadurch können bei Bedarf alle Events in einer Funktion verarbeitet werden.*<br>**parm1**<br>*Enthält abhängig vom Event entsprechende Daten. Der Inhalt von param1 ist bei den Event-Erklärungen enthalten.* |

#### 10.3.1.4.1 Events

**trying**

*Es wird versucht, einen Stream mit der in param1 übergebenen Technologie zu starten. Die verfügbaren Technologien sind als Konstanten in rM2M.StreamTec hinterlegt.*

**start**
*Ein Stream wurde erfolgreich gestartet. Param1 enthält wie bei trying die Technologie.*

**stopping**
*Der Stopp-Vorgang eines Streams wurde initiiert. Param1 enthält die Technologie.*

**stop**
*Der Stream wurde gestoppt. Param1 enthält die Technologie.*

**data**
*Daten wurden empfangen. Param1 enthält ein Objekt mit den empfangenen Daten.*

#### 10.3.1.5 Ausgabeformate des HTTP-GET-Kommandos

Die Auswahl des Ausgabeformats erfolgt mit Hilfe des optionalen Parameters `{data_type}` (siehe "Aufbau des HTTP-GET-Kommandos" auf Seite 170). Ohne Angabe dieses Parameters wird als Ausgabeformat das XML-Dokument verwendet.

##### 10.3.1.5.1 Format des XML-Ergebnisses

Das Ergebnis ist ein XML-Dokument dessen Wurzel-Element `<Result>` ist. Pro Datensatz wird mindestens ein Child-Element mit dem Titel der Tabelle ausgegeben.

Die Datensätze haben generell folgende Attribute:

| Attribut | Erklärung |
|---|---|
| id | eindeutige Nummer des Datensatzes |
| stamp | Zeitstempel des Datensatzes |

Die Datenfelder haben folgende Attribute:

| Attribut | Erklärung |
|---|---|
| title | Titel des Datenfelds (bei Messdaten abhängig von der Einstellung der Messstelle) |
| units | Einheiten des Datenfeldes (bei Messdaten abhängig von der Einstellung der Messstelle) |
| calc | OPTIONAL<br><br>wird nur angezeigt, wenn es sich um einen vom Server berechneten Kanal handelt |

#### 10.3.1.5.1.1 Beispiel

```
<?xml version="1.0"?>
  <Result>
    <geraetid id="A15D6246F7AEBFAA" stamp="13.2.2013 09:10:05">
      <tag>FFFF0005</tag>
      <kunde title="Kunde">A1519985809987EC</kunde>
      <devsn title="Seriennummer">010146AF25XXXXXX</devsn>
      <devclass title="Geräteklasse">6700</devclass>
      <swversion title="Firmware Version">0000</swversion>
      <modemversion title="Modem Version">0005</modemversion>
      <osversion title="OS version">07.0474</osversion>
      :
      :
    </geraetid>
    <geraetid  id="A1526911CF1A42CC"  stamp="13.2.2013 09:33:54" >
      :
      :
    </geraetid>
  </Result>
```

#### 10.3.1.5.2 Format des JSON-Ergebnisses

Das Ergebnis ist ein JSON-Dokument, das ein oder mehrere Objekte enthält.

Jedes dieser Objekte besitzt 4 Eigenschaften:

| Eigenschaft | Erklärung |
|---|---|
| name | Tabellenbezeichner |
| id | eindeutige Identifikation des Datensatzes |
| stamp | Zeitstempel des Datensatzes |
| fields | Array aus Objekten, die die Felder der Tabelle repräsentieren |

Die Objekte, die die Felder der Tabelle repräsentieren, besitzen folgende Eigenschaften:

| Eigenschaft | Erklärung |
|---|---|
| name | Interne Feldbezeichnung |
| title | Titel des Datenfelds (bei Messdaten abhängig von der Einstellung der Messstelle) |
| units | Einheiten des Datenfeldes (bei Messdaten abhängig von der Einstellung der Messstelle) |
| value | Wert des Datenfeldes |
| calc | OPTIONAL<br><br>wird nur angezeigt, wenn es sich um einen vom Server berechneten Kanal handelt (nur bei Messdaten) |

***Hinweis:*** *Die Eigenschaften "name" und "value" sind immer vorhanden, die anderen Eigenschaften nur, falls benötigt.*

#### 10.3.1.5.2.1 Beispiel

> ***Hinweis:*** *In dem folgenden Beispiel wurden zur leichteren Lesbarkeit zusätzliche Zeilenumbrüche eingefügt.*

```
{name: 'geraetid', id: 'A15D6246F7AEBFAA', stamp: '3.5.2012%2018%3a12%3a25',
   fields: [{name: 'tag', value: 'FFFF0005'},
            {name: 'kunde', title: 'Kunde', value: 'A1519985809987EC'},
            {name: 'devsn', title: 'Seriennummer', value: '010146AF25XXXXXX'},
            {name: 'devclass', title: 'Ger%c3%a4teklasse', value: '6700'},
            {name: 'swversion', title: 'Firmware%20Version', value: '0000'},
            {name: 'modemversion', title: 'Modem%20Version', value: '0005'},
            {name: 'osversion', title: 'OS%20version', value: '07.0474'}, ... ]}
{name: 'geraetid', id: 'A1526911CF1A42CC', stamp: '8.11.2012%2017%3a15%3a02',
    fields: [ ... ]}
```

# 10.4 XML-Import

> ***Wichtiger Hinweis:*** *Der XML-Import ist nicht bei allen Cloud-Servern verfügbar. Sollte der directlogin-Parameter, der den Benutzer am Server eindeutig identifiziert, in der Eingabemaske zur Bearbeitung der Benutzerdaten (siehe "Benutzerhandbuch für myDataweb-Server " 206.886) nicht angezeigt werden, ist der Import von Daten nicht möglich.*

Der XML-Import dient dazu, die über die vom Kunden erstellte Benutzerschnittstelle erfassten oder veränderten Daten wieder zurück in den Cloud-Server zu importieren.

- Es können nur vorhandene Messdaten und Einstellungen geändert bzw. neue Messdaten angefügt werden. Es können keine historischen Messdaten eingefügt werden!
- Es ist darauf zu achten, dass der `stamp` der Datensätze immer aufsteigend angelegt wird. Es kann zu Fehlern kommen, wenn dies nicht der Fall ist!
- Werden Einstellungen geändert, muss darauf geachtet werden, dass der `stamp` auf das aktuelle Datum/Uhrzeit gesetzt wird. Ansonsten werden die Daten nicht mit dem Gerät synchronisiert.
- Beim Anlegen von Messdaten muss immer die `id="0"` und der `<tag>00006700</tag>` (67 steht für die Geräteklasse) angegeben werden.
- Beim Löschen muss immer `<tag>0</tag>` angegeben werden. Das Löschen von historischen Daten (z.B. Messdaten) ist nicht möglich!

## 10.4.1 Übertragung der Daten an den Cloud-Server

Für die Übertragung einer XML-Datei zum Import am Cloud-Server kann ein HTTP-Post Kommando verwendet werden.

Ein HTTP-POST Kommando kann zum Beispiel mit dem frei verfügbaren Kommandozeilen-Tool "Wget" (http://www.gnu.org/software/wget/) durchgeführt werden.

Beispiel für die Übertragung einer XML-Datei mittels "Wget":

```
Verwendung: wget [Optionen] [URL]

wget --post-file=file.xml --output-document=response.txt --no-check-certificate
"https://cloud.microtronics.at/send.xml?directlogin={login}"
```

| Parameter | Erklärung |
|---|---|
| --post-file= | Verwendung der POST-Methode um eine Datei zu senden |
| file.xml | Datei, die die zu übertragenden Daten enthält. Details zum Aufbau der Datei finden Sie unter "Aufbau der XML-Datei" auf Seite 186. |
| --output-document= | OPTIONAL<br>Die Antwort des Servers soll in der angegebenen Datei gespeichert werden. |
| response.txt | OPTIONAL<br>Name der Datei, in der die Antwort des Servers gespeichert werden soll. Konnten die Daten erfolgreich übertragen werden, enthält diese Datei folgenden Eintrag: "`+OK: n Records processed`", wobei "n" für die Anzahl der übernommen Einträge steht. |
| --no-check-certificate | ***Wichtiger Hinweis:*** *Deaktiviert die Überprüfung der Zertifikate. Nicht für den Dauerbertrieb empfohlen. Nur für Testzwecke!*<br>"Wget" ist nicht in der Lage auf das Verzeichnis in dem Windows die Zertifikate speichert, zuzugreifen. Diese müssen in den Programmordner von "Wget" kopiert werden. Informationen dazu entnehmen Sie bitte der Dokumentation von "Wget". |
| "https://...." | HTTP-POST Kommando. Für Details siehe "Aufbau des HTTP-POST-Kommandos" auf Seite 185. |

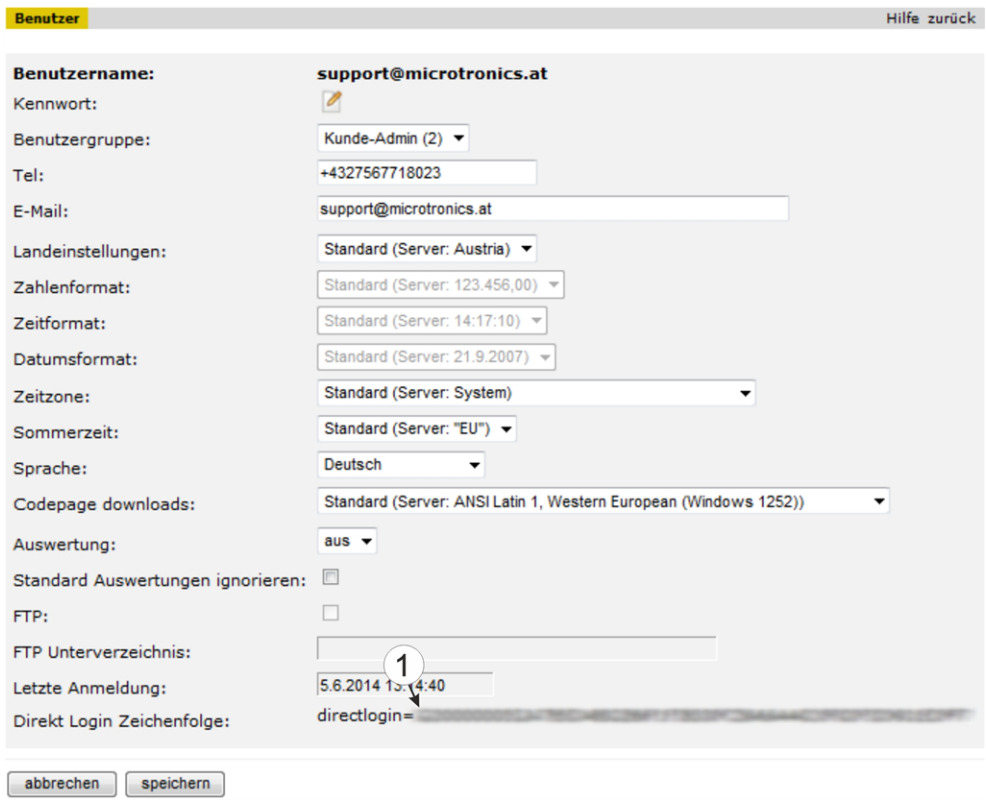
### 10.4.1.1 Aufbau des HTTP-POST-Kommandos

> ***Wichtiger Hinweis:*** *Die Übermittlung von XML-Dateien an den Server erfolgt nicht an den xPath-Server (Standard-Port 8080), sondern an den HTTP-Server (Standard-Port 80).*

Das HTTP-POST Kommando setzt sich wie folgt zusammen:

```
https://{URL}[:{Port}]/send.xml?directlogin={login}
```

Mit diesem POST-Kommando muss die XML-Datei übermittelt werden. Der Aufbau dieser wird im Kapitel "Aufbau der XML-Datei" auf Seite 186 beschrieben.

| Parameter | Erklärung |
|---|---|
| {URL} | Adresse des Servers, an den die Daten übermittelt werden sollen (Cloud-Server) |
| {Port} | OPTIONAL<br><br>Port, auf dem der HTTP-Server läuft. Standardwert ist 80. |
| {login} | Der directlogin-Parameter identifiziert den Benutzer am Server eindeutig. Er kann nur vom jeweiligen Benutzer für sich selber abgefragt werden. Hierfür muss der Benutzer am Cloud-Server im Bereich "Benutzer" die Eingabemaske zur Bearbeitung seiner Benutzerdaten (siehe "Benutzerhandbuch für myDataweb-Server " 206.886) öffnen. Dort befindet sich im unteren Drittel die "Direkt Login Zeichenfolge" (1).<br><br> |

#### 10.4.1.2 Aufbau der XML-Datei

##### 10.4.1.2.1 Ändern von vorhandenen Daten

Soll ein bestehender Datensatz geändert werden, muss als Attribut `id` die eindeutige Identifikationsnummer des Datensatzes, der geändert werden soll, angegeben werden (z.B. `id ="F000002200000282"` ). Die XML-Datei muss nur jene Elemente, enthalten die auch verändert werden sollen. Alle nicht enthaltenen Elemente bleiben unverändert. Die Attribute (`title` , `units` , `calc` , ...) sind für den Datenimport nicht relevant und können weggelassen werden.

> ***Hinweis:*** *Formate (Komma, Datumsformat, ...) werden von den Benutzereinstellungen bzw. von den Servereinstellungen übernommen.*

```
<?xml version="1.0"?>
<Result>
  <rm2mraw  id="F000002200000282" >
    <data>
        0000003B188A5F8541BF8000441700000000000000000000000000000
        0000000000000000000000000000000000000000000000000000000
        0000000000000000000000000000000000000000000000000000000
        0000000000000000000000000000000000000000000000000000000
        00000000000000000000000000000000
    </data>
    </rm2mraw >
</Result>
```

**rm2mraw**
*für den Datenimport nicht relevant. Es empfiehlt sich jedoch die entsprechende Tabellenbezeichnung zu benutzen.*

**id**
*eindeutige Identifikationsnummer des zu ändernden Datensatzes*

**data**
*Wert, auf den die Daten geändert werden sollen*

##### 10.4.1.2.2 Import von Daten

Soll ein neuer Datensatz angelegt werden, muss das Attribut `id="0"` angegeben werden. Über das Attribut `stamp` kann der Zeitstempel für den Datensatz festgelegt werden. Es ist darauf zu achten, dass die Zeitstempel der Daten immer aufsteigend sind. Das Attribut `stamp` kann auch weg gelassen werden. In diesem Fall wird die aktuelle Uhrzeit als Zeitstempel für den Datensatz verwendet. Für alle Elemente, die nicht in der XML-Datei angegeben werden, werden bei der Neuanlage des Datensatzes die Defaultwerte des Cloud-Servers verwendet. Das `tag` Element ist ein Pflichtfeld und gibt die Tabelle an, in welche der Datensatz gespeichert wird. Die Verknüpfung des Datensatzes mit der Messstelle (wenn der Datensatz in einer der Messstelle zugeordneten Tabelle gespeichert wird) bzw. des Geräts (wenn der Datensatz in einer dem Gerät zugeordneten Tabelle gespeichert wird), wird über das `context` Element angegeben. Dabei handelt es sich um die eindeutige Datensatzidentifikationsnummer (`id` ) der Messstelle bzw. des Geräts. Die Attribute (`title` , `units` , `calc` , ...) sind für den Datenimport nicht relevant und können weggelassen werden.

***Hinweis:*** *Formate (Komma, Datumsformat, ...) werden von den Benutzereinstellungen bzw. von den Servereinstellungen übernommen.*

Messstelle, mit der der Datensatz verknüpft werden soll:

```
<Result>
  <rm2mid  id="A1526911CF9CCAA0"  stamp="13.2.2013 10:00:54" >
    <tag> FFFF6700 </tag>
    <name title="Name"> Messstelle1 </name>
    :
    :
  </rm2mid >
</Result>
```

XML-Datei:

```
<?xml version="1.0"?>
<Result>
  <rm2mraw  id="0" stamp="16.1.2013 09:00">
    <data>
        0000003B188A5F8541BF8004417000000000000000000000000000000
        000000000000000000000000000000000000000000000000000000000
        000000000000000000000000000000000000000000000000000000000
        000000000000000000000000000000000000000000000000000000000
        00000000000000000000000000000000000
    </data>
    <tag>00006700</tag>
    <context>A1526911CF9CCAA0</context>
    </rm2mraw >
</Result>
```

### rm2mraw

*für den Datenimport nicht relevant. Es empfiehlt sich jedoch die entsprechende Tabellenbezeichnung zu benutzen.*

### id

*Beim Import von Daten muss immer* `id="0"` *angegeben werden. Ein neuer Datensatz mit einer automatisch generierten ID wird angelegt.*

### stamp

*OPTIONAL*

*Zeitstempel, der für den Datensatz verwendet werden soll. Es ist darauf zu achten, dass die Zeitstempel der Daten immer aufsteigend sind. Wird dieses Attribut weggelassen, wird die aktuelle Uhrzeit als Zeitstempel für den Datensatz verwendet.*

### data

*die eigentlichen Messdaten, die importiert werden sollen.*

### tag

*gibt die Tabelle an, in welcher der Datensatz gespeichert wird*

### context

*gibt an, mit welcher Messstelle bzw. mit welchem Gerät der Datensatz verknüpft wird (siehe "Messstelle mit der der Datensatz verknüpft werden soll")*

6872
6872
6872

# Kapitel 11 Inbetriebnahme

## 11.1 Inbetriebnahme des Systems

1. Loggen Sie sich am Cloud-Server ein. Sollten Sie noch nicht über einen Account am Cloud-Server verfügen, erstellen Sie zunächst einen Account (siehe "Neu registrieren und anmelden" auf Seite 117).
2. Legen Sie eine Messstelle für den Betrieb am Cloud-Server an (siehe "Anlegen einer neuen Messstelle" auf Seite 125).
3. Führen Sie alle im Kapitel "Anschluss der Sensoren, der Aktoren und der Versorgung" auf Seite 94 beschriebenen Schritte durch. Danach ist das rapidM2M Modul M120 betriebsbereit und sollte bereits die erste Verbindung zum Cloud-Server durchgeführt haben.
4. Laden Sie das gewünschte Script mittels rapidM2M Toolset (siehe "Kateireiter "Script"" auf Seite 148) in das rapidM2M Modul M120 .

   ***Wichtiger Hinweis:*** *Achten Sie darauf, dass bevor Sie ein Script über die USB-Schnittstelle ins rapidM2M Modul M120 einspielen, eine Messstelle für den Betrieb am Cloud-Server angelegt wurde und das Gerät bereits ein Mal einen Abgleich mit dieser Messstelle durchgeführt hat (siehe "Kommunikation mit dem Gerät testen" auf Seite 190). Andernfalls wird das über USB eingespielte Script beim ersten Abgleich mit der Messstelle gelöscht.*

Der folgende Schritt ist nicht zwingend erforderlich.

5. Überprüfen Sie, ob die Verbindung zum Cloud-Server und die Übertragung der Daten korrekt funktioniert hat (siehe "Kommunikation mit dem Gerät testen" auf Seite 190).

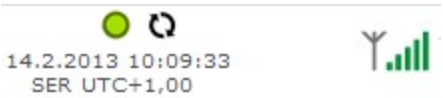
## 11.2 Kommunikation mit dem Gerät testen

1. Legen Sie eine Messstelle für den Betrieb am Cloud-Server an (siehe "Anlegen einer neuen Messstelle" auf Seite 125).
2. Versorgen Sie das rapidM2M Developer Board entweder über den DC-Hohlstecker oder die Klemmleiste (siehe "Anschluss der Sensoren, der Aktoren und der Versorgung" auf Seite 94).

   Sollte der Jumper zum Einschalten des Wireless GSM-Controllers (J37 "MODUL ON") geschlossen sein (Auslieferungszustand, siehe "Übersicht" auf Seite 67), startet der Controller des M120 Moduls nach dem Herstellen der Energieversorgung des Developer Boards sobald er das "Modul_ON" Signal erkannt hat (ca. 1,5sec. nach Herstellung der Energieversorgung). Daraufhin sollte die Status-LED ca. 1sec. leuchten, um den Boot-Vorgang anzuzeigen (siehe "Status-LED (LED100)" auf Seite 80). Anschließend bucht sich das rapidM2M Developer Paket M12x ins Netz ein und nimmt die Verbindung zum Cloud-Server auf.

Der folgende Schritt ist nur erforderlich, wenn der Jumper zum Einschalten des Wireless GSM-Controllers (J37 "MODUL ON") nicht geschlossen ist.

3. Drücken Sie als nächstes die Taste zum Einschalten des Wireless GSM Controllers S101 (siehe "Taster zum Einschalten des Wireless GSM-Controllers am rapidM2M Modul M120 (S101)" auf Seite 81). Daraufhin sollte die Status-LED ca. 1sec. leuchten, um den Boot-Vorgang anzuzeigen (siehe "Status-LED (LED100)" auf Seite 80). Anschließend bucht sich das rapidM2M Developer Paket M12x ins Netz ein und nimmt die Verbindung zum Cloud-Server auf.
4. Warten Sie bis in der Messgeräteliste (siehe "Messstellen" auf Seite 123) 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 M120 Modul mit dem Server verbunden ist sehr kurz. Daher kann auch geprüft werden ob der Zeitstempel der letzten Verbindung (unter dem grünen Statussymbol) aktualisiert wurde.

   Alternativ dazu kann die Statusanzeige des rapidM2M Toolset (siehe "Übersicht" auf Seite 146) ausgewertet werden.

   Bewerten Sie in beiden Fällen auch die GSM-Signalstärke (Antennensymbol mit Signalstärkebalken).

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

5. Laden Sie das gewünschte Script mittels rapidM2M Toolset (siehe "Kateireiter "Script"" auf Seite 148) in das rapidM2M Modul M120 .

   ***Wichtiger Hinweis:*** *Achten Sie darauf, dass bevor Sie ein Script über die USB-Schnittstelle ins rapidM2M Modul M120 einspielen, eine Messstelle für den Betrieb am Cloud-Server angelegt wurde und das Gerät bereits ein Mal einen Abgleich mit dieser Messstelle durchgeführt hat. Andernfalls wird das über USB eingespielte Script beim ersten Abgleich mit der Messstelle gelöscht.*

6. Verdrahten Sie anschließend die Sensoren (siehe "Anschluss der Sensoren, der Aktoren und der Versorgung" auf Seite 94).

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

7. Für die Überprüfung der Datenübertragung stehen 3 Möglichkeiten zur Verfügung:
   - Mittels rapidM2M Toolset über den Kateireiter "Server" (siehe "Karteireiter "Server"" auf Seite 153)
   - Mit Hilfe der Beispiel Web-Applikation "Data Streaming Sample", die dem Kunden als Anleitung zum Erstellen einer eigenen Benutzerschnittstelle dienen soll (siehe http://support.rapidm2m.com/lib/test.php t.b.d.):
   - Mittels Cloud-Server unter der Verwendung der "Auswertungen" (siehe "Auswertungen" auf Seite 1219. Dazu ist allerdings die Konfiguration des myDatanet Connector (siehe "myDatanet Connector " auf Seite 283) erforderlich.

8. Nachdem Sie die für die von Ihnen gewählte Möglichkeit erforderlichen Vorbereitungen getroffen haben, lösen Sie, falls Sie dies in ihrem PAWN-Script vorgesehen haben, direkt am Gerät eine Übertragung aus. Haben Sie keine Möglichkeit zum Auslösen einer Übertragung vorgesehen, warten Sie bis zur nächsten planmäßigen Datenübertragung.

9. Werten Sie die ankommenden Daten aus.

6872
6872
6872

# Kapitel 12 Benutzerschnittstelle am Cloud-Server

## 12.1 Messstellenkonfiguration

*Hinweis: Abhängig vom jeweiligen Benutzerlevel sind einige der in den folgenden Unterkapiteln erwähnten Konfigurationsfelder unter Umständen ausgeblendet.*

Die Eingabemaske zur Konfiguration der Messstelle erreichen Sie durch Klicken auf den Messstellennamen in der Messstellenliste (siehe "Messstellen" auf Seite 123).

### 12.1.1 Messstelle

**Kunde**
*gibt an, welchem Account die Messstelle zugeordnet ist*

**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!)*

### 12.1.2 Kommentar

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

### 12.1.3 Steuerung

*Hinweis: Wurde die Messstelle aus einer Applikations-Vorlage (siehe "Anlegen einer neuen Messstelle" auf Seite 125) erstellt, ist dieser Konfigurationsabschnitt sichtbar.*

| | | | |
|---|---|---|---|
| Script Typ | aus | Script deaktiviert | |
| | Pawn | Aktiviert die Scriptabarbeitung und teilt dem M120 Modul mit, dass es sich bei dem Script um ein „Pawn Script“ handelt. | |
| Script Quelle | Script | Script | Eingabefenster zum Editieren des Scripts, das in das M120 Modul geladen wird (siehe "Pawn Script" auf Seite 197) |
| | Hochladen eines kompilierten Scripts | Datei hochladen | Auswahl des Script Binary-Files (*.amx), das auf den Cloud-Server hochgeladen und bei der nächsten Verbindung in das M120 Modul 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 283) | | |

### 12.1.4 Konfiguration 0 - Konfiguration 1

Diese Konfigurationsabschnitte ermöglichen es die Parameter aus den vom Kunden frei definierbaren, voneinander unabhängigen Speicherblöcken über die Oberfläche des Cloud-Servers verändern bzw. anzeigen zu können. Dazu muss der logische Aufbau der Konfiguratiosdatenblöcke mit Hilfe des myDatanet Connector (siehe "myDatanet Connector " auf Seite 283) definiert werden.

### 12.1.5 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 KB eingegeben werden. |

### 12.1.6 Grundeinstellung

| | | |
|---|---|---|
| Zeitzone | Regionseinstellungen (nicht relevant für Rohmessdaten, da diese in UTC gespeichert werden) | |
| Sommerzeit | Konfiguration für automatische Zeitumstellung | |
| | standard | Die Konfiguration für die Zeitumstellung wird von der globalen Servereinstellung übernommen (siehe t.b.d.). |
| | 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 | |

### 12.1.7 FTP-Export Einstellungen

*Hinweis: Dieser Konfigurationsabschnitt ist nur sichtbar, wenn die Lizenz "FTP Agent Extended" für den Cloud-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 myDataweb-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] | |

## 12.2 Gerätekonfiguration

*Hinweis: Abhängig vom jeweiligen Benutzerlevel sind einige der in den folgenden Unterkapiteln erwähnten Konfigurationsfelder unter Umständen ausgeblendet.*

Die Eingabemaske zur Konfiguration des Geräts erreichen Sie durch Klicken auf die Seriennummer in der Messstellenliste (siehe "Messstellen" auf Seite 123).

### 12.2.1 Kommentar

**Kommentar**

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

### 12.2.2 Messgerät

| | |
|---|---|
| Kunde | Name des Kunden, dem das Messgerät zugeordnet ist |
| Seriennummer | Seriennummer des Geräts |
| Geräteklasse | Damit ein Gerät mit einer Messstelle verbunden werden kann, müssen die Geräteklasse der Messstelle und die des Geräts übereinstimmen. |
| 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 | | |
| 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). |
| Hardware Version | Hardwareversion des ins rapidM2M Developer Board eingesetzten M120 Moduls | |

# Kapitel 13 Pawn Script

## 13.1 Allgemein

Das folgende Kapitel beschreibt die Funktionalität des Pawn Scripts. PAWN (vormals SMALL) ist eine C-ähnliche Skriptsprache, welche auf embedded Systemen läuft. Weitere Informationen finden Sie unter http://en.wikipedia.org/wiki/Pawn_(programming_language).

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 M120 Modul zu erstellen:

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

### 13.1.1 Direkte Eingabe eines Pawn Scripts

Die Eingabe des Pawn Scripts erfolgt über den Konfigurationsabschnitt „Steuerung“ (siehe "Steuerung" auf Seite 193) der Eingabemaske zur Konfiguration der Messstelle. Als „Script Type“ muss „Pawn“ ausgewählt werden, damit das M120 Modul die unter „Script“ eingegebenen Befehle als Pawn Script interpretiert.

### 13.1.2 Hochladen eines Binary-Files

Wurde über die Listenauswahl "Script Quelle" im Konfigurationsabschnitt „Steuerung“ (siehe "Steuerung" auf Seite 193) 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 148) erstelltes Binary-File auf den Cloud–Server hochgeladen werden. Dieses wird dann bei der nächsten Verbindung in das M120 Modul geladen. Als „Script Type“ muss auch bei dieser Methode „Pawn“ ausgewählt werden, damit das M120 Modul die Befehle als Pawn Script interpretiert.

### 13.1.3 Script-Editor des rapidM2M Toolset

Das rapidM2M Toolset beinhaltet unter dem Kateireiter "Script" (siehe "Kateireiter "Script"" auf Seite 148) einen Editor zum Erstellen und Kompilieren der Pawn Scripts. Zum Funktionsumfang des rapidM2M Toolset gehört auch das Übertagen der kompilierten Scripts per USB-Verbindung in das M120 Modul.

## 13.2 rapidM2M API

### 13.2.1 Core Funktionen

**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*<br>• *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*<br>• *false, wenn das Argument oder der Index ungültig sind*<br><br>*Diese Funktion setzt ein Argument in einer variablen Argumentenliste. Wenn das Argument ein Array ist, gibt "index" den Index des gewünschten Arrayelements an.* |

**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.* |

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)* |

native min(value1, value2);

*liefert den kleineren der beiden übergebenen Wert*

| *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 Wert*

| *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*<br>• *"min", wenn der Wert kleiner "min" ist*<br>• *"max", wenn der Wert größer "max" ist* |

## 13.2.2 Consolen Funktionen

**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*<br>• *ERROR, falls nicht erfolgreich* |

## 13.2.3 rapidM2M Funktionen

### 13.2.3.1 Arrays mit symbolischen Indizes

#### TrM2M_Id

*Informationen zur Identifikation des Moduls*

```
// string     rapidM2M Modulidentifikation (z.B. "rapidM2M M1 HW1.1")
// module     rapidM2M Modultyp (z.B. "M1")
// 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}             ]
```

#### TrM2M_GSMPos

*Informationen über eine GSM-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_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 204)

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

### 13.2.3.2 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
```

#### 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
```

#### Einstellung der Signalrichtung der GPIOs

```
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"
```

### 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   = 0b00000001, // 1 Stopbit
RM2M_UART_2_STOPBIT   = 0b00000010, // 2 Stopbit
RM2M_UART_PARITY_NONE = 0b00000000, // keine Parität
RM2M_UART_PARITY_ODD  = 0b00000100, // ungerade Parität
RM2M_UART_PARITY_EVEN = 0b00001000, // gerade Parität
RM2M_UART_7_DATABIT   = 0b00000000, // 7 Datenbits
RM2M_UART_8_DATABIT   = 0b00010000, // 8 Datenbits
RM2M_UART_FLOW_NONE   = 0b00000000, // keine Flusskontrolle
RM2M_UART_FLOW_RTSCTS = 0b01000000, // RTS/CTS Handshake
```

### 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
```

### Verbindungsflags

*Steuerflags für die Funktion rM2M_TxStart()*

```
RM2M_TX_POSUPDATE  = 0b00000001, /* Update der GSM-Positionsdaten beim
                                    Verbindungsaufbau */
```

### Kommunikationsmodi

*Kommunikationsmodi für die Funktion rM2M_TxSetMode()*

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

### Verbindungsstatus

*Rückgabewerte der Funktion rM2M_TxGetStatus()*

```
RM2M_TX_FAILED     = 0b00000001, // Verbindungsaufbau fehlgeschlagen
RM2M_TX_ACTIVE     = 0b00000010, // GPRS-Verbindung besteht
RM2M_TX_STARTED    = 0b00000100, // Verbindungsaufbau gestartet
RM2M_TX_RETRY      = 0b00001000, // Wartezeit bis zum Retry
RM2M_TX_WAKEUPABLE = 0b00010000, // Modem ins GSM-Netz eingebucht
```

### 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
```

### Konfigurierungsflags für die Funktion rM2M_CfgInit()

```
RM2MCFG_VOLATILE   = 0b00000001, // flüchtige Speicherung (RAM)
```

### Indizes der Registrierungsspeicherblöcke

*auf die mittels der Funktionen "rM2M_RegGetString()", "rM2M_RegGetValue()", "rM2M_RegSetString()" und "rM2M_RegSetValue()" zugegriffen werden kann. Detaillierte Informationen zu den Registrierungsspeicherblöcken finden Sie im Kapitel "Registrierungsspeicherblöcke" auf Seite 63.*

```
//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)

//Anzahl der Registrierungsspeicherblöcke
RM2M_REG_NUM_REGS  = 4,
```

#### 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.
```

### 13.2.3.3 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* |

#### 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*<br>• *RM2M_GPIO_LOW für "low" am interruptfähingen Pin*<br>• *RM2M_GPIO_HIGH für "high" am interruptfähingen Pin* |

#### 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 o für den ersten Speicherblock, der geändert wurde* |

### 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 204), der geändert wurde* |

### 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* |

#### 13.2.3.4 Funktionen

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

*liefert die Informationen zur Identifikation des Moduls*

| *Parameter* | *Erklärung* |
|---|---|
| *id* | *Struktur zur Aufnahme der Informationen zur Identifikation des Moduls (siehe "TrM2M_Id" im Kapitel "Arrays mit symbolischen Indizes" auf Seite 202)* |
| *len* | *Größe (in Cells) der Struktur zur Aufnahme der Informationen - OPTIONAL* |

| | *Erklärung* |
|---|---|
| *Rückgabewert* | • *Verwendete Größe (in Cells) der Struktur zur Aufnahme der Informationen*<br>• *ERROR, wenn Adresse und/oder Länge der id-Struktur ungültig sind (außerhalb des Skript-Datenspeichers)*<br>• *‹ OK, wenn ein anderer Fehler auftritt (siehe "Returncodes für allgemeine Zwecke" im Kapitel "Konstanten" auf Seite 204)*<br><br>***Hinweis:*** *Die Firmware des Moduls 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 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*<br>*= 0: Es wird die aktuelle Systemzeit (in UTC) konvertiert.*<br>*› 0: Es wird der übergebene Zeitstempel konvertiert.*<br>*(Der Zeitstempel muss in Sekunden seit 31.12.1999 angegeben werden.)* |

| | *Erklärung* |
|---|---|
| *Rückgabewert* | • *timestamp = 0: Sekunden seit 31.12.1999 (aktuelle Systemzeit in UTC)*<br>• *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*<br>***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*<br>*= 0: Es wird das Datum für die aktuelle Systemzeit (in UTC) ermittelt.*<br>*› 0: Es wird das Datum für den übergebene Zeitstempel ermittelt.*<br>*(Der Zeitstempel muss in Sekunden seit 31.12.1999 angegeben werden.)* |

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

### native rM2M_GetTimezoneOffset();

*liefert die Differenz (in Sekunden) zwischen Systemzeit (UTC) und der für die Messstelle am Cloud-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 Cloud-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*<br><br>*Typ der Funktion: public func();* |

| | *Erklärung* |
|---|---|
| *Rückgabewert* | • *OK, wenn erfolgreich*<br>• *ERROR, wenn kein gültiger Index übergeben wurde, bei einem internen Fehler oder wenn keine weiteren Timer mehr angelegt werden können (maximale Anzahl erreicht)*<br>• *‹ OK, wenn ein anderer Fehler auftritt (siehe "Returncodes für allgemeine Zwecke" im Kapitel "Konstanten" auf Seite 204)* |

**native rM2M_TimerRemove(funcidx);**

*entfernt einen 1s Timer*

| *Parameter* | *Erklärung* |
|---|---|
| *funcidx* | *Index der öffentlichen Funktion des zu entfernenden Timers*<br><br>*Typ der Funktion: public func();* |

| | *Erklärung* |
|---|---|
| *Rückgabewert* | • *OK, wenn erfolgreich*<br>• *ERROR, wenn kein gültiger Index übergeben wurde oder bei einem internen Fehler*<br>• *‹ OK, wenn ein anderer Fehler auftritt (siehe "Returncodes für allgemeine Zwecke" im Kapitel "Konstanten" auf Seite 204)* |

**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*<br><br>*Typ der Funktion: public func();* |
| *cyclic* | *Einstellung für das Verhalten nach Ablauf des Timerintervalls:*<br><br>*true: Der Timer soll nach dem Ablauf des Intervalls neu gestartet werden.*<br>*false: Der Timer wird nach Ablauf des Intervalls gestoppt.* |
| *time* | *Timerintervall in Millisekunden* |

| | *Erklärung* |
|---|---|
| *Rückgabewert* | • *OK, wenn erfolgreich*<br>• *ERROR, wenn kein gültiger Index übergeben wurde, bei einem internen Fehler oder wenn keine weiteren Timer mehr angelegt werden können (maximale Anzahl erreicht)*<br>• *‹ OK, wenn ein anderer Fehler auftritt (siehe "Returncodes für allgemeine Zwecke" im Kapitel "Konstanten" auf Seite 204)* |

native rM2M_TimerRemoveExt(funcidx);

*entfernt einen ms Timer*

| *Parameter* | *Erklärung* |
|---|---|
| *funcidx* | *Index der öffentlichen Funktion des zu entfernenden Timers*<br><br>*Typ der Funktion: public func();* |

| | *Erklärung* |
|---|---|
| *Rückgabewert* | • *OK, wenn erfolgreich*<br>• *ERROR, wenn kein gültiger Index übergeben wurde oder bei einem internen Fehler*<br>• *‹ OK, wenn ein anderer Fehler auftritt (siehe "Returncodes für allgemeine Zwecke" im Kapitel "Konstanten" auf Seite 204)* |

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:*<br><br>*RM2M_GPIO_INPUT : Eingang*<br>*RM2M_GPIO_OUTPUT : Ausgang* |

| | *Erklärung* |
|---|---|
| *Rückgabewert* | • *OK, wenn erfolgreich*<br>• *‹ OK, wenn ein Fehler auftritt (siehe "Returncodes für allgemeine Zwecke" im Kapitel "Konstanten" auf Seite 204)* |

### 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* |
| *dir* | *Angabe des auszugebenden Levels:*<br>*RM2M_GPIO_LOW : Ausgangslevel auf "low"*<br>*RM2M_GPIO_HIGH : Ausgangslevel auf "high"* |

| | *Erklärung* |
|---|---|
| *Rückgabewert* | • *OK, wenn erfolgreich*<br>• *ERROR_BAD_STATE, wenn der betreffende GPIO nicht als Ausgang konfiguriert wurde*<br>• *< OK, wenn ein anderer Fehler auftritt (siehe "Returncodes für allgemeine Zwecke" im Kapitel "Konstanten" auf Seite 204)* |

### 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*<br>• *RM2M_GPIO_HIGH für "high" am Eingang*<br>• *ERROR_BAD_STATE, wenn der betreffende GPIO nicht als Eingang konfiguriert wurde*<br>• *< OK, wenn ein anderer Fehler auftritt (siehe "Returncodes für allgemeine Zwecke" im Kapitel "Konstanten" auf Seite 204)* |

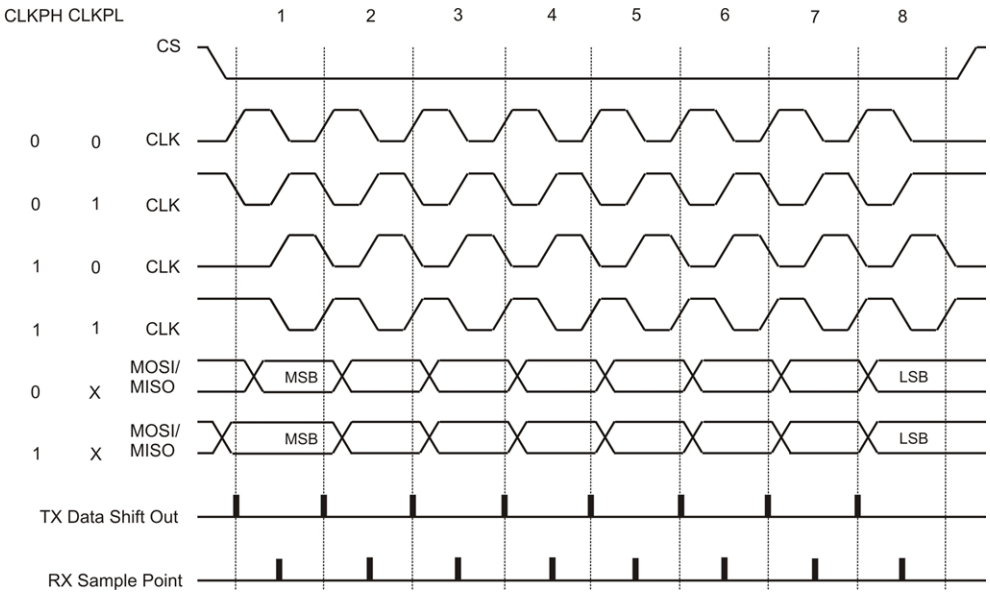
native rM2M_SpiInit(spi, clock, config);

*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 "SPI" auf Seite 111).* |
| *config* | *Konfiguration von Clock-Polarität und Clock-Phase*<br><br>*Bito: Clock-Polarität*<br>*0 = Idle "low"*<br>*1 = Idle "high"*<br><br>*Bit1: Clock-Phase*<br>*0 = Daten werden bei der ersten Flanke ausgegeben.*<br>*1 = Daten werden bei der ersten Flanke übernommen.*<br><br>***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 204). Die Konstanten lassen sich auch durch "oder"-Verknüpfung kombinieren.* |

| | *Erklärung* |
|---|---|
| *Rückgabewert* | • *OK, wenn erfolgreich*<br>• *ERROR, wenn die Nummer des SPI-Interfaces ungültig ist*<br>• *ERROR_ALREADY_SUBSCRIBED, wenn das SPI-Interface bereits initialisiert wurde*<br>• *‹ OK, wenn ein anderer Fehler auftritt (siehe "Returncodes für allgemeine Zwecke" im Kapitel "Konstanten" auf Seite 204)* |

***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*<br>• *ERROR, wenn die Nummer des SPI-Interfaces ungültig ist*<br>• *‹ OK, wenn ein anderer Fehler auftritt (siehe "Returncodes für allgemeine Zwecke" im Kapitel "Konstanten" auf Seite 204)* |

### 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*<br>• *ERROR, wenn einer der folgenden Fehler auftritt*<br>  ◦ *Anzahl der zu sendenden Bytes › 255*<br>  ◦ *Anzahl der zu empfangenden Bytes › 255*<br>• *‹ OK, wenn ein anderer Fehler auftritt (siehe "Returncodes für allgemeine Zwecke" im Kapitel "Konstanten" auf Seite 204)* |

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 "I2C" auf Seite 112).* |
| *config* | *reserviert für Erweiterungen* |

| | *Erklärung* |
|---|---|
| *Rückgabewert* | • *OK, wenn erfolgreich*<br>• *ERROR, wenn die Nummer des I²C-Interfaces ungültig ist*<br>• *ERROR_ALREADY_SUBSCRIBED, wenn das I²C-Interface bereits initialisiert wurde*<br>• *‹ OK, wenn ein anderer Fehler auftritt (siehe "Returncodes für allgemeine Zwecke" im Kapitel "Konstanten" auf Seite 204)* |

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*<br>• *ERROR, wenn die Nummer des I²C-Interfaces ungültig ist*<br>• *‹ OK, wenn ein anderer Fehler auftritt (siehe "Returncodes für allgemeine Zwecke" im Kapitel "Konstanten" auf Seite 204)* |

**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*<br>• *ERROR, wenn einer der folgenden Fehler auftritt*<br>  ◦ *Anzahl der zu sendenden Bytes › 255*<br>  ◦ *Anzahl der zu empfangenden Bytes › 255*<br>• *‹ OK, wenn ein anderer Fehler auftritt (siehe "Returncodes für allgemeine Zwecke" im Kapitel "Konstanten" auf Seite 204)* |

**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 "UART" auf Seite 110).* |
| *mode* | *Bit 0...1*<br>*1 = 1 Stopbit*<br>*2 = 2 Stopbit*<br><br>*Bit 2...3*<br>*0 = keine Parität*<br>*1 = ungerade Parität*<br>*2 = gerade Parität*<br><br>*Bit 4...5*<br>*0 = 7 Datenbits*<br>*1 = 8 Datenbits*<br><br>*Bit 6...7*<br>*0 = keine Flusskontrolle*<br>*1 = RTS/CTS Handshake*<br><br>***Hinweis:** Sie können für diesen Parameter auch die vordefinierten Konstanten verwenden (siehe "Konfiguration des UART-Interface" im Kapitel "Konstanten" auf Seite 204). Die Konstanten lassen sich auch durch "oder"-Verknüpfung kombinieren.* |
| *funcidx* | *Index der öffentlichen Funktion für den UART-Zeichenempfang*<br><br>*Typ der Funktion: public func(const data{}, len);* |

| | *Erklärung* |
|---|---|
| *Rückgabewert* | • *OK, wenn erfolgreich*<br>• *‹ OK, wenn ein Fehler auftritt (siehe "Returncodes für allgemeine Zwecke" im Kapitel "Konstanten" auf Seite 204)* |

**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*<br>• *< OK, wenn ein Fehler auftritt (siehe "Returncodes für allgemeine Zwecke" im Kapitel "Konstanten" auf Seite 204)* |

**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* | • *OK, wenn erfolgreich*<br>• *ERROR, wenn die Anzahl der zu sendenden Bytes > 256*<br>• *< OK, wenn ein anderer Fehler auftritt (siehe "Returncodes für allgemeine Zwecke" im Kapitel "Konstanten" auf Seite 204)* |

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*<br>*RM2M_IRQ_RISING : bei steigender Flanke*<br>*RM2M_IRQ_FALLING : bei fallender Flanke* |
| *funcidx* | *Index der öffentlichen Funktion, die im Falle eines Interrupts ausgeführt werden soll*<br>*Typ der Funktion: public func(pinstate);* |

| | *Erklärung* |
|---|---|
| *Rückgabewert* | • *OK, wenn erfolgreich*<br>• *ERROR_ALREADY_SUBSCRIBED, wenn die Interrupt-Funktionalität für den betreffenden Pin bereits initialisiert wurde*<br>• *ERROR, wenn einer der folgenden Fehler auftritt*<br>  • *Interrupt-Funktionalität für diesen Pin nicht verfügbar*<br>  • *Die im Fall eines Interrupts auszuführende Funktion ist nicht öffentlich.*<br>• *‹ OK, wenn ein anderer Fehler auftritt (siehe "Returncodes für allgemeine Zwecke" im Kapitel "Konstanten" auf Seite 204)* |

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*<br>• *‹ OK, wenn ein Fehler auftritt (siehe "Returncodes für allgemeine Zwecke" im Kapitel "Konstanten" auf Seite 204)* |

**native rM2M_TxStart(flags=0);**

*löst eine Verbindung zum Server aus*

| *Parameter* | *Erklärung* |
|---|---|
| *flags* | *Wenn Bito (RM2M_TX_POSUPDATE) gesetzt ist, erfolgt auch ein Update der GSM-Positionsdaten (siehe t.b.d.).* |

| | *Erklärung* |
|---|---|
| *Rückgabewert* | • *OK, wenn erfolgreich*<br>• *ERROR_SIM_STATE, wenn aufgrund des aktuellen SIM-Status eine Verbindung nicht möglich ist (siehe "SIM-Status" im Kapitel "Konstanten" auf Seite 204)*<br>• *‹ OK, wenn ein anderer Fehler auftritt (siehe "Returncodes für allgemeine Zwecke" im Kapitel "Konstanten" auf Seite 204)* |

native rM2M_TxSetMode(mode);

*setzt die zu verwendende Verbindungsart*

| *Parameter* | *Erklärung* |
|---|---|
| *mode* | *zu verwendende Verbindungsart:*<br><br>*RM2M_TXMODE_TRIG: Die Verbindung erfolgt beim Aufruf der Funktion "rM2M_TxStart()"*<br><br>*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 "Messstellen" auf Seite 123). Dazu bucht sich das Gerät unverzüglich ins GSM-Netz ein sobald dieser Modus gesetzt wurde .*<br><br><br><br>*RM2M_TXMODE_ONLINE: Das Gerät trennt die Verbindung nicht und übermittelt kontinuierlich die Messdaten. Der Verbindungsaufbau erfolgt unverzüglich sobald dieser Modus gesetzt wurde. Ein Aufruf der Funktion "rM2M_TxStart()" ist nicht erforderlich.* |

| | *Erklärung* |
|---|---|
| *Rückgabewert* | • *OK, wenn erfolgreich*<br>• *ERROR_SIM_STATE, wenn aufgrund des aktuellen SIM-Status der Modus nicht möglich ist (siehe "SIM-Status" im Kapitel "Konstanten" auf Seite 204)*<br>• *‹ OK, wenn ein anderer Fehler auftritt (siehe "Returncodes für allgemeine Zwecke" im Kapitel "Konstanten" auf Seite 204)* |

***Hinweis:*** *Ergänzende Erklärung zu den Verbindungsarten*

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

native rM2M_TxGetStatus();

*liefert den aktuellen Verbindungsstatus*

| | *Erklärung* | |
|---|---|---|
| *Rückgabewert* | *Bito (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)* |

native rM2M_GSMGetRSSI();

*liefert die GSM-Signalstärke*

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

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*<br><br>*= 0: Die aktuelle Systemzeit wird als Zeitstempel verwendet.*<br>*› 0: Der übergebene Zeitstempel wird verwendet.*<br>*(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*<br>• *-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.*<br>• *ERROR, wenn einer der folgenden Fehler auftritt*<br>  • *Speicherbereich (data{}, len) ist ungültig.*<br>  • *Mehr als 10 Aufrufe in einem Skriptdurchlauf*<br>  • *Anzahl der zu speichernden Bytes › 1024 Byte*<br>  • *FLASH-Schreibvorgang nicht erfolgreich*<br>  • *Übergabeparameter timestamp liegt mehr als 5 Minuten in der Zukunft*<br>• *‹ OK, wenn ein anderer Fehler auftritt (siehe "Returncodes für allgemeine Zwecke" im Kapitel "Konstanten" auf Seite 204)* |

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

*schreibt den übergebenen Wert 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 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:*<br><br>*true: "Big Endian" wird verwendet*<br>*false: "Little Endian" wird verwendet* |

| | *Erklärung* |
|---|---|
| *Rückgabewert* | • *OK, wenn erfolgreich*<br>• *‹ OK, wenn ein Fehler auftritt (siehe "Returncodes für allgemeine Zwecke" im Kapitel "Konstanten" auf Seite 204)* |

***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* |

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*

| *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:*<br><br>*true: Die Daten sind im "Big Endian"-Format im Array gespeichert.*<br>*false: Die Daten sind im "Little Endian"-Format im Array gespeichert.* |

| | *Erklärung* |
|---|---|
| *Rückgabewert* | • *OK, wenn erfolgreich*<br>• *‹ OK, wenn ein Fehler auftritt (siehe "Returncodes für allgemeine Zwecke" im Kapitel "Konstanten" auf Seite 204)* |

***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* |

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

*Funktion für den Zugriff auf gepackte Daten. Wurde das Bito (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*<br><br>*Set Packed: Array, in das der Wert geschrieben werden soll*<br>*Get Packed: Array, aus dem der Wert gelesen werden soll* |
| *pos* | *Byteoffset innerhalb des Arrays*<br><br>*Set Packed: Position, an die der Wert geschrieben werden soll*<br>*Get Packed: Position, von der der Wert gelesen werden soll* |
| *value* | *Set Packed: Wert, der in das Array geschrieben werden soll*<br>*Get Packed: Wert, der aus dem Array gelesen werden soll* |
| *type* | *Konfigurierungsflags für die Funktion*<br><br>*Bito: Auswahl Set Packed / Get Packed*<br>*0 = Wert soll geschrieben werden*<br>*1 = Wert soll gelesen werden*<br><br>*Bit1: Byte-Reihenfolge*<br>*0 = "Little Endian"-Format*<br>*1 = "Big Endian"-Format*<br><br>*Bit2...3*<br>*reserviert für Erweiterungen*<br><br>*Bit4...7: Datentyp*<br>*1 = 8-Bit Unsigned*<br>*2 = 16-Bit Unsigned*<br>*4 = 32-Bit Unsigned / 32-Bit Float*<br>*9 = 8-Bit Signed*<br>*10 = 16-Bit Signed*<br>*12 = 32-Bit Signed*<br><br>***Hinweis:*** *Sie können für diesen Parameter auch die vordefinierten Konstanten verwenden (siehe "Konfigurierungsflags für die Funktion rM2M_Pack()" im Kapitel "Konstanten" auf Seite 204). Die Konstanten lassen sich auch durch "oder"-Verknüpfung kombinieren.* |

| | *Erklärung* |
|---|---|
| *Rückgabewert* | • *OK, wenn erfolgreich*<br>• *‹ OK, wenn ein Fehler auftritt (siehe "Returncodes für allgemeine Zwecke" im Kapitel "Konstanten" auf Seite 204)* |

**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*<br><br>*Bito: Art der Speicherung*<br>*0 (default) = nichtflüchtig im FLASH gespeichert*<br>*RM2MCFG_VOLATILE = flüchtig im RAM gespeichert* |

| | *Erklärung* |
|---|---|
| *Rückgabewert* | • *OK, wenn erfolgreich*<br>• *< OK, wenn ein Fehler auftritt (siehe "Returncodes für allgemeine Zwecke" im Kapitel "Konstanten" auf Seite 204)* |

***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 Bito gesetzt wurde (Bito = RM2MCFG_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 (Bito = RM2MCFG_VOLATILE) oder nichtflüchtig im FLASH (Bito = 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 Cloud-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*<br>• *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.)*<br>• *‹ OK, wenn ein anderer Fehler auftritt (siehe "Returncodes für allgemeine Zwecke" im Kapitel "Konstanten" auf Seite 204)* |

### 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 (Bito = RM2MCFG_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*<br>• *‹ OK, wenn ein Fehler auftritt (siehe "Returncodes für allgemeine Zwecke" im Kapitel "Konstanten" auf Seite 204)* |

### 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*<br>• *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.)*<br>• *‹ OK, wenn ein anderer Fehler auftritt (siehe "Returncodes für allgemeine Zwecke" im Kapitel "Konstanten" auf Seite 204)* |

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*<br>• *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.)*<br>• *‹ OK, wenn ein anderer Fehler auftritt (siehe "Returncodes für allgemeine Zwecke" im Kapitel "Konstanten" auf Seite 204)* |

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*<br><br>*Typ der Funktion: public func(cfg);* |

| | *Erklärung* |
|---|---|
| *Rückgabewert* | • *OK, wenn erfolgreich*<br>• *ERROR, wenn kein gültiger Index einer öffentlichen Funktion übergeben wurde*<br>• *‹ OK, wenn ein anderer Fehler auftritt (siehe "Returncodes für allgemeine Zwecke" im Kapitel "Konstanten" auf Seite 204)* |

**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 63.*

| *Parameter* | *Erklärung* |
|---|---|
| *reg* | *Index des Registrierungsspeicherblocks (siehe "Indizes der Registrierungsspeicherblöcke" im Kapitel "Konstanten" auf Seite 204)* |
| *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*<br>• *‹ OK, wenn ein Fehler auftritt (siehe "Returncodes für allgemeine Zwecke" im Kapitel "Konstanten" auf Seite 204)* |

**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 63.*

| *Parameter* | *Erklärung* |
|---|---|
| *reg* | *Index des Registrierungsspeicherblocks (siehe "Indizes der Registrierungsspeicherblöcke" im Kapitel "Konstanten" auf Seite 204)* |
| *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*<br>• *‹ OK, wenn ein Fehler auftritt (siehe "Returncodes für allgemeine Zwecke" im Kapitel "Konstanten" auf Seite 204)* |

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 63.*

| *Parameter* | *Erklärung* |
|---|---|
| *reg* | *Index des Registrierungsspeicherblocks (siehe "Indizes der Registrierungsspeicherblöcke" im Kapitel "Konstanten" auf Seite 204)* |
| *name* | *Name des Eintrags*<br><br>*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*<br>• *‹ OK, wenn ein Fehler auftritt (siehe "Returncodes für allgemeine Zwecke" im Kapitel "Konstanten" auf Seite 204)* |

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 63.*

| *Parameter* | *Erklärung* |
|---|---|
| *reg* | *Index des Registrierungsspeicherblocks (siehe "Indizes der Registrierungsspeicherblöcke" im Kapitel "Konstanten" auf Seite 204)* |
| *name* | *Name des Eintrags*<br><br>*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*<br>• *‹ OK, wenn ein Fehler auftritt (siehe "Returncodes für allgemeine Zwecke" im Kapitel "Konstanten" auf Seite 204)* |

**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 63.*

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

| | *Erklärung* |
|---|---|
| *Rückgabewert* | • *OK, wenn erfolgreich*<br>• *‹ OK, wenn ein Fehler auftritt (siehe "Returncodes für allgemeine Zwecke" im Kapitel "Konstanten" auf Seite 204)* |

**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*<br><br>*Typ der Funktion: public func(const SmsTel[], const SmsText[]);* |
| *config* | *reserviert für Erweiterungen* |

| | *Erklärung* |
|---|---|
| *Rückgabewert* | • *OK, wenn erfolgreich*<br>• *‹ OK, wenn ein Fehler auftritt (siehe "Returncodes für allgemeine Zwecke" im Kapitel "Konstanten" auf Seite 204)* |

**native rM2M_SmsClose();**

*deaktiviert den SMS-Empfang*

| | *Erklärung* |
|---|---|
| *Rückgabewert* | *OK* |

native rM2M_WriteLog(log, param);

*erzeugt einen Eintrag im Gerätelog*

| *Parameter* | *Erklärung* |
|---|---|
| *log* | *zu erzeugender Log-Eintrag (gültiger Bereich: 0...999).*<br><br>*Die Log-Einträge werden auf den gültigen Bereich eingeschränkt. D.h. bei Unterschreiten des gültigen Bereichs wird 0 verwendet, bei Überschreiten des gültigen Bereichs wird 999 verwendet.*<br><br>***Wichtiger Hinweis:** Die Log-Einträge werden für die Anzeige am Server in den Bereich "MODULE ERR" (2000-2999) gemappt.* |
| *param* | *zusätzlicher Parameter zur genaueren Spezifizierung des Log-Eintrags (gültiger Bereich: -32768 ... 32767)* |

| | *Erklärung* |
|---|---|
| *Rückgabewert* | *OK* |

**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 Cloud-Server übertragen und können z.B. per XML-Schnittstelle (siehe "rm2mpos" auf Seite 167) ausgelesen werden.*

| *Parameter* | *Erklärung* |
|---|---|
| *Lat* | *geographische Breite in Grad (Auflösung: 0,000001°)*<br>*-90 000 000 = Südpol 90° Süd*<br>*0 = Äquator*<br>*+90 000 000 = Nordpol 90° Nord* |
| *Long* | *geographische Länge in Grad (Auflösung: 0,000001°)*<br>*-180 000 000 = 180° West*<br>*0 = Nullmeridian (Greenwich)*<br>*+180 000 000 = 180° Ost* |
| *Elev* | *Höhe über dem Meeresspiegel in Meter (gültiger Bereich: -999...+9999)* |
| *Qual* | *Qualitätskennung (GPS quality indicator)*<br>*0 = Invalid/no fix*<br>*1 = Non-differential GPS fix*<br>*2 = Differential GPS fix*<br>*3 = Precise Positioning Service PPS*<br>*4 = Real Time Kinematic RTK*<br>*5 = Float RTK*<br>*6 = Estimated fix (dead reckoning, Koppelnavigation)*<br>*7 = Manual input mode*<br>*8 = Simulation mode* |
| *SatUsed* | *Anzahl der zur Positionsbestimmung verwendeten Satelliten (gültiger Bereich: 0 ... 99)* |

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

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 Cloud-Server übertragen und können z.B. per XML-Schnittstelle (siehe "rm2mpos" auf Seite 167) ausgelesen werden.*

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

| | *Erklärung* |
|---|---|
| *Rückgabewert* | • *OK, wenn erfolgreich*<br>• *‹ OK, wenn ein Fehler auftritt (siehe "NMEA Fehlercodes" im Kapitel "Konstanten" auf Seite 204)* |

**native rM2M_GetPos(&Lat, &Long, &Elev);**

*liest die im Gerät gespeicherten GPS-Positionsinformationen aus*

| *Parameter* | *Erklärung* |
|---|---|
| *Lat* | *Variable zur Aufnahme der geographischen Breite in Grad (Auflösung: 0,000001°)*<br>*-90 000 000 = Südpol 90° Süd*<br>*0 = Äquator*<br>*+90 000 000 = Nordpol 90° Nord* |
| *Long* | *Variable zur Aufnahme der geographischen Länge in Grad (Auflösung: 0,000001°)*<br>*-180 000 000 = 180° West*<br>*0 = Nullmeridian (Greenwich)*<br>*+180 000 000 = 180° Ost* |
| *Elev* | *Variable zur Aufnahme der Höhe über dem Meeresspiegel in Meter (gültiger Bereich: -999...+9999)* |

| | *Erklärung* |
|---|---|
| *Rückgabewert* | • *OK, wenn gültige GPS-Positionsinformationen im Gerät gespeichert sind*<br>• *‹ OK, wenn ein Fehler auftritt (siehe "Returncodes für allgemeine Zwecke" im Kapitel "Konstanten" auf Seite 204)* |

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

*liefert die Anzahl der GSM-Zellen, für die gültige Informationen im Gerät gespeichert sind (posidx ‹ 0 ) bzw. liest die im Gerät gespeicherten Informationen über eine GSM-Zelle im Empfangsbereich aus (posidx ›= 0)*

| *Parameter* | *Erklärung* |
|---|---|
| *posidx* | *Auswahl der von der Funktion gelieferten Information*<br>*posidx ‹ 0:* *Anzahl der GSM-Zellen für die gültige Informationen im Gerät gespeichert sind auslesen*<br>*posidx ›=0:* *Nummer des GSM-Zellen-Informationsblock, der ausgelesen werden soll* |
| *pos* | *posidx ‹ 0:* *nicht erforderlich*<br>*posidx ›=0:* *Struktur zur Aufnahme der Informationen über eine GSM-Zelle im Empfangsbereich (siehe "TrM2M_GSMPos" im Kapitel "Arrays mit symbolischen Indizes" auf Seite 202)* |

| | *Erklärung* |
|---|---|
| *Rückgabewert* | *posidx ‹ 0:* *Anzahl der GSM-Zellen, für die gültige Informationen im Gerät gespeichert sind (max. 10)*<br>*posidx ›=0:*<br>• *OK, wenn der gewünschte GSM-Zellen-Informationsblock gültige Daten enthält*<br>• *ERROR* |

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 202)* |
| *len* | *Größe (in Cells) der Struktur zur Aufnahme der Informationen - OPTIONAL* |

| | *Erklärung* |
|---|---|
| *Rückgabewert* | • *Verwendete Größe (in Cells) der Struktur zur Aufnahme der Informationen*<br>• *ERROR, wenn Adresse und/oder Länge der Info-Struktur ungültig sind (außerhalb des Skript-Datenspeichers)*<br>• *‹ OK, wenn ein anderer Fehler auftritt (siehe "Returncodes für allgemeine Zwecke" im Kapitel "Konstanten" auf Seite 204)* |

## 13.2.4 File Transfer Funktionen

### 13.2.4.1 Arrays mit symbolischen Indizes

#### TFT_Info

*Eigenschaften eines Dateieintrags*

```
// name       Name der Datei
// stamp      Zeitstempel der Datei (Sekunden seit 31.12.1999)
// size       Dateigröße in Byte
// crc        Ethernet CRC32 der Datei (siehe "CRC32()" )
// flags      Datei Flags (siehe "Datei Flags" im
//            Kapitel "Konstanten" auf Seite 241)
#define TFT_Info[  .name{256}, .stamp, .size, .crc, .flags ]
```

### 13.2.4.2 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.
```

### 13.2.4.3 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 241) 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:<ul><li>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></li><li>FT_CMD_WRITE: Array, das die Daten, die vom Cloud-Server erhalten wurden, enthält.</li></ul></td></tr>
<tr><td>len</td><td>Dieser Parameter ist nur beim Empfang der folgenden File Transfer Kommandos relevant:<ul><li>FT_CMD_READ: Anzahl der vom Cloud-Server angeforderten Bytes</li><li>FT_CMD_STORE: Größe des vom Cloud-Server erhaltenen Dateieigenschaftenblocks</li><li>FT_CMD_WRITE: Anzahl der vom Cloud-Server erhaltenen Bytes</li></ul></td></tr>
<tr><td>ofs</td><td>Dieser Parameter ist nur beim Empfang der folgenden File Transfer Kommandos relevant:<ul><li>FT_CMD_READ: Byteoffset innerhalb der Datei des an den Cloud-Server zu übertragenden Datenblocks</li><li>FT_CMD_WRITE: Byteoffset innerhalb der Datei des vom Cloud-Server erhaltenen Datenblocks</li></ul></td></tr>
</table>

#### 13.2.4.4 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*<br>*Typ der Funktion: public func(id, cmd, const data{}, len, ofs);*<br>***Wichtiger Hinweis:** Alle File Transfer Kommandos (siehe "File Transfer Kommandos" im Kapitel "Konstanten" auf Seite 241) müssen von dieser öffentlichen Funktion behandelt werden.* |

| | *Erklärung* |
|---|---|
| *Rückgabewert* | • *OK, wenn erfolgreich*<br>• *‹ OK, wenn ein Fehler auftritt (siehe "Returncodes für allgemeine Zwecke" im Kapitel "Konstanten" auf Seite 204)* |

**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*<br>• *‹ OK, wenn ein Fehler auftritt (siehe "Returncodes für allgemeine Zwecke" im Kapitel "Konstanten" auf Seite 204)* |

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.*

| *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 (siehe "CRC32()" )* |
| *flags* | *Datei Flags (siehe "Datei Flags" im Kapitel "Konstanten" auf Seite 241)* |

| | *Erklärung* |
|---|---|
| *Rückgabewert* | • *OK, wenn erfolgreich*<br>• *‹ OK, wenn ein Fehler auftritt (siehe "Returncodes für allgemeine Zwecke" im Kapitel "Konstanten" auf Seite 204)* |

native FT_Read(id, const data{}, len);

*übergibt die Daten an das System, um sie zum Cloud-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 Cloud-Server übergeben werden sollen* |
| *len* | *Anzahl der zu übergebenden Bytes* |

| | *Erklärung* |
|---|---|
| *Rückgabewert* | • *OK, wenn erfolgreich*<br>• *‹ OK, wenn ein Fehler auftritt (siehe "Returncodes für allgemeine Zwecke" im Kapitel "Konstanten" auf Seite 204)* |

native FT_Accept(id, newid=-1);

*akzeptiert die Datei, die der Cloud-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 242) 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*<br>• *‹ OK, wenn ein Fehler auftritt (siehe "Returncodes für allgemeine Zwecke" im Kapitel "Konstanten" auf Seite 204)* |

native FT_Written(id, len);

*bestätigt das schreiben der Daten, die vom Cloud-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 242) 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*<br>• *‹ OK, wenn ein Fehler auftritt (siehe "Returncodes für allgemeine Zwecke" im Kapitel "Konstanten" auf Seite 204)* |

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*<br>• *‹ OK, wenn ein Fehler auftritt (siehe "Returncodes für allgemeine Zwecke" im Kapitel "Konstanten" auf Seite 204)* |

## 13.2.5 Mathematische Funktionen

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* |

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.*

### 13.2.6 String Funktionen

> ***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 '\o')*

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

| | *Erklärung* |
|---|---|
| *Rückgabewert* | *Anzahl der Zeichen ohne der abschließenden '\o'* |

**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*<br>• *Anzahl der Zeichen, die geschrieben worden wäre, wenn das Array dest lang genug gewesen wäre (ohne '\o').*<br><br>*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 '\o').*

| *Parameter* | *Erklärung* |
|---|---|
| *dest* | *Array zur Aufnahme der zu kopierenden Zeichenkette* |
| *source* | *zu kopierende Zeichenkette* |
| *maxlength* | *Anzahl der zu kopierenden Zeichen - 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 '\o')*

| *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* | *Anzahl der anzufügenden Zeichen - 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*<br>• *0: die beiden Zeichenketten sind gleich (zumindest die berücksichtigte Länge)*<br>• *-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*<br>• *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*<br>• *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*<br>• *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*<br>• *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*<br>• *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*<br>• *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*<br><br>*2-36: Die angegebene Basis wird verwendet*<br>*0: Als Basis wird 8, 10 oder 16 verwendet, abhängig von der umzuwandelnden Zeichenkette*<br>*Basis 8: bei einer führenden 0*<br>*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* |

## 13.2.7 Hilfsfunktionen

### 13.2.7.1 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]
```

## 13.2.7.2 Konstanten

### Fehlercodes der Funktionen "CalcTable" und "CalcTableF"

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

## 13.2.7.3 Funktionen

### native getapilevel();

*gibt das implementierte API-Level der Skript-Engine aus*

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

### native CRC16(data{}, len);

*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* |

| | *Erklärung* |
|---|---|
| *Rückgabewert* | *berechnete CRC16* |

### native CRC32(data{}, len);

*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* |

| | *Erklärung* |
|---|---|
| *Rückgabewert* | *berechnete CRC32* |

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

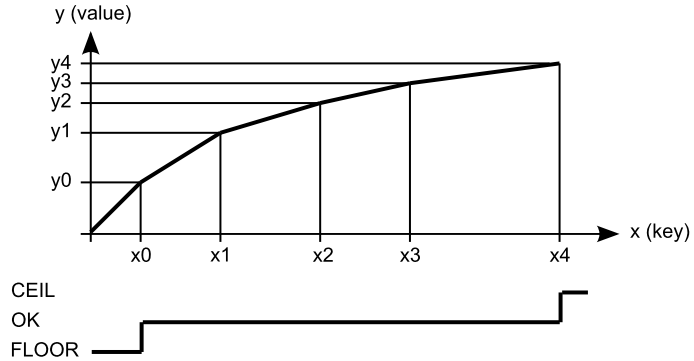
*sucht einen bestimmten Wert in der "key"-Spalte der übergebenen Stützpunktabelle und liefert den entsprechenden Wert der "value"-Spalte der Tabelle. Liegt der gesuchte Wert zwischen zwei Stützpunkten, wird der Rückgabewert zwischen den zwei angrenzenden "value"-Spaltenwerten linear interpoliert (Geradengleichung: 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*<br>• *TAB_ERR_FLOOR, wenn der gesuchte Wert kleiner als der erste Tabelleneintrag ist. "value" beinhaltet den ersten Tabelleneintrag.*<br>• *TAB_ERR_CEIL, wenn der gesuchte Wert größer als der letzte Tabelleneintrag ist. "value" beinhaltet den letzten Tabelleneintrag.*<br>• *‹ OK, wenn ein anderer Fehler auftritt (siehe "Returncodes für allgemeine Zwecke" im Kapitel "Konstanten" auf Seite 204)* |

***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.*

## 13.3 Pawn Script Fehlercodes

Sollte beim Durchlaufen des Pawn Scripts ein Fehler auftreten, wird die Scriptausführung gestoppt und deaktiviert. Zudem wird in das Gerätelog der entsprechende Fehlercode eingetragen. 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-Bito des 32-Bit Instruction Pointers. Eine Anleitung zum Auswerten des Gerätelogs finden Sie im Kapitel "Kateireiter "Log""( siehe "Karteireiter "Log"" auf Seite 154).

| 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.)<br><br>Das Script wird deaktiviert und das Errorhandling aktiviert (siehe "Errorhandling" auf Seite 62). 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.<br><br>Sollte innerhalb von 24h ein weiterer Laufzeitfehler auftreten, wird das Script deaktiviert und das Errorhandling aktiviert (siehe "Errorhandling" auf Seite 62). |
| | | 6 | SCRIPT UPDATE ERROR | Verbindungsabbruch während des Scriptdownloads<br><br>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. |

| Log-Eintrag | | Parameter | | Beschreibung |
|---|---|---|---|---|
| Code | Klartext | Code | Klartext | |
| 3001 | AMX_ERR_EXIT | ## | --- | Abbruch<br>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 |
| 3005 | AMX_ERR_ MEMACCESS | ## | --- | ungültiger Speicherzugriff<br>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 |

| Log-Eintrag | | Parameter | | Beschreibung |
|---|---|---|---|---|
| Code | Klartext | Code | Klartext | |
| 3022 | AMX_ERR_INIT | ## | --- | AMX nicht initialisiert (oder doppelt initialisiert) |
| 3023 | AMX_ERR_ USERDATA | ## | --- | Benutzer-Datenfeld kann nicht gesetzt werden (Tabelle voll) |
| 3024 | AMX_ERR_INIT_JIT | ## | --- | JIT kann nicht initialisiert werden. |
| 3025 | AMX_ERR_PARAMS | ## | --- | fehlerhafter Parameter |
| 3026 | AMX_ERR_DOMAIN | ## | --- | Domain-Fehler. Das Ergebnis des Ausdrucks ist nicht im gültigen Bereich. |
| 3027 | AMX_ERR_GENERAL | ## | --- | allgemeiner Fehler (unbekannter oder nicht spezifizierter Fehler) |
| 3028 | AMX_ERR_OVERLAY | ## | --- | Overlays werden nicht unterstützt (JIT) oder sind nicht initialisiert. |

# 13.4 Syntax

## 13.4.1 Allgemeine Syntax

### 13.4.1.1 Format

Bezeichner, Zahlen und Zeichen werden durch Leerzeichen, Tabulatoren, Zeilenumbrüche und "Form Feed" getrennt. Eine Serie von einer oder mehreren dieser Separatoren wird als Leerraum erkannt.

### 13.4.1.2 Optionale Semikolons

Semikolons (um ein Statement zu beenden) sind optional, wenn sie am Ende einer Zeile auftreten. Semikolons sind notwendig, um mehrere Statements in einer Zeile zu trennen. Ein Ausdruck kann auf mehrere Zeilen aufgeteilt werden, jedoch müssen Postfix-Operatoren in derselben Zeile wie der Operand stehen.

### 13.4.1.3 Kommentare

Text zwischen den Symbolen /* und */ (beide Symbole können auf derselben oder auf unterschiedlichen Zeilen stehen) und Text nach // (bis zum Ende einer Zeile) sind Kommentare. Kommentare dürfen nicht verschachtelt werden. Der Compiler betrachtet Kommentare als Leerzeichen. Ein Kommentar, der mit "/** " (zwei Sterne und ein Leerzeichen nach dem zweiten Stern) beginnt und mit einem "*/" endet, ist ein Dokumentationskommentar. Ein Kommentar, der mit "/// " (drei Schrägstriche und ein Leerzeichen nach dem dritten Schrägstrich) beginnt, ist ebenfalls ein Dokumentationskommentar. Der Parser kann den Dokumentationskommentar in unterschiedlicher Weise unterstützen, zum Beispiel könnte er eine Online-Hilfe daraus generieren.

#### 13.4.1.4 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.

#### 13.4.1.5 Reservierte Schlüsselworte

| Statements | Operator | Direktiven | Andere |
|---|---|---|---|
| assert | defined | defined | defined |
| break | sizeof | sizeof | sizeof |
| case | state | state | state |
| continue | tagof | tagof | tagof |
| default | | | |
| do | | | |
| else | | | |
| exit | | | |
| for | | | |
| goto | | | |
| if | | | |
| return | | | |
| sleep | | | |
| state | | | |
| switch | | | |
| while | | | |

#### 13.4.1.6 Numerische Konstanten

##### 13.4.1.6.1 Numerische Integer-Konstanten

**Binär**
*ob gefolgt von einer Serie von 0 und 1*

**Dezimal:**
*eine Serie von Ziffern zwischen 0 und 9*

**Hexadezimal**
*ox gefolgt von einer Serie von Ziffern zwischen 0 und 9 und den Buchstaben a bis f*

#### 13.4.1.6.2 Numerische Gleitkomma-Konstanten

Eine Gleitkommazahl ist eine Zahl mit einem Nachkommateil. Eine Gleitkommazahl beginnt mit einer oder mehreren Ziffern, beinhaltet einen Dezimaltrennpunkt und hat zumindest eine Ziffer nach dem Dezimaltrennpunkt. z.B. "12.0" und "0.75" sind gültige Gleitkommazahlen. Optional kann noch ein Exponent angehängt werden. Die Notation ist der Buchstabe "e" (Kleinbuchstabe), gefolgt von einer ganzzahligen numerischen Konstante. Z.B. "3.12e4" oder "12.3e-3" sind gültige Gleitkommazahlen mit Exponent.

### 13.4.2 Variablen

#### 13.4.2.1 Deklaration

Das Schlüsselwort "new" deklariert eine neue Variable. Für spezielle Deklarationen wird das Schlüsselwort "new" durch "static" ersetzt (siehe "Statische lokale Deklaration" auf Seite 260). 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 272) - lokale Variable

Beispiel:

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

#### 13.4.2.2 Lokale Deklaration

Eine lokale Deklaration erscheint innerhalb eines Anweisungs-Blocks. Auf eine Variable kann nur innerhalb dieses Blocks und der darin enthaltenen Blöcke zugegriffen werden. Eine Deklaration innerhalb des ersten Ausdrucks einer Schleifenanweisung ist ebenfalls eine lokale Deklaration.

#### 13.4.2.3 Globale Deklaration

Eine globale Deklaration erscheint außerhalb einer Funktion und eine globale Variable kann in jeder Funktion verwendet werden. Globale Variablen können nur mit konstanten Ausdrücken initialisiert werden.

#### 13.4.2.4 Statische lokale Deklaration

Eine lokale Variable wird zerstört, wenn die Ausführung den Block verlässt, in dem die Variable geschaffen wurde. Lokalen Variablen in einer Funktion existieren nur während der Laufzeit der genannten Funktion. Jede neuer Aufruf der Funktion erstellt und initialisiert neue lokale Variablen. Wenn eine lokale Variable mit dem Schlüsselwort "static" anstatt "new" deklariert ist, bleibt die

Variable auch nach dem Ende einer Funktion im Speicher. Dies bedeutet, dass statische lokale Variablen eine private, dauerhafte Speicherung bereitstellen, die nur in einer einzigen Funktion (oder einem Block) zugänglich sind. Wie globale Variablen, können statische lokale Variablen nur mit konstanten Ausdrücken initialisiert werden.

#### 13.4.2.5 Statische globale Deklaration

Eine statische globale Variable verhält sich wie eine globale Variable, mit dem Unterschied, dass die Variable nur in der Datei gültig ist, in der sie deklariert wurde. Um eine globale Variable statisch zu deklarieren, ersetzen Sie das Schlüsselwort "new" mit "static".

#### 13.4.2.6 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)
```

### 13.4.3 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 277.

## 13.4.4 Array Variablen

### 13.4.4.1 Eindimensionales Array

Die Syntax name[constant] deklariert "name" als ein Array aus “constant” Elementen, wobei jedes Element ein Eintrag ist. "name" ist ein Platzhalter für den Namen der Variable und "constant" ist ein positiver Wert ungleich Null. "constant" ist optional und kann weggelassen werden. Wenn kein Wert zwischen den Klammern steht, ist die Anzahl von Elementen gleich der Anzahl der Initialwerte. Der Array-Index-Bereich ist "Null-basierend", das bedeutet, dass das erste Element "name[0]" und das letzte Element "name[constant-1]" ist.

### 13.4.4.2 Initialisierung

Datenobjekte können bei ihrer Deklaration initialisiert werden. Der initialisierte Wert von globalen Datenobjekten muss ein konstanter Wert sein. Arrays, global oder lokal, müssen ebenfalls mit konstanten Werten initialisiert werden. Nicht initialisierte Daten sind standardmäßig Null.

Beispiele:

Auflistung: gültige Deklaration

```
new i = 1
new j                           /* j ist 0 */
new k = 'a'                     /* k hat den Zeichencode von 'a' */
new a[] = [1,4,9,16,25]         /* a hat 5 Elemente */
new s1[20] = ['a','b']          /* die restlichen 18 Elemente sind 0 */
new s2[] = ''Hello world...''   /* ein unpacked string */
```

Auflistung: ungültige Deklaration

```
new c[3] = 4                    /* Ein Array kann nicht auf einen einzelnen
                                   Wert gesetzt werden */
new i = "Good-bye"              /* Nur ein Array kann einen String halten. */
new q[]                         /* Unbekannte Größe für ein Array */
new p[2] = { i + j, k - 3 }     /* Arrayinitialisierer müssen Konstanten sein. */
```

### 13.4.4.3 Progressive Initialisierung für Arrays

Der Punkte-Operator führt die Initialisierung des Arrays aufgrund der letzten beiden initialisierten Werte weiter. Der Punkte-Operator (drei Punkte, "...") initialisiert das Array bis zur Arraygrenze.

Beispiel: Auflistung: Arrayinitialisierer

```
new a[10] = { 1, ... }              // setzt alle Elemente auf 1
new b[10] = { 1, 2, ... }           // b = 1, 2, 3, 4, 5, 6, 7, 8, 9, 10
new c[8]  = { 1, 2, 40, 50, ... }   // c = 1, 2, 40, 50, 60, 70, 80, 90
new d[10] = { 10, 9, ... }          // d = 10, 9, 8, 7, 6, 5, 4, 3, 2, 1
```

### 13.4.4.4 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 263).

### 13.4.4.5 Arrays und der "sizeof"-Operator

Der "sizeof"-Operator gibt die Anzahl der Elemente einer Variablen zurück. Für eine einfache (nicht Array) Variable ist das Ergebnis von "sizeof" immer 1.

Ein Array mit einer Dimension enthält eine Anzahl von Elementen und der "sizeof"-Operator gibt diese Anzahl zurück. Der Codeausschnitt unterhalb würde deshalb "5" ausgeben, da das Array 4 Zeichen und den Null-Terminator enthält.

```
new msg[] = ''Help''
printf(''%d'', sizeof msg);
```

Der "sizeof"-Operator gibt immer die Anzahl der Einträge, auch für ein "packed" Array, zurück. Der Codeausschnitt unterhalb gibt auch "5" aus, da die Variable 5 Einträge enthält, auch wenn diese im Speicher weniger Platz benötigt.

```
new msg{} = "Help"
printf(''%d'', sizeof msg);
```

Bei mehrdimensionalen Arrays kann der "sizeof"-Operator die Anzahl der Elemente jeder Dimension zurückgeben. Für die letzte (niedrigste) Dimension ist ein Element ein einzelner Eintrag, jedoch für die höchste Dimension ist es ein Sub-Array. Beachten Sie, dass im nachfolgenden Codeausschnitt die Syntax "sizeof matrix" die Anzahl der Elemente der höhere Dimension zurückgibt, und dass die Syntax "sizeof matrix[]" die niedrigere Dimension des zweidimensionalen Arrays ausgibt. Der Codeausschnitt gibt 3 (höhere Dimension) und 2 (niedrigere Dimension) aus.

```
new matrix[3][2] = { { 1, 2 }, { 3, 4 }, { 5, 6 } }
printf(''%d %d'', sizeof matrix, sizeof matrix[]);
```

Die Anwendung des "sizeof"-Operators auf mehrdimensionale Arrays ist besonders praktisch, wenn er als Standardwert für Funktionsargumente verwendet wird.

## 13.4.5 Operatoren und Ausdrücke

### 13.4.5.1 Zeichenerklärung

Die Anwendung von einigen Operatoren hängt von der jeweiligen Art des Operanden ab. Aus diesem Grund wird in diesem Kapitel folgende Notation angewendet:

**e**
: *beliebiger Ausdruck (eng. expression)*

**v**
: *beliebiger Ausdruck, dem ein Wert zugewiesen werden kann (“lvalue” Ausdruck - Variable)*

**a**
: *ein Array*

**f**
: *eine Funktion*

**s**
: *ein Symbol - dies kann eine Variable, eine Konstante oder eine Funktion sein*

### 13.4.5.2 Ausdrücke

Ein Ausdruck besteht aus ein oder mehreren Operanten mit einem Operator. Der Operand kann eine Variable, eine Konstante oder ein anderer Ausdruck sein. Ein Ausdruck gefolgt von einem Semikolon ist ein Statement.

Beispiele für Ausdrücke:

```
v++ f(a1, a2)
v = (ia1 * ia2) / ia3
```

### 13.4.5.3 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.*

### 13.4.5.4 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 o aufgefüllt. |
| ‹‹ | e1 ‹‹ e2 | Ergebnis: Verschiebung nach links von e1 durch e2 Bits. Die freien Bits des Ergebnisses werden mit o aufgefüllt. Es gibt keinen Unterschied zwischen einer arithmetischen und einer logischen Verschiebung nach links. |
| & | e1 & e2 | Ergebnis ist das bitweise logische "und" von e1 und e2. |
| \| | e1 \| e2 | Ergebnis ist das bitweise logische "oder" von e1 und e2. |
| ^ | e1 ^ e2 | Ergebnis ist das bitweise "exklusiv oder" von e1 und e2. |

### 13.4.5.5 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 Operanten nach der arithmetischen oder bitweisen Operation.*

| Operator | Beispiel | Erklärung |
|---|---|---|
| += | v += e | erhöht v um e |
| -= | v -= e | vermindert v um e |
| *= | v *= e | multipliziert v mit e |
| /= | v /= e | dividiert v mit e |
| %= | v %= e | weist v den Rest der Division von v und e zu |
| ››= | v ››= e | verschiebt v arithmetisch um e Bits nach rechts |
| ›››= | v ›››= e | verschiebt v logisch um e Bits nach rechts |
| ‹‹= | v ‹‹= e | verschiebt v um e Bits nach links |
| &= | v &= e | führt ein bitweises "und" von v und e aus und weist das Ergebnis v zu |
| \|= | v \|= e | führt ein bitweises "oder" von v und e aus und weist das Ergebnis v zu |
| ^= | v ^= e | führt ein bitweises "exklusiv oder" von v und e aus und weist das Ergebnis v zu |

### 13.4.5.6 Vergleichsoperatoren

Ein logisches "false" wird durch einen Integer-Wert von 0 repräsentiert; ein logisches "true" durch einen Wert, der nicht 0 ist. Ergebnisse eines Vergleichs-Ausdrucks sind entweder 0 oder 1 und ihr "tag" wird auf "bool" gesetzt.

| Operator | Beispiel | Erklärung |
|---|---|---|
| == | e1 == e2 | Das Ergebnis ist "true", wenn e1 und e2 gleich sind. |
| != | e1 != e2 | Das Ergebnis ist "true", wenn e1 nicht gleich e2 ist. |

*Hinweis: Die folgenden Operatoren können verkettet werden, wie im Ausdruck "e1 <= e2 <= e3". Dies bedeutet, dass das Ergebnis "1" ist, wenn jeder einzelne Vergleich zutrifft und "0", wenn zumindest ein Vergleich nicht zutrifft.*

| Operator | Beispiel | Erklärung |
|---|---|---|
| < | e1 < e2 | Das Ergebnis ist ein logisches "true", wenn e1 kleiner ist als e2. |
| <= | e1 <= e2 | Das Ergebnis ist ein logisches "true", wenn e1 kleiner oder gleich e2 ist. |
| > | e1 > e2 | Das Ergebnis ist ein logisches "true", wenn e1 größer ist als e2. |
| >= | e1 >= e2 | Das Ergebnis ist ein logisches "true", wenn e1 größer oder gleich e2 ist. |

### 13.4.5.7 Boolean

Ein logisches "false" wird durch einen Integer-Wert von 0 repräsentiert, ein logisches "true" durch einen Wert, der nicht 0 ist. Ergebnisse eines Vergleichs-Ausdrucks sind entweder 0 oder 1 und ihr "tag" wird auf "bool" gesetzt.

| Operator | Beispiel | Erklärung |
|---|---|---|
| ! | !e | Das Ergebnis ist eine logisches "true", wenn e logisch "false" ist. |
| \|\| | e1 \|\| e2 | Das Ergebnis ist "true", wenn entweder e1 oder e2 (oder beide) logisch "true" sind. Der Ausdruck e2 wird nur ausgewertet, wenn e1 logisch "false" ist. |
| && | e1 && e2 | Das Ergebnis ist "true", wenn e1 und e2 logisch "true" sind. Der Ausdruck e2 wird nur ausgewertet, wenn e1 logisch "true" ist. |

### 13.4.5.8 Sonstiges

| Operator | Beispiel | Erklärung |
|---|---|---|
| [ ] | a[e] | Array Index: Das Ergebnis ist der Eintrag an der Position e des Arrays a. |
| { } | a{e} | Array Index: Das Ergebnis ist das Zeichen an der Position e des "packed" Arrays a. |
| ( ) | f(e1, e2, ... eN) | Das Ergebnis ist der Wert, der von der Funktion f zurückgegeben wird. Die Funktion wird mit den Parametern e1, e2, ... eN aufgerufen. Die Reihenfolge der Auswertung der Parameter ist nicht definiert. (Die Implementation der Script-Engine wertet die Parameter möglicherweise in umgekehrter Reihenfolge aus.) |
| ? : | e1 ? e2 : e3 | Das Ergebnis ist entweder e2 oder e3, abhängig vom Wert von e1. Der bedingte Ausdruck ist ein zusammengesetzter Ausdruck mit einem zweiteiligen Operator, "?" und ":". Der Ausdruck e2 wird ausgewertet, wenn e1 logisch "true" ist, e3 wird ausgewertet, wenn e1 logisch "false" ist. |
| : | tagname: e | "tag" Überschreibung: Der Wert des Ausdrucks ändert sich nicht, jedoch ändert sich der "tag". |
| defined | defined s | Ergebnis ist "1", wenn das Symbol definiert wurde. Das Symbol kann eine Konstante oder eine globale oder lokale Variable sein. Der "tag" des Ausdrucks ist "bool". |
| sizeof | sizeof s | Das Ergebnis ist die Anzahl der Elemente der angegeben Variable. Für einfache Variablen und für eindimensionale Arrays ist ein Element ein Eintrag. Für mehrdimensionale Arrays ist das Ergebnis die Anzahl der Elemente (Sub Arrays) in der höchsten Dimension. Fügen Sie [] zum Namen des Arrays hinzu, um eine niedrigere Dimension anzugeben. Wenn die Größe der Variable nicht bekannt ist, dann ist das Ergebnis 0. Wenn dieser Operator in einem "default"-Wert einer Funktion verwendet wird, dann wird der Ausdruck zum Zeitpunkt des Aufrufs der Funktion und nicht zum Zeitpunkt der Definition ausgeführt. |
| tagof | tagof s | Das Ergebnis ist eine eindeutige Zahl, die den "tag" der Variablen, der Konstanten, des Rückgabewerts einer Funktion oder des Names der "tag"-Bezeichnung repräsentiert. Wenn dieser Operator in einem "default"-Wert einer Funktion verwendet wird, dann wird der Ausdruck zum Zeitpunkt des Aufrufs der Funktion und nicht zum Zeitpunkt der Definition ausgeführt. |

### 13.4.5.9 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 |
|---|---|---|
| ( )<br>[ ]<br>{ } | Funktionsaufruf<br>Array Index (Element)<br>Array Index (Zeichen) | links-nach-rechts |
| !<br>~<br>-<br>++<br>--<br>:<br>defined<br>sizeof<br>tagof | logisches Nicht<br>Einerkomplement<br>Zweierkomplement (unäres Minus)<br>Erhöhung<br>Verringerung<br>"tag"-Überschreibung<br>Symbol Definitions-Status<br>Symbol Größe in "Elementen"<br>eindeutige Zahl des "tags" | rechts-nach-links |
| *<br>/<br>% | Multiplikation<br>Division<br>Modulo | links-nach-rechts |
| +<br>- | Addition<br>Subtraktion | links-nach-rechts |
| ››<br>›››<br>‹‹ | arithemtische Verschiebung nach rechts<br>logische Verschiebung nach rechts<br>Verschiebung nach links | links-nach-rechts |
| & | bitweises "und" | links-nach-rechts |
| ^ | bitweises "exklusiv oder" | links-nach-rechts |
| \| | bitweises "oder" | links-nach-rechts |
| ‹<br>‹=<br>›<br>›= | kleiner als<br>kleiner oder gleich als<br>größer als<br>größer oder gleich als | links-nach-rechts |
| ==<br>!= | gleich<br>ungleich | links-nach-rechts |
| && | logisches "und" | links-nach-rechts |
| \|\| | logisches "oder" | links-nach-rechts |
| ? : | bedingte Ausführung | rechts-nach-links |
| = | Zuweisung<br>*= /= %= += -= ››= ›››= ‹‹= &= ^= \|= | rechts-nach-links |
| , | Komma | links-nach-rechts |

## 13.4.6 Anweisungen

Ein Statement kann aus einer oder mehreren Zeilen bestehen. Eine Zeile kann zwei oder mehrere Statements enthalten.

Statements zur Ablaufsteuerung (if, if-else, for, while, do-while und switch) können geschachtelt werden.

### 13.4.6.1 Statement-Etikett

Ein Etikett besteht aus einem Identifizierer gefolgt von einem ":". Ein Etikett ist ein "Sprung-Ziel" eines "goto" Statements.

Jede Anweisung kann mit einem Etikett versehen werden. Es muss dem Etikett ein Statement folgen, dies kann auch ein "leeres Statement" sein.

Der Gültigkeitsbereich eines Etiketts ist die Funktion in der es deklariert wurde d.h. ein "goto"-Statement kann nicht aus der aktuellen Funktion in eine andere Funktion springen.

### 13.4.6.2 Zusammengesetzte Anweisungen

Eine zusammengesetzte Anweisung (auch Block genannt) ist eine Serie von Null oder mehreren Anweisungen, welche durch Klammern ("{" und "}") umgeben ist. Die schließende Klammer ("}") darf nicht mit einem Semikolon abgeschlossen werden. Jede Anweisung kann durch einen Block ersetzt werden. Eine zusammengesetzte Anweisung, die keine Anweisungen enthält, ist ein Spezialfall und wird "leeres Statement" genannt.

### 13.4.6.3 Ausdrucksanweisung

Jeder Ausdruck wird zu einem Statement, wenn ein Semikolon (";") angehängt wird. Ein Ausdruck wird auch zu einem Statement, wenn dem Ausdruck bis zum Ende der Zeile nur Leerzeichen folgen und der Ausdruck nicht in der nächsten Zeile weitergeführt werden kann.

### 13.4.6.4 Leeres Statement

Eine leeres Statement führt keine Anweisungen aus und besteht aus einem Block-Statement ohne Anweisungen, d.h. es besteht aus dem Symbol "{ }". Leere Statements werden in Kontrollflussanweisungen ohne Aktionen eingesetzt (z.B. "while (!iskey()) {}") oder, wenn ein Etikett genau vor einer schließenden Klammer eines Block-Statements definiert wird. Ein leeres Statement endet nicht mit einem Semikolon.

### 13.4.6.5 assert Ausdruck

bricht das Programm mit einem Laufzeitfehler ab, wenn der Ausdruck logisch "false" ergibt

> ***Hinweis:*** *Dieser Ausdruck schützt vor "unmöglich" oder ungültigen Bedingungen. Im folgenden Beispiel ist eine negative Fibonacci-Zahl ungültig. Die assert-Anweisung markiert diesen Fehler als Programmierer-Fehler. assert-Anweisungen sollten nur Programmierer-Fehler kennzeichnen und niemals Benutzereingaben.*

Beispiel:

```
fibonacci(n)
{
  assert n > 0

  new a = 0, b = 1
  for (new i = 2; i < n; i++)
  {
    new c = a + b
    a = b
    b = c
  }
  return a + b
}
```

### 13.4.6.6 break

beendet und verlässt das kleinste, umschließende "do"-, "for"- oder "while"-Statement an jedem beliebigen Punkt in der Schleife. Das "break"-Statement bewegt den Programmfluss zum nächsten Statement außerhalb der Schleife.

Beispiel:

```
example(n)
{
  new a = 0

  for(new i = 0; i < n ; i++ )
  {
    a += i

    if(i>10)
      break

    a += 1
  }
  return a
}
```

### 13.4.6.7 continue

beendet die aktuelle Iteration der kleinsten umschließenden "do"-, "for"- oder "while"-Anweisung und bewegt die Programmsteuerung an den Bedingungsteil der Schleife.

Beispiel

```
example(n)
{
  new a = 0

  for(new i = 0; i < n ; i++ )
  {
    a += i

    if(i>10)
      continue

    a += 1
  }
  return a
}
```

### 13.4.6.8 do Statement while ( Ausdruck )

führt ein Statement aus, bevor der Bedingungsteil (die "while"-Bedingung) evaluiert wird. Das Statement wird wiederholt, solange die Bedingung logisch "true" ist. Das Statement wird zumindest einmal ausgeführt.

Beispiel:

```
example(n)
{
  new a = 0

  do
  {
    a++
  }
  while(n >= 0)

  return a
}
```

### 13.4.6.9 exit Ausdruck

bricht das Programm ab. Der Ausdruck ist optional, aber wenn er vorhanden ist, muss er in der selben Zeile wie das "exit"-Statement beginnen und enden. Die "exit"-Anweisung gibt den Wert des Ausdrucks zurück an die Hauptanwendung oder gibt Null zurück, wenn kein Ausdruck angegeben wird.

### 13.4.6.10 for ( Ausdruck 1 ; Ausdruck 2 ; Ausdruck 3 ) Statement

Alle drei Ausdrucke sind optional.

**Ausdruck 1:**

*wird nur einmal ausgewertet, vor Eintritt in die Schleife. Dieser Ausdruck kann zum Initialisieren einer Variablen genutzt werden. Dieser Ausdruck hält auch die Variablendeklaration mittels der "new"-Syntax. Eine Variable, die an dieser Stelle deklariert wird, ist nur innerhalb der Schleife gültig. Es ist nicht möglich einen Ausdruck (mit bereits vorhandenen Variablen) und eine Deklaration von neuen Variablen in diesem Feld zu kombinieren. Es müssen entweder alle Variablen in diesem Feld bereits vorhanden sein, oder sie müssen alle in diesem Bereich deklariert werden.*

**Ausdruck 2:**

*Dieser Ausdruck wird vor jedem Durchlauf der Schleife ausgeführt und beendet die Schleife, wenn der Ausdruck logisch "false" zurückgibt. Wenn dieser Ausdruck weggelassen wird, wird das Ergebnis des Ausdrucks 2 als logisch "true" angenommen.*

**Ausdruck 3:**

*Dieser Ausdruck wird nach jeder Ausführung des Statements ausgeführt. Die Programmsteuerung bewegt sich von Ausdruck 3 zum Ausdruck 2 für die nächste (bedingte) Iteration der Schleife.*

Beispiel:

```
example(n)
{
  new a = 0

  for(new i = 0; i < n; i++)
  {
    a++
  }

  return a
}
```

Das Statement "for ( ; ; )" ist gleich dem Statement "while (true)".

### 13.4.6.11 goto Etikett

bewegt die Programmsteuerung (unbedingt) zu der Anweisung, die dem angegebenen Etikett folgt. Das Etikett muss innerhalb der gleichen Funktion wie die "goto"-Anweisung sein. Eine "goto"-Anweisung kann nicht aus einer Funktion springen.

### 13.4.6.12 if ( Ausdruck ) Statement 1 else Statement 2

führt das Statement 1 aus, wenn das Ergebnis des Ausdrucks logisch "true" ergibt. Die "else"-Klausel des "if"-Statements ist optional. Wenn das Ergebnis des Ausdruck logisch "false" ergibt und eine "else"-Klausel existiert, dann wird das Statement, das mit der "else"-Klausel assoziiert ist, (Statement 2) ausgeführt.

Beispiel:

```
example(n)
{
  if(n < 0)
    return -1
  else if (n == 0)
    return 0
  else
    return 1
}
```

### 13.4.6.13 return Ausdruck

beendet die aktuelle Funktion und bewegt die Programmsteuerung zum nächsten Statement nach dem Funktionsaufruf. Der Wert des Ausdrucks wird als Funktionsergebnis zurückgeliefert. Der Ausdruck kann ein Array oder eine Zeichenfolge sein. Der Ausdruck ist optional, wenn er jedoch vorhanden ist, muss er an der selben Zeile beginnen, wie das "return"-Statement. Wenn kein Ausdruck angegeben ist, dann wird Null zurückgeliefert.

### 13.4.6.14 switch ( Ausdruck ) { case Liste }

überträgt die Ablaufsteuerung an unterschiedliche Statements innerhalb des "switch" in Abhängigkeit vom Wert des "switch"-Ausdrucks. Der Hauptteil der "switch"-Anweisung ist eine zusammengesetzte Anweisung, die eine Reihe von "case"-Klauseln enthält. Jede "case"-Klausel beginnt mit dem Schlüsselwort "case", gefolgt von einer Liste von Konstanten und einem Statement. Die Liste der Konstanten ist eine Serie von Ausdrücken, getrennt durch Kommas, die jeweils zu einem konstanten Wert ausgewertet werden. Diese Liste endet mit einem Doppelpunkt. Um einen Bereich in dieser Liste anzugeben, trennen Sie die untere und obere Grenze des Bereichs mit einem doppelten Punkt (".."). Ein Beispiel für einen Bereich ist: "case 1..9:".

Das "switch"-Statement bewegt die Ablaufsteuerung zu einer "case"-Klausel, wenn ein Wert der Liste dem Wert des "switch"-Ausdrucks entspricht.

Die "default"-Klausel besteht aus dem Schlüsselwort "default" und einem Doppelpunkt. Die "default"-Klausel ist optional, aber wenn sie angegeben wird, muss sie als letzter Eintrag in der "case"-Liste eingetragen sein. Das "switch"-Statement bewegt die Ablaufsteuerung zur "default"-Klausel, wenn keine der "case"-Klauseln mit dem "switch"-Ausdruck übereinstimmt.

Beispiel:

```
example(n)
{
  new a = 0

  switch (n)
  {
    case 0..3:
      a = 0
    case 4,6,8,10:
      a = 1
    case 5,7:
      a = 2
    case 9:
      a = 3
    default:
      a = -1
  }

  return a
}
```

#### 13.4.6.15 while ( Ausdruck ) Statement

wertet den Ausdruck aus und führt das Statement aus, wenn das Ergebnis des Ausdrucks logisch "true" ergibt. Nachdem die Anweisung ausgeführt wurde, kehrt die Programmsteuerung erneut zu dem Ausdruck zurück. Das Statement wird daher ausgeführt, solange der Ausdruck logisch "true" ist.

Beispiel:

```
example(n)
{
  new a = 0

  while(n >= 0)
  {
    a++
  }

  return a
}
```

### 13.4.7 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 264 behandelt.

Normalerweise beinhalten Funktionen lokale Variablendeklarationen und bestehen aus einer Block-Anweisung.

> ***Hinweis:*** *Im nächsten Beispiel verhindert die "assert"-Anweisung negative Werte für den Exponenten*

```
power(x, y)
{
  /* gibt x hoch y zurück*/
  assert y >= 0

  new r = 1
  for (new i = 0; i < y; i++)
    r *= x

  return r
}
```

Eine Funktion kann mehrere "return"-Anweisungen enthalten, eine wird z.B. benutzt um schnell eine Funktion zu beenden, wenn ungültige Parameter übergeben werden, oder wenn sich herausstellt, dass die Funktion nichts zu tun hat. Wenn eine Funktion ein Array zurückgibt, müssen alle "return" Anweisungen ein Array mit derselben Anzahl von Einträgen zurückgeben.

### 13.4.7.1 Funktionsargumente ("call-by-value" versus "call-by-reference")

Die "faculty"-Funktion im nächsten Beispiel hat einen Parameter, der in der Schleife benutzt wird um die Fakultät dieser Zahl zu berechnen. Was Aufmerksamkeit verdient ist, dass die Funktion das Argument modifiziert.

```
main()
{
  new v = 5
  new f = faculty(v)
}

faculty(n)
{
  assert n >= 0

  new result = 1
  while (n > 0)
    result *= n--

  return result
}
```

Egal welchen (positiven) Wert die Variable "n" am Beginn der "while"-Schleife hatte, am Ende der Funktion wird "n" Null sein. Am Beispiel der Funktion "faculty" wird der Parameter als Wert ("by value") übergeben, damit ist eine Änderung der Variable "n" nur lokal in der Funktion "faculty" gültig. Mit anderen Worten, die Variable "v" in der Funktion "main()" hat vor und nach dem Aufruf der Funktion denselben Wert.

Argumente können als Wert ("by value") oder als Referenz ("by reference") übergeben werden. Einem Funktionsargument, das als Referenz übergeben werden soll, muss als Präfix "&" dem Namen vorangestellt werden. Als Standard werden der Funktion die Argumente als Wert übergeben.

Beispiel:

```
swap(&a, &b)
{
  new temp = b
  b = a
  a = temp
}
```

Um eine Array einer Funktion zu übergeben, fügen Sie ein Klammernpaar ("[]") dem Namen des Arguments an. Es kann auch zusätzlich die Anzahl der Einträge angegeben werden. Dies verbessert die Fehlererkennung des Parsers des Compilers.

Beispiel:

```
addvector(a[], const b[], size)
{
  for (new i = 0; i < size; i++)
    a[i] += b[i]
}
```

Arrays werden immer als Referenz übergeben.

> ***Hinweis:*** *Das Array "b" im oben gezeigten Beispiel wird in der Funktion nicht verändert. Dieses Funktionsargument wurde als "const" deklariert, um dies explizit zu machen. Zusätzlich zur verbesserten Fehlererkennung, erlaubt es dem Compiler einen effizenteren Code zu generieren.*

Das folgende Codebeispiel ruft die Funktion "addvector" auf und addiert zu jedem Element der Variablen "vect" den Wert 5:

```
new vect[3] = [ 1, 2, 3 ]

addvector(vect, [5, 5, 5], 3)

/* vect[] beinhaltet nun die Werte 6, 7 und 8 */
```

### 13.4.7.2 Benannte Parameter versus positionsgebundene Parameter

In den vorangegangenen Beispielen war die Reihenfolge der Parameter bei einem Funktionsaufruf wichtig, da jeder Parameter zu dem Funktionsparameter mit derselben Position kopiert wurde. Zum Beispiel bei der "weekday" Funktion, die unterhalb definiert wird, würde der Aufruf "weekday(12, 31, 1999)" lauten, um den Wochentag des letzten Tages des letzten Jahrhunderts zu erhalten.

```
weekday(month, day, year)
{
  /* gibt den Tag der Woche zurück: 0=Samstag, 1=Sonntag, etc. */
  if (month <= 2)
    month += 12, --year

  new j = year % 100
  new e = year / 100
  return (day + (month+1)*26/10 + j + j/4 + e/4 - 2*e) % 7
}
```

Das Datumsformat unterscheidet sich je nach Kultur und Nation, während in den Vereinten Staaten von Amerika das Format Monat/Tag/Jahr verbreitet ist, verwenden europäische Staaten oft das Format Tag/Monat/Jahr und in technischen Publikationen wird Jahr/Monat/Tag (ISO/IEC 8824) verwendet. Mit anderen Worten, keine Reihenfolge der Parameter ist "standardisiert" oder "normal". Aus diesem Grund gibt es eine alternative Möglichkeit, um Parameter an eine Funktion zu übergeben: die "benannten Parameter". Diese wird im nächsten Beispiel gezeigt (die Funktion wurde genau wie im vorherigen Beispiel deklariert).

```
new wkday1 = weekday( .month = 12, .day = 31, .year = 1999)
new wkday2 = weekday( .day = 31, .month = 12, .year = 1999)
new wkday3 = weekday( .year = 1999, .month = 12, .day = 31)
```

Bei "benannten Parametern" wird ein Punkt (".") dem Namen des Arguments vorangestellt. Das Argument der Funktion kann auf einen beliebigen Ausdruck gesetzt werden, der gültig für das Argument ist. Das Gleichheitszeichen ("=") hat im Falle eines benannten Parameters nicht die Bedeutung einer Zuordnung, sondern verknüpft den Ausdruck mit einem Funktionsargument.

Es können positionsgebundene und benannte Parameter vermischt werden, jedoch müssen die positionsgebundenen vor den benannten Parametern angegeben werden.

### 13.4.7.3 Standardwerte von Funktionsargumenten

Ein Funktionsargument kann einen Standardwert haben. Der Standardwert eines Funktionsarguments muss eine Konstante sein. Um einen Standardwert anzugeben, fügen Sie an den Namen des Parameters ein Gleichheitszeichen ("=") und den Wert an.

Wenn bei einem Funktionsaufruf ein Platzhalter anstelle eines gültigen Funktionsparameters angegeben wird, wird der Standardwert übernommen. Der Platzhalter ist das Unterstrichzeichen ("_"). Der Argumentplatzhalter ist nur für Parameter mit einem Standardwert gültig.

Die rechten Argumentplatzhalter können von der Argumentenliste entfernt werden.

Zum Beispiel, wenn die Funktion "increment" wie folgt definiert ist:

```
increment(&value, incr=1)
{
  value += incr
}
```

sind die folgenden Funktionsaufrufe alle gleich:

```
increment(a)
increment(a, _)
increment(a, 1)
```

Standardwerte für Argumente, die als Referenz übergeben werden, sind hilfreich um diese Parameter optional zu machen. Zum Beispiel, wenn die Funktion "divmod" geschrieben wurde, um sowohl den Quotienten als auch den Rest als Parameter zu übergeben.

```
divmod(a, b, &quotient=0, &remainder=0)
{
  quotient = a / b
  remainder = a % b
}
```

Mit der vorangegangenen Definition der Funktion "divmod" sind die folgenden Funktionsaufrufe alle gültig:

```
new p, q

divmod(10, 3, p, q)
divmod(10, 3, p, _)
divmod(10, 3, _, q)
divmod(10, 3, p)
divmod 10, 3, p, q
```

Das nächste Beispiel addiert die Werte von einem Array zu einem anderen. Wenn nur ein Parameter angegeben wird, dann werden die Werte des Arrays um 1 erhöht:

```
addvector(a[], const b[] = {1, 1, 1}, size = 3)
{
  for (new i = 0; i < size; i++)
    a[i] += b[i]
}
```

## 13.5 Beispiele

t.b.d.

## 13.6 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 259).
- 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 277). Das "Platzhalter"-Argument ersetzt einige Verwendungen des NULL-Zeigers (siehe "Standardwerte von Funktionsargumenten" auf Seite 279).
- Zahlen können mit Hexadezimal-, Dezimal-, oder Binärbasis angegeben werden. Die Oktale Basis wird nicht unterstützt (siehe "Numerische Konstanten" auf Seite 259). 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 274). 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 268.
- Eine leere Anweisung ist ein leerer Block (mit {}), nicht ein Semikolon (siehe "Zusammengesetzte Anweisungen" auf Seite 270). 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 264).
- 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 deklariere 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 14 myDatanet Connector

Das Grundprinzip des rapidM2M Moduls ist eine sogenannte Storage-2-Storage-Datenübertragung. Für diese Art der Datenübermittlung müssen weder das rapidM2M Modul 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 rapidM2M Modul 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 10kB zur Verfügung, die ebenfalls beliebig genutzt werden können.

Um die von einem rapidM2M Modul 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.

## 14.1 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 284) 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 |

## 14.2 Split-tag

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

Ein rapidM2M Modul 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") wird das erste Byte ausgewertet. Bei Werten kleiner/gleich 9 wird der Datensatz, vorausgesetzt es ist kein anders lautender Split-tag vorhanden, automatisch in den jeweiligen strukturierten Messdatenkanal kopiert. Enthält das erste Byte den Wert 0, wird der Datensatz in Messdatenkanal 0 ("histdata0") kopiert. Enthält das erste Byte den Wert 9, wird der Datensatz in Messdatenkanal 9 ("histdata9") kopiert. Eine nachträgliche Verarbeitung ist nicht vorgesehen.*

***Wichtiger Hinweis:*** *Wollen Sie nur den Rohdatensatzes ("rm2mraw") verwenden, sollten Sie dafür Sorge tragen, dass das erste Byte des Rohdatensatzes immer einen Wert größer 9 enthält.*

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**

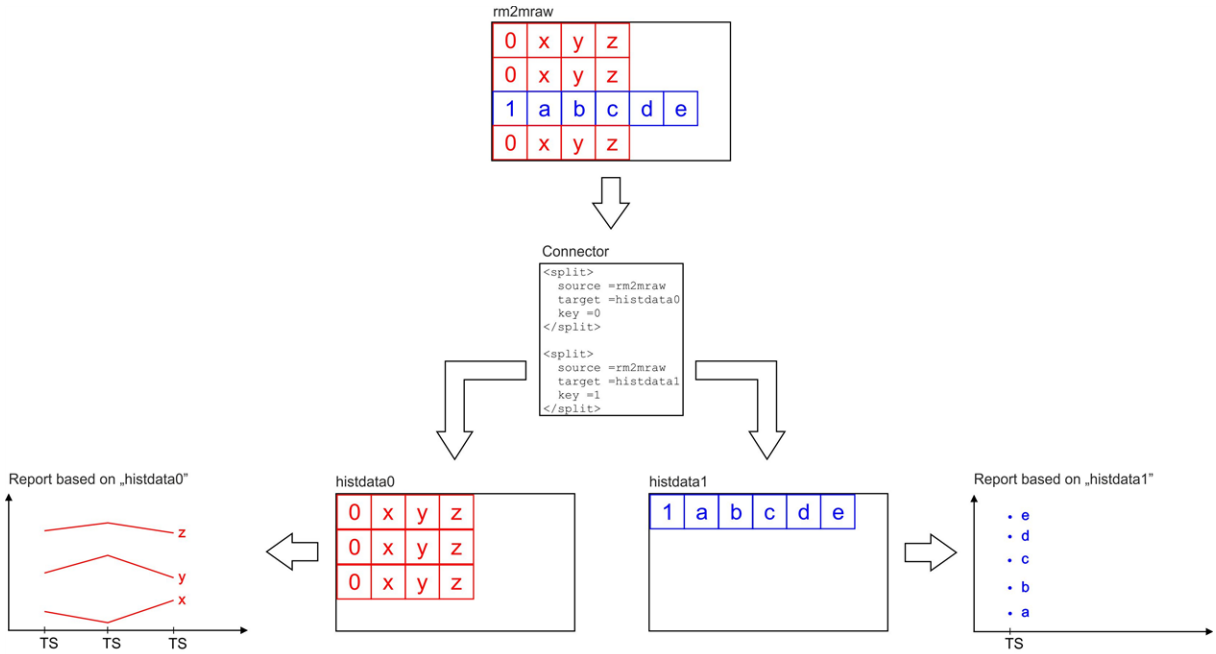
*Dieses Feld bietet die Möglichkeit zu bestimmen, ob der aktuelle Datensatz kopiert werden soll:*

- *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*

# 14.3 Datenstruktur

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

> ***Wichtiger Hinweis:*** *Soll ein strukturierter Messdatenkanal oder Konfigurationsblock am Cloud-Server 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 291.

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

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

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

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

## 14.3.1 Attribute der Feld-Definition

**name**

*Alphanumerisch.*

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

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

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

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

**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 Cloud-Servers bzw. die Eingabe über die Oberfläche des Cloud-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 Cloud-Servers sichtbar ist.*

edit

*Numerisch, ganzzahlig, positiv. Gibt den Benutzerlevel an, der erforderlich ist, um den Feldinhalt über die Oberfläche des Cloud-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.*

## 14.4 Beispiel

| | |
|---|---|
| `<table>` | Beginn des Table-Tags |
| `  name   =histdata0` | Name der zu beschreibenden Datentabelle |
| `  <field>` | Beginn einer Feld-Definition (cho) |
| `    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, cho, ch2, ch3.

## 14.5 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 | oxFF | oxFFFF | oxFFFFFFFF |
| OF | oxFE | oxFFFE | oxFFFFFFFE |
| UF | oxFD | oxFFFD | oxFFFFFFFD |
| OL | oxFC | oxFFFC | oxFFFFFFFC |
| SC | oxFB | oxFFFB | oxFFFFFFFB |

**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 | 0x7F8xxxxx | 0x7FFxxxxxxxxxxxx |
| OF | nicht verfügbar | nicht verfügbar |
| UF | nicht verfügbar | nicht verfügbar |
| OL | nicht verfügbar | nicht verfügbar |
| SC | nicht verfügbar | nicht verfügbar |

Für f32 und f64 zählen für die Interpretation von NaN nur die ersten 12 Bytes.

# Kapitel 15 Fehlersuche und Behebung

## 15.1 Allgemeine Probleme

| Problem | Ursache/Lösung |
|---|---|
| Modul zeigt keine Reaktion (Status-LED nach Drücken der Taste zum Einschalten des Wireless GSM Controllers immer aus) | • Kabelverbindungen überprüfen (siehe "Anschluss der Sensoren, der Aktoren und der Versorgung" auf Seite 94)<br>• Prüfen Sie, ob der Jumper zum Aktivieren der Energieversorgung des Moduls (J104) korrekt gesetzt ist (siehe "Jumper zum Aktivieren der Energieversorgung des M120 Moduls (J104) " auf Seite 81).<br>• Überprüfen Sie, ob das rapidM2M Modul M120 korrekt in die Aufnahme am rapidM2M Developer Board eingesetzt ist (siehe "Einsetzen des M120 Moduls" auf Seite 93). |
| Kommunikationsprobleme | • Laden Sie das Gerätelog mit Hilfe des rapidM2M Toolset vom M120 Modul (siehe "Karteireiter "Log"" auf Seite 154). Eine Auflistung aller möglichen Fehlercodes finden Sie im Kapitel „Log-Einträge und Fehlercodes“ (siehe "Log-Einträge und Fehlercodes " auf Seite 295). |
| 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 t.b.d.). Lösung: Übertragung auslösen oder auf die nächste zyklische Übertragung warten.<br>• Der myDatanet Connector wurde nicht korrekt konfiguriert (siehe "myDatanet Connector " auf Seite 283). |
| Alarmnachricht wurde nicht zugestellt, obwohl der Alarm signalisiert wird. | • Einstellungen des Alarmrufplans prüfen (siehe t.b.d.).<br>• Adressdaten des Alarmrufplans prüfen (siehe t.b.d.). |
| 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 193).<br>• Laden Sie das Gerätelog mit Hilfe des rapidM2M Toolset vom M120 Modul (siehe "Karteireiter "Log"" auf Seite 154). Eine Auflistung aller möglichen Fehlercodes des Pawn Scripts finden Sie im Kapitel „Pawn Script Fehlercodes“ (siehe "Pawn Script Fehlercodes" auf Seite 255).<br>• Durch einen Kontextwechsel (Zuweisung einer anderen Messstelle / Applikation) wurde das bisherige Script durch jenes der neu zugewiesenen Messstelle / Applikation ersetzt.<br>• Durch die Zuweisung einer neuen oder anderen Messstelle / Applikation wurde das per USB eingespielte Script durch jenes der neu zugewiesenen Messstelle / Applikation ersetzt. |

| Problem | Ursache/Lösung |
| --- | --- |
| Der Buzzer gibt einen Dauerton ab. | • Prüfen Sie, ob Sie per Script den GPIO1 auf "low" gesetzt haben.<br>• Überprüfen Sie die Stellung der Schalter 1 und 2 des Dip-Switches (S207) (siehe "Ausgabe von digitalen Signalen" auf Seite 73). Durch Setzen beider Schalter auf "OFF", wird die Signalausgabe in jedem Fall unterbunden. |
| An einem GIPO wird nicht der gewünschte Pegel ausgegeben. | • Prüfen Sie, ob der GPIO mittels der PAWN-Script-Funktion "rM2M_GpioDir()" als Ausgang konfiguriert wurde.<br>• Überprüfen Sie, ob mittels Dip-Switch S206 der Pull up- bzw. Pull down-Widerstand aktiviert ist (siehe "Ausgabe von digitalen Signalen" auf Seite 73). Die Ausgangsstufe des GPIOs ist nicht in der Lage den Pegel entgegen dem Widerstand zu treiben. D.h. ist der Pull up aktiv, kann die Ausgangsstufe den Pegel nicht auf "low" ziehen. |

## 15.2 Log-Einträge und Fehlercodes

| Log-Eintrag | | Parameter | | Beschreibung |
|---|---|---|---|---|
| Code | Klartext | Code | Klartext | |
| 1000 | POWER ON | 0 | --- | Systemstart abgeschlossen |
| | | › 0 | --- | Interner System-Fehler<br><br>Neustart aufgrund eines internen System-Fehlers. Sollte der "POWER ON" Log-Eintrag mehrmals mit einem Parameter-Code › 0 im Gerätelog enthalten sein, liegt unter Umständen ein Hardwareproblem vor. Kontaktieren Sie in diesem Fall den Hersteller (siehe "Kontaktinformationen" auf Seite 313). |
| 1020 | ERROR SOD | ## | --- | Interner System-Fehler<br><br>Sollte dieser Fehler mehrmals mit demselben Parameter-Code im Gerätelog enthalten sein, liegt unter Umständen ein Hardwareproblem vor. Kontaktieren Sie in diesem Fall den Hersteller (siehe "Kontaktinformationen" auf Seite 313). |
| 1033 | MODEM_UPDATE | ## | --- | Update der Firmware des Modemcontrollers wurde erfolgreich durchgeführt.<br><br>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. |
| 1161 | LOG REFORMATFILE | --- | --- | Das Filesystem wurde neu formatiert. D.h. alle bisher aufgezeichneten Daten sowie Log-Einträge wurden gelöscht. |
| 1200 | MODEM ERROR | | | Modemfehler (siehe "Modemfehler" auf Seite 298) |
| 1202 | MODEM CMME ERROR | ## | --- | Das GPRS-Modem meldet einen +CME-Fehler. Der Parameter gibt, an um welchen Fehler es sich handelt. |

| Log-Eintrag | | Parameter | | Beschreibung |
|---|---|---|---|---|
| Code | Klartext | Code | Klartext | |
| 1203 | SELECTED NETWORK | ## | --- | Ein neues GSM-Netzwerk wurde gewählt.<br>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. |
| 1204 | BANNED NETWORK | ## | --- | Ein GSM-Netzwerk wurde auf die Banlist gesetzt.<br>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. |
| 1205 | BAN LIST CLEARED | --- | --- | Die Banlist wurde gelöscht, da sich keine GSM-Netze mehr im Empfangsbereich befinden, die nicht auf die Banlist gesetzt wurden. |
| 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. |
| 1224 | MODEM SIM RESET | --- | --- | Das GRPS-Modem wurde neu gestartet, um den SIM-Chip neu zu initialisieren. |
| 1252 | MODEM TO CON | ## | --- | Timeout während des Verbindungsaufbaus. Der Parameter enthält dabei einen Zähler, der angibt wie viele Verbindungsaufbauversuche hintereinander nicht funktioniert haben. |
| 1281 | ZLIB STREAMPROCESS ERR | ## | --- | interner Fehler<br>Sollte dieser Fehler mehrmals im Gerätelog enthalten sein, kontaktieren Sie den Hersteller (siehe "Kontaktinformationen" auf Seite 313). |

| Log-Eintrag | | Parameter | | Beschreibung |
|---|---|---|---|---|
| Code | Klartext | Code | Klartext | |
| 1282 | ZLIB STREAMFINISH ERR | ## | --- | interner Fehler<br>Sollte dieser Fehler mehrmals im Gerätelog enthalten sein, kontaktieren Sie den Hersteller (siehe "Kontaktinformationen" auf Seite 313). |
| 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 - 2999 | MODULE ERR | | | Bereich für die kundenspezifischen Fehlercodes, die mittels der Pawn Script Funktion "rM2M_WriteLog()" in das Gerätelog geschrieben werden können (siehe "rapidM2M Funktionen" auf Seite 202). |
| 3000 - 3099 | | | | Fehlercodes der Scriptausführung (siehe "Pawn Script Fehlercodes" auf Seite 255). |

## 15.2.1 Modemfehler

| Log-Eintrag | | Parameter | | Beschreibung |
|---|---|---|---|---|
| Code | Klartext | Code | Klartext | |
| GPRS-Fehler (1/2) | | | | |
| 1200 | BEARER LINE BUSY | -993 | --- | Leitung belegt<br>später erneut versuchen |
| | BEARER NO ANSWER | -992 | --- | Keine Antwort<br>• später erneut versuchen<br>• Überprüfen Sie, ob sich das Gerät im Versorgungsbereich befindet (www.microtronics.at/footprint). |
| | BEARER NO CARRIER | -991 | --- | Kein Träger<br>• später erneut versuchen<br>• Überprüfen Sie, ob sich das Gerät im Versorgungsbereich befindet (www.microtronics.at/footprint). |
| | BEARER GPRS FAILED | -988 | --- | GPRS Setup-Fehler<br>• Versuchen Sie die Antennenposition zu verbessern.<br>• Überprüfen Sie, ob sich das Gerät im Versorgungsbereich befindet (www.microtronics.at/footprint). |
| | BEARER PPP LCP FAILED | -987 | --- | Fehler beim Aushandeln des LCP<br>interner Fehler |
| | BEARER PPP AUTH FAILED | -986 | --- | Fehler bei der PPP Authentifizierung<br>interner Fehler |
| | BEARER PPP IPCP FAILED | -985 | --- | Fehler beim Aushandeln des IPCP<br>interner Fehler |
| | BEARER PPP LINK FAILED | -984 | --- | PPP Gegenstelle reagiert nicht auf Echo-Request<br>interner Fehler |
| | BEARER PPP TERM REQ | -983 | --- | PPP Session wurde von der Gegenstelle beendet.<br>interner Fehler |

| Log-Eintrag | | Parameter | | Beschreibung |
|---|---|---|---|---|
| Code | Klartext | Code | Klartext | |
| GPRS-Fehler (2/2) | | | | |
| | BEARER CALL REFUSED | -982 | --- | eingehender Anruf abgelehnt<br>später erneut versuchen |
| | BEARER UNKNOWN | -981 | --- | später erneut versuchen |
| Fehler beim Aufbauen der GPRS-Verbindung | | | | |
| 1200 | BEARER BAD HDL | -980 | --- | invalid handle<br>interner Fehler |
| | BEARER BAD STATE | -979 | --- | Der Träger wurde nicht gestoppt.<br>interner Fehler |
| | BEARER DEV | -978 | --- | Fehler bei der Link-Layer Initialisierung<br>interner Fehler |
| | BEARER GPRS START UNKNOWN | -977 | --- | interner Fehler |
| Fehler beim Abbauen der GPRS-Verbindung | | | | |
| 1200 | BEARER GPRS STOP BAD HDL | -976 | --- | invalid handle<br>interner Fehler |
| | BEARER GPRS STOP UNKNOWN | -975 | --- | interner Fehler |
| | | | | |
| 1200 | NETLOCK FAILURE | -970 | --- | Fehler bei der Netzauswahl<br>• Überprüfen Sie, ob sich das Gerät im Versorgungsbereich befindet (www.microtronics.at/footprint). |

| Log-Eintrag | | Parameter | | Beschreibung |
|---|---|---|---|---|
| Code | Klartext | Code | Klartext | |
| 1200 | BAND SEL FAILED | -969 | --- | Es konnte weder auf dem GSM900/1800 noch auf dem GSM85071900 Band ein Netzwerk gefunden werden.<br>• Versuchen Sie die Antennenposition zu verbessern.<br>• Überprüfen Sie, ob sich das Gerät im Versorgungsbereich befindet (www.microtronics.at/footprint). |
| TCP Channel Fehler (1/2) | | | | |
| 1200 | CHANNEL ABORTED | -965 | --- | Es wird versucht auf einen/von einem nicht mehr verfügbaren TCP-Client zu schreiben/lesen.<br>später erneut versuchen |
| | CHANNEL CSTATE | -964 | --- | Der Channel ist nicht im "WIP_CSTATE_ READY" Zustand.<br>später erneut versuchen |
| | CHANNEL NOT SUPPORTED | -963 | --- | Die Option wird vom Channel nicht unterstützt.<br>später erneut versuchen |
| | CHANNEL OUT OF RANGE | -962 | --- | Der Options-Wert liegt außerhalb des zulässigen Bereichs.<br>später erneut versuchen |
| | CHANNEL MEMORY | -961 | --- | interner Fehler |
| | CHANNEL INTERNAL | -960 | --- | interner Fehler |
| | CHANNEL INVALID | -959 | --- | ungültige Option oder Parameter<br>interner Fehler |
| | TCP DNS FAILURE | -958 | --- | Der Name konnte nicht in eine IP-Adresse aufgelöst werden.<br>interner Fehler |
| | CHANNEL RESOURCES | -957 | --- | Es stehen keine weiteren TCP-Puffer mehr zur Verfügung.<br>später erneut versuchen |

| Log-Eintrag | | Parameter | | Beschreibung |
|---|---|---|---|---|
| Code | Klartext | Code | Klartext | |
| TCP Channel Fehler (2/2) | | | | |
| 1200 | CHANNEL PORT IN USE | -956 | --- | TCP-Server-Port wird bereits verwendet.<br>später erneut versuchen |
| | CHANNEL REFUSED | -955 | --- | Die TCP-Verbindung wurde vom Server abgelehnt.<br>später erneut versuchen |
| | CHANNEL HOST UNREACHABLE | -954 | --- | keine Route zum Host.<br>später erneut versuchen |
| | CHANNEL NETWORK UNREACHABLE | -953 | --- | kein Netz erreichbar<br>später erneut versuchen |
| | CHANNEL PIPE BROKEN | -952 | --- | TCP-Verbindung unterbrochen<br>später erneut versuchen |
| | CHANNEL TIMEOUT | -951 | --- | Timeout (DNS-Request, TCP-Verbindung, Ping-Response,..)<br>später erneut versuchen |
| | CHANNEL UNKNOWN | -950 | --- | interner Fehler |

6872
6872
6872

# Kapitel 16 Ersatzteile und Zubehör

## 16.1 rapid M2M-Module

| Beschreibung | Menge | Bestellnummer |
|---|---|---|
| rapidM2M Modul M11 | 1 | 300068 |
| rapidM2M Modul M120 | 1 | 300087 |

## 16.2 Antennen

| Beschreibung | Menge | Bestellnummer |
|---|---|---|
| Magnetfußantenne SMA-M 2,5m | 1 | 300085 |
| Antennenverlängerung SMA-M/SMA-F 2,5 m | 1 | 206.807 |

## 16.3 Versorgung

| Beschreibung | Menge | Bestellnummer |
|---|---|---|
| Netzgerät 24V 1A (U: 24V, I max.: 1A, P max.: 24W) | 1 | 213.814 |
| Netzgerät 12V 500mA (U: 12V, I max.: 0,5A, P max.: 6W) | 1 | 206.623 |
| Primärstecker EU für Ladegerät [1)] | 1 | 300027 |
| Primärstecker UK für Ladegerät [1)] | 1 | 300028 |
| Primärstecker US für Ladegerät [1)] | 1 | 300029 |
| Primärstecker AUS für Ladegerät [1)] | 1 | 300030 |

[1)] nicht geeignet für das im Standardlieferumfang des Developer Pakets(300144) enthaltene Steckernetzgerät.

## 16.4 Adapter

| Beschreibung | Menge | Bestellnummer |
|---|---|---|
| Nullmodemadapter 9pol. D-Sub female/male | 1 | 206.686 |

## 16.5 Sonstiges Zubehör

| Beschreibung | Menge | Bestellnummer |
|---|---|---|
| myDatanet Tool Pen | 1 | 206.646 |

6872
6872
6872

# Kapitel 17 Dokumentenhistorie

| Rev. | Datum | Änderungen |
|---|---|---|
| 01 | 22.02.2013 | Erste Version |
| 02 | 10.06.2014 | Hardware Version 1.2<br>*Der Jumper (J113) zur Auswahl der USB-Versorgung für das Modul wurde hinzugefügt. Dadurch können sowohl "M1x" als auch "M2x" Module verwendet werden.*<br>*Die Klemmleiste für die analogen Differenzeingänge und Konstantstromausgänge wurde um 2 Anschlüsse für die Analogmasse erweitert.*<br>*Der Buzzer (LS200) summt nun nicht mehr, wenn kein Script in das rapidM2M Modul M120 eingespielt ist.*<br>*Auf Pin 13 der Stiftleiste mit den geraden Pinnummern des M120 Moduls (J107 und J110) ist jetzt das "Reset"-Signal aufgelegt.*<br>*Der Jumper (J37) zum Einschalten des Wireless GSM Controllers am Modul wurde hinzugefügt. (Bei gesetztem Jumper J37 erübrigt sich das Drücken des Tasters S101.)*<br>Kapitel "Technische Daten" auf Seite 15<br>*Angabe der Speichergröße für das PAWN-Binary hinzugefügt*<br>Kapitel "Übersicht" auf Seite 67<br>*Beschreibung an Hw-Version 1.2 angepasst*<br>Kapitel "Jumper zur Auswahl der USB-Versorgung für das Modul (J113)" auf Seite 75<br>*Kapitel hinzugefügt*<br>Kapitel "Stiftleiste auf der die Pins mit den geraden Nummern des M120 Moduls aufgelegt sind (J107 und J110)" auf Seite 78<br>*Auf Pin 13 der Stiftleiste mit den geraden Pinnummern des M120 Moduls (J107 und J110) ist jetzt das "Reset"-Signal aufgelegt.*<br>Kapitel "Jumper zum Einschalten des Wireless GSM-Controllers am rapidM2M Modul M120 (J37)" auf Seite 81<br>*Kapitel hinzugefügt*<br>Kapitel "Technische Daten" auf Seite 99<br>*Angabe der Speichergröße für das PAWN-Binary hinzugefügt*<br>Kapitel "52pin Steckerleiste (X2)" auf Seite 102<br>*Detailansichten der Steckerleisten hinzugefügt*<br>Kapitel "Speicherorganisation" auf Seite 105<br>*Kapitel hinzugefügt.*<br>Kapitel "Technische Details zur Energieversorgung" auf Seite 107<br>*Tabelle mit Informationen zu benötigten Versorgungsspannungen und Stromaufnahmen hinzugefügt* |

| Rev. | Datum | Änderungen |
|---|---|---|
| 02 | 10.06.2014 | **Kapitel "ENPower-Signal" auf Seite 108**<br>*Kapitel hinzugefügt*<br>**Kapitel "Technische Details zu den Level Shiftern" auf Seite 110**<br>*Angaben zur Stromaufnahme der Levelshifter hinzugefügt*<br>**Kapitel "Typische Anwendung" auf Seite 114**<br>*Kapitel hinzugefügt*<br>**Kapitel "Cloud-Server" auf Seite 117**<br>*Sceenshots und Erklärungen an Serverversion V39.2 angepasst*<br>**Kapitel "Neu registrieren und anmelden" auf Seite 117**<br>*Anleitung an die aktelle Version der Homepage angepasst*<br>**Kapitel "Bereich "Statistik"" auf Seite 136**<br>*Kapitel hinzugefügt.*<br>**Kapitel "rapidM2M Toolset " auf Seite 141**<br>*Sceenshots und Erklärungen an Version 1.6 angepasst*<br>**Kapitel "Allgemein gültige Tabellenbezeichner" auf Seite 160**<br>*Kapitel hinzugefügt*<br>**Kapitel "Geräteabhängige Tabellenbezeichner" auf Seite 166**<br>*Kapitel hinzugefügt.*<br>**Kapitel "Steuerung" auf Seite 193**<br>*Beschreibung bezüglich des Hochladens von Binary-Files auf den Server hinzugefügt*<br>*Script-Templates werden von der Serverversion V39.2 nicht mehr unterstützt.*<br>**Kapitel "Konfiguration 0 - Konfiguration 1" auf Seite 194**<br>*Kapitel hinzugefügt*<br>**Kapitel "Pawn Script" auf Seite 197**<br>*Aufbau der Unterkapitel überarbeitet*<br>**Kapite "Allgemein" auf Seite 197**<br>*Beschreibung bezüglich des Hochladens von Binary-Files auf den Server hinzugefügt*<br>*Script-Templates werden von der Serverversion V39.2 nicht mehr unterstützt.*<br>**Kapitel "rapidM2M API" auf Seite 198**<br>*Angabe des Include-Files ist nur mehr für die String Funktionen notwendig.*<br>**Kapitel "Aufbau des HTTP-GET-Kommandos" auf Seite 170**<br>*Erklärung des optionalen Parameters "verbose_time" hinzugefügt*<br>**Kapitel "Datensatzverdichtung" auf Seite 176**<br>*Genauere Beschreibung der Methoden zur Summenbildung hinzugefügt.* |

| Rev. | Datum | Änderungen |
|---|---|---|
| 03 | 05.03.2015 | **Kapitel "Technische Daten" auf Seite 15**<br>*Angabe der Größen der Speicherbereiche (Datenspeicher, Konfigurationsspeicher und Registrierungsspeicher) passend zur Firmwareversion 00v014 aktualisiert*<br>**Kapitel "Allgemeine Produktinformationen" auf Seite 19**<br>*Informationen die Registrierungsspeicherblöcke betreffend hinzugefügt*<br>**Kapitel "Tutorial" auf Seite 23**<br>*Kapitel hinzugefügt*<br>**Kapitel "Empfohlene Vorgehensweise" auf Seite 58**<br>*Kapitel hinzugefügt*<br>**Kapitel "Vorgehensweise bei Verbindungsabbrüchen" auf Seite 60**<br>*Kapitel hinzugefügt*<br>**Kapitel "Errorhandling" auf Seite 62**<br>*Kapitel hinzugefügt*<br>**Kaptiel "Registrierungsspeicherblöcke" auf Seite 63**<br>*Kapitel hinzugefügt*<br>**Kapitel "File Transfer" auf Seite 64**<br>*Kapitel hinzugefügt*<br>**Kapitel "Bedeutung des SIM-Status" auf Seite 65**<br>*Kapitel hinzugefügt*<br>**Kapitel "RS232-Schnittstellen" auf Seite 71**<br>*Hinweis hinzugefügt, dass bei direkter Einspeisung der UART Signale über die Stiftleiste auf der die Pins mit den ungeraden Nummern des M120 Moduls aufgelegt sind, die RS232-Pegelwandler des Developer Boards ausgelötet werden müssen.*<br>*Das Hardware-Handeshake wird nun von der Firmware unterstützt.*<br>**Kapitel "Simulationsschaltungen" auf Seite 72**<br>*Korrektur der Vorgängerversion: Der Buzzer gibt einen Ton ab, wenn der GPIO1 auf "low" gesetzt wird.*<br>**Kapitel "Stiftleiste auf der die Pins mit den geraden Nummern des M120 Moduls aufgelegt sind (J107 und J110)" auf Seite 78**<br>*Korrektur der Vorgängerversion: An Pin 26 der Stiftleiste ist der GPIO6 aufgelegt.*<br>**Kaptiel "Stiftleiste auf der die Pins mit den ungeraden Nummern des M120 Moduls aufgelegt sind (J106 und J109)" auf Seite 76**<br>*Hinweis hinzugefügt, dass bei direkter Einspeisung der UART Signale über diese Stiftleiste die RS232-Pegelwandler des Developer Boards ausgelötet werden müssen.*<br>**Kapitel "Versorgung (J105 bzw. J108) " auf Seite 80**<br>*Hinweis auf die Verpolungsschutzdiode des Versorgungsspannungseingangs hinzugefügt* |

| Rev. | Datum | Änderungen |
|---|---|---|
| 03 | 05.03.2015 | **Kapitel "Technische Daten" auf Seite 99**<br>*Angabe der Größen der Speicherbereiche (Datenspeicher, Konfigurationsspeicher und Registrierungsspeicher) passend zur Firmwareversion 00v014 aktualisiert*<br>**Kapitel "Speicherorganisation" auf Seite 105**<br>*Informationen die Registrierungsspeicherblöcke betreffend hinzugefügt*<br>**Kapitel "UART" auf Seite 110**<br>*Genauere Beschreibung zur Zuordnung von UART des M120 Moduls und COM Anschluss des Developer Boards hinzugefügt*<br>**Kapitel "Interrupt Eingänge" auf Seite 114**<br>*Genauere Erklärung der Unterschiede zwischen GPIO- und Interrupt-Eingänge hinzugefügt*<br>**Kapitel "Typische Anwendung" auf Seite 114**<br>*Hinweis auf die Verpolungsschutzdiode des Versorgungsspannungseingangs hinzugefügt*<br>**Kaptiel "Messstellen" auf Seite 123**<br>*Erklärung der applikationsspezifischen Steuerelemente hinzugefügt*<br>**Kapitel "Anlegen einer neuen Messstelle" auf Seite 125**<br>*Erklärung zur Auswahl einer Applikations-Vorlage hinzugefügt*<br>**Kapitel "Dateiliste" auf Seite 127**<br>*Kapitel hinzugefügt*<br>**Kapitel "Bereich "Benutzer"" auf Seite 130**<br>*Erklärung zum Suchfeld hinzugefügt*<br>**Kapitel "Installation" auf Seite 143**<br>*Hinweis, dass nun 2 USB-Treiber installiert werden, hinzugefügt*<br>**Kapitel "Übersicht" auf Seite 146**<br>*Hinweis hinzugefügt, dass vor dem Zugriff per USB dem Modul eine Messstelle zugeordnet sein muss*<br>**Kapitel "Kateireiter "Script"" auf Seite 148**<br>*Beschreibung der Oberfläche an Version 1.12 angepasst*<br>**Kapitel "Karteireiter "Server"" auf Seite 153**<br>*Erklärung hinzugefügt, dass dieser Kateireiter nun zur Testseiter der REST-API weiterleitet*<br>**Kapitel "Karteireiter "FW"" auf Seite 155**<br>*OS Version des Moduls wird nun auch angezeigt.*<br>**Kapitel "Hot-Keys" auf Seite 156**<br>*Kapitel hinzugefügt* |

| Rev. | Datum | Änderungen |
|---|---|---|
| 03 | 05.03.2015 | **Kapitel "Aufbau der Datenbank" auf Seite 157**<br>*Auflistung der geräteabhängigen Tabellenbezeichner um die Messdatenkanäle ("histdata0" - "histdata9"), das Gerätelog, den Gerätestatus, die Lokalisierungsinformationen und die Registrierungsspeicherblöcke erweitert*<br>*Detaillierte Erklärung der Zuordnung der Tabellen zu Gerät bzw. Messstelle hinzugefügt*<br>**Kapitel "rm2mlog" auf Seite 166**<br>*Kapitel hinzugefügt*<br>**Kapitel "rm2mstate" auf Seite 166**<br>*Kapitel hinzugefügt*<br>**Kapitel "rm2mlocals" auf Seite 167**<br>*Kapitel hinzugefügt*<br>**Kapitel "rm2mpos" auf Seite 167**<br>*Liste der Funktionen, die die Daten dieser Tabelle erstellen können, ergänzt.*<br>**Kapitel "rm2m_ctrlscript" auf Seite 168**<br>*Erklärung des Feldes "script_error" hinzugefügt*<br>**Kapitel "reg_sys_otp" auf Seite 169**<br>*Kapitel hinzugefügt*<br>**Kapitel "reg_sys_flash" auf Seite 169**<br>*Kapitel hinzugefügt*<br>**Kapitel "reg_app_otp" auf Seite 169**<br>*Kapitel hinzugefügt*<br>**Kapitel "reg_app_flash" auf Seite 170**<br>*Kapitel hinzugefügt*<br>**Kapitel "Datensatzverdichtung" auf Seite 176**<br>*Hinweis hinzugefügt, dass bei aktivierter Datensatzverdichtung die Ausgabe als JSON-Dokuments nicht möglich ist*<br>**Kapitel "Ausgabeformate des HTTP-GET-Kommandos" auf Seite 181**<br>*Kapitel hinzugefügt*<br>**Kapitel "Übertragung der Daten an den Cloud-Server" auf Seite 183**<br>*Detaillierte Erklärung zur Verwendung des Kommandozeilen-Tools "Wget" hinzugefügt*<br>**Kapitel "Steuerung" auf Seite 193**<br>*Hinweis, dass dieser Konfigurationsabschnitt nur sichtbar ist, wenn die Messstelle nicht aus einer Applikations-Vorlage erstellt wurde, hinzugefügt*<br>**Kapitel "Messgerät" auf Seite 195**<br>*Erklärung des Felds "Hardware Version" hinzugefügt*<br>**Kapitel "Consolen Funktionen" auf Seite 201**<br>*Erklärung der Rückgabewerte ergänzt* |

| Rev. | Datum | Änderungen |
|---|---|---|
| 03 | 05.03.2015 | **Kapitel "rapidM2M Funktionen" auf Seite 202**<br>*Erklärung der Arrays mit symbolischen Indizes hinzugefügt*<br>*Erklärung der Konstanten hinzugefügt*<br>*Erklärung der Callback Funktionen hinzugefügt*<br>*Erklärungen der Funktionen "rM2M_GetId", "rM2M_GetTime" "rM2M_GetDate", passend zur Firmwareversion 00v014 aktualisiert*<br>*Erklärungen der Funktionen "rM2M_GetTimezoneOffset", "rM2M_Pack", "rM2M_CfgInit" ,"rM2M_CfgWrite", "rM2M_CfgFlush", "rM2M_RegGetString" ,"rM2M_RegSetString", "rM2M_RegGetValue", "rM2M_RegSetValue", "rM2M_RegOnChg", "rM2M_SmsInit", "rM2M_SmsClose", "rM2M_WriteLog", "rM2M_SetPos", "rM2M_SetPosNMEA", "rM2M_GetPos", "rM2M_GetGSMPos" und "rM2M_GSMGetInfo" hinzugefügt*<br>*Erklärung der Rückgabewerte ergänzt*<br>**Kapitel "File Transfer Funktionen" auf Seite 241**<br>*Kapitel hinzugefügt*<br>**Kapitel "Hilfsfunktionen" auf Seite 252**<br>*Erklärung der Arrays mit symbolischen Indizes hinzugefügt*<br>*Erklärung der Rückgabewerte ergänzt*<br>*Erklärung der Funktionen "CalcTable" und "CalcTableF" hinzugefügt*<br>**Kapitel "Pawn Script Fehlercodes" auf Seite 255**<br>*Erklärung des Fehlercodes "SCRIPT_ERR" hinzugefügt*<br>*Mögliche Ursachen für Fehlercodes "AMX_ERR_EXIT" und "AMX_ERR_MEMACCESS" hinzugefügt*<br>**Kapitel "Split-tag" auf Seite 284**<br>*Erklärung der Konfigurationsspeicherblöcke überarbeitet.*<br>**Kapitel "myDatanet Connector " auf Seite 283**<br>*Erklärung der Konfigurationsspeicherblöcke überarbeitet.*<br>**Kapitel "Datenstruktur" auf Seite 286**<br>*Erklärung der Attribute "name" hinsichtlich der Konfigurationsspeicherblöcke überarbeitet*<br>*Erklärung der Attribute "para1", "editmask", "view" und "edit" hinzugefügt*<br>**Kapitel "Fehlersuche und Behebung" auf Seite 293**<br>*Mögliche Fehlerbehebungen für den Fall, dass das Pawn Script nicht korrekt ausgeführt wird, der Buzzer einen Dauerton abgibt, oder an den GPIOs nicht der gewünschte Pegel ausgegeben wird*<br>**Kapitel "Log-Einträge und Fehlercodes " auf Seite 295**<br>*Erklärung der Log-Einträge "POWER ON", "LOG REFORMATFILE", "MODEM SIM RESET" und "SIM_STATE" hinzugefügt*<br>**Kapitel "Glossar" auf Seite 311**<br>*Kapitel hinzugefügt* |

# Kapitel 18 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.at/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.*

6872
6872
6872

# Kapitel 19 Kontaktinformationen

**Support & Service:**
Microtronics Engineering GmbH
Hauptstrasse 7
3244 Ruprechtshofen
Austria, Europe
Tel. +43 (0)2756 7718023
support@microtronics.at
www.microtronics.at

**Microtronics Engineering GmbH (Headquarter)**
Hauptstrasse 7
3244 Ruprechtshofen
Austria, Europe
Tel. +43 (0)2756 77180
Fax. +43 (0)2756 7718033
office@microtronics.at
www.microtronics.at

**Microtronics Engineering GmbH**
Windthorststrasse 5
63179 Obertshausen
Germany, Europe
office@microtronics.de
www.microtronics.de