

# User manual myDatalogC33x

Valid from:

- Firmware version: 01v024
- Server version: 50v007
- Hardware version: 1.2





# Chapter 1 Table of contents

<b>Cover</b> .....	<b>1</b>
<b>Chapter 1 Table of contents</b> .....	<b>3</b>
<b>Chapter 2 Declaration of Conformity</b> .....	<b>11</b>
2.1 myDatalogC33x WIFI/2G/3G/4G World .....	11
<b>Chapter 3 Specifications</b> .....	<b>13</b>
<b>Chapter 4 General specifications</b> .....	<b>17</b>
4.1 Translation .....	17
4.2 Copyright .....	17
4.3 General descriptive names .....	17
4.4 Safety instructions .....	17
4.4.1 Use of the hazard warnings .....	18
4.4.2 General safety instructions .....	18
4.4.3 Safety and preventative measures for handling mobile network modems .....	18
4.4.3.1 Safety and precautionary measures for the mobile network modem installation .....	19
4.4.3.2 Safety measures for installing the antenna .....	19
4.5 Overview .....	20
4.5.1 System architecture .....	21
4.5.2 Block diagram .....	22
4.6 Intended use .....	23
4.7 General product information .....	23
4.8 Device labelling .....	25
4.9 Installation of spare and wear parts .....	25
4.10 Storage of the product .....	26
4.11 Warranty .....	26
4.12 Disclaimer .....	27
4.13 Obligation of the operator .....	27
4.14 Personnel requirements .....	28
<b>Chapter 5 Functional principle</b> .....	<b>29</b>
5.1 Recommended procedure .....	31
5.1.1 Development of M2M/loT application .....	31
5.2 Functionality of the internal data memory .....	31

---

5.3 Memory organisation.....	33
5.4 Procedure in case of connection aborts.....	34
5.4.1 Connection abort in "online" mode.....	35
5.4.2 Connection abort during a Device Logic download.....	35
5.5 Timeout monitoring in online mode.....	35
5.6 Automatic selection of the GSM network.....	35
5.7 Determining the GSM/UMTS/LTE signal strength.....	36
5.8 Determining the GSM position data.....	36
5.9 Error handling.....	36
5.10 Registration memory blocks.....	37
5.10.1 REG_APP_OTP.....	37
5.11 File transfer.....	38
5.12 Meaning of the SIM state.....	39
<b>Chapter 6 Storage, delivery and transport.....</b>	<b>41</b>
6.1 Inspection of incoming deliveries.....	41
6.2 Scope of supply.....	41
6.3 Storage.....	41
6.4 Transport.....	41
6.5 Return.....	42
<b>Chapter 7 Installation.....</b>	<b>43</b>
7.1 Dimensions.....	43
7.2 Installing the myDatalogC33x.....	43
7.2.1 Top-hat rail assembly.....	44
7.2.2 Assembly in a control cabinet.....	45
7.3 Safety instructions for cabling.....	46
7.3.1 Information on preventing electrostatic discharges (ESD).....	46
7.4 Electrical installation.....	46
7.4.1 Connecting the sensors, actuators and power supply.....	46
7.4.1.1 Connection examples.....	49
7.4.2 Connection of the GSM antenna.....	50
7.4.3 Connecting the extension modules.....	51
7.4.3.1 CAN bus without branch lines.....	52
7.4.3.2 CAN bus with branch line.....	56

---

---

7.4.4 Technical details about the universal inputs.....	59
7.4.4.1 0/4...20mA mode.....	59
7.4.4.2 0...2V mode.....	60
7.4.4.3 0...10V mode.....	60
7.4.4.4 Standard digital modes (PWM, frequency, digital, counter).....	60
7.4.5 Technical details about the RS485 interface.....	60
7.4.6 Technical details about the CAN interface.....	61
7.4.7 Technical details about the RS232 interface.....	63
7.4.8 Technical details about the USB interface.....	64
7.4.9 Technical details about the outputs.....	65
7.4.9.1 Isolated switch contact (NO, CC).....	65
7.4.10 Technical details about the integrated rechargeable buffer battery.....	65
7.4.11 Technical details about the energy supply.....	67
7.4.12 Technical details about the system time.....	67
<b>Chapter 8 Initial Start-Up.....</b>	<b>69</b>
8.1 User information.....	69
8.2 Applicable documents.....	69
8.3 General principles.....	69
8.4 Commissioning the system.....	69
8.5 Testing communication with the device.....	70
<b>Chapter 9 User interfaces.....</b>	<b>73</b>
9.1 User interface on the myDatalogC33x.....	73
9.1.1 Operating elements.....	73
9.2 User interface on the myDatanet server.....	74
9.2.1 Site configuration.....	74
9.2.1.1 Site.....	74
9.2.1.2 Comments.....	74
9.2.1.3 Control.....	75
9.2.1.4 Configuration 0 - Configuration 9.....	75
9.2.1.5 Alarm settings.....	75
9.2.1.6 Basic settings.....	76
9.2.1.7 FTP export settings.....	76
9.2.2 Device configuration.....	77

---

9.2.2.1 Comments .....	77
9.2.2.2 Measurement instrument .....	77
9.2.2.3 GPRS.....	78
<b>Chapter 10 DeviceConfig.....</b>	<b>79</b>
10.1 General.....	79
10.2 Prerequisites.....	79
10.3 Functional principle.....	80
10.4 Installation.....	81
10.5 Menu of the DeviceConfig.....	83
10.5.1 Settings.....	83
10.5.1.1 Options.....	83
10.6 Connecting a Device via USB.....	85
10.7 "Log" tab.....	87
10.8 "Firmware" tab.....	89
<b>Chapter 11 myDatanet server.....</b>	<b>91</b>
11.1 Overview.....	91
11.1.1 Explanation of the symbols.....	91
11.2 "Customer" area.....	92
11.3 "Sites / Applications" area at customer level.....	94
11.3.1 Reports.....	95
11.3.2 Map view.....	95
11.4 Recommended procedure.....	95
11.4.1 Creating the site.....	95
<b>Chapter 12 rapidM2M Studio.....</b>	<b>99</b>
12.1 General.....	99
12.2 Prerequisites.....	100
12.3 Project dashboard.....	101
12.4 CODEbed.....	102
12.5 TESTbed.....	103
<b>Chapter 13 Device Logic.....</b>	<b>105</b>
13.1 General.....	105
13.1.1 Direct entry of a device logic.....	105
13.1.2 Uploading a binary file.....	105

---

13.1.3 Using the CODEbed of the web-based development environment rapidM2M Studio.....	105
13.2 Compiler options.....	106
13.3 Device API.....	106
13.3.1 Constants.....	106
13.3.2 Timer, date & time.....	107
13.3.2.1 Arrays with symbolic indices.....	107
13.3.2.2 Constants.....	107
13.3.2.3 Functions.....	107
13.3.3 Uplink.....	112
13.3.3.1 Arrays with symbolic indices.....	112
13.3.3.2 Constants.....	113
13.3.3.3 Callback functions.....	117
13.3.3.4 Functions.....	118
13.3.4 Encoding.....	128
13.3.4.1 Constants.....	128
13.3.4.2 Functions.....	129
13.3.5 Registry.....	134
13.3.5.1 Constants.....	134
13.3.5.2 Callback functions.....	135
13.3.5.3 Functions.....	135
13.3.6 Position.....	139
13.3.6.1 Arrays with symbolic indices.....	139
13.3.6.2 Constants.....	141
13.3.6.3 Functions.....	143
13.3.7 Math.....	151
13.3.8 64 bit signed integer.....	154
13.3.9 Char & String.....	157
13.3.10 CRC & hash.....	165
13.3.10.1 Arrays with symbolic indices.....	165
13.3.10.2 Functions.....	165
13.3.11 Various.....	166
13.3.11.1 Arrays with symbolic indices.....	166
13.3.11.2 Constants.....	167

---

---

13.3.11.3 Functions .....	167
13.3.12 Console .....	173
13.3.13 SMS .....	175
13.3.13.1 Callback functions .....	175
13.3.13.2 Functions .....	175
13.3.14 File transfer .....	176
13.3.14.1 Arrays with symbolic indices .....	176
13.3.14.2 Constants .....	176
13.3.14.3 Callback functions .....	176
13.3.14.4 Functions .....	178
13.4 Device Logic error codes .....	182
13.5 Syntax .....	186
13.5.1 General syntax .....	186
13.5.1.1 Format .....	186
13.5.1.2 Optional semicolons .....	186
13.5.1.3 Comments .....	186
13.5.1.4 Identifier .....	186
13.5.1.5 Reserved keywords .....	186
13.5.1.6 Numerical constants .....	187
13.5.1.6.1 Numerical integer constants .....	187
13.5.1.6.2 Numerical floating-point constants .....	187
13.5.2 Variables .....	187
13.5.2.1 Declaration .....	187
13.5.2.2 Local declaration .....	187
13.5.2.3 Global declaration .....	187
13.5.2.4 Static local declaration .....	188
13.5.2.5 Static global declaration .....	188
13.5.2.6 Floating point values .....	188
13.5.3 Constant variables .....	188
13.5.4 Array variables .....	188
13.5.4.1 One-dimensional arrays .....	188
13.5.4.2 Initialisation .....	189
13.5.4.3 Progressive initialisation for arrays .....	189

---



---

13.5.4.4 Multi-dimensional arrays.....	189
13.5.4.5 Arrays and the "sizeof" operator.....	190
13.5.5 Operators and expressions.....	191
13.5.5.1 Notational conventions.....	191
13.5.5.2 Expressions.....	191
13.5.5.3 Arithmetic.....	191
13.5.5.4 Bit manipulation.....	192
13.5.5.5 Assignment.....	192
13.5.5.6 Comparative operators.....	193
13.5.5.7 Boolean.....	193
13.5.5.8 Other.....	194
13.5.5.9 Priority of the operators.....	194
13.5.6 Statements.....	195
13.5.6.1 Statement label.....	195
13.5.6.2 Composite statements.....	196
13.5.6.3 Expression statement.....	196
13.5.6.4 Empty statement.....	196
13.5.6.5 Assert expression.....	196
13.5.6.6 Break.....	197
13.5.6.7 Continue.....	197
13.5.6.8 Do statement while (expression).....	198
13.5.6.9 Exit expression.....	198
13.5.6.10 For (expression 1; expression 2; expression 3) statement.....	198
13.5.6.11 Goto label.....	199
13.5.6.12 If (expression) statement 1 else statement 2.....	199
13.5.6.13 Return expression.....	199
13.5.6.14 switch (expression) {case list}.....	199
13.5.6.15 While (expression) statement.....	200
13.5.7 Functions.....	201
13.5.7.1 Function arguments ("call-by-value" versus "call-by-reference").....	201
13.5.7.2 Named parameters versus fixed parameters.....	203
13.5.7.3 Standard values of function arguments.....	203
13.6 Differences to C.....	204

---

---

<b>Chapter 14 Data Descriptor</b> .....	<b>207</b>
14.1 Data structure.....	207
14.1.1 Division of a structured measurement data channel into individual data fields.....	208
14.1.2 Division of a configuration memory block into individual data fields.....	209
14.1.3 Division of the aloga data into individual data fields.....	210
14.1.4 Attributes of the field definition.....	210
14.2 Example.....	215
14.3 Special values of the data types.....	217
<b>Chapter 15 API</b> .....	<b>219</b>
15.1 Backend API.....	219
15.2 rapidM2M Playground.....	219
15.2.1 Overview.....	220
<b>Chapter 16 Maintenance</b> .....	<b>221</b>
16.1 General maintenance.....	221
16.2 Fuse replacement.....	221
<b>Chapter 17 Removal/disposal</b> .....	<b>223</b>
<b>Chapter 18 Troubleshooting and repair</b> .....	<b>225</b>
18.1 General problems.....	225
18.2 Log entries and error codes.....	227
18.2.1 Modem error.....	232
18.3 Evaluating the device log.....	233
18.3.1 Evaluating the device log on the myDatanet server.....	233
18.3.2 Evaluating the device log using DeviceConfig.....	234
<b>Chapter 19 Spare parts and accessories</b> .....	<b>235</b>
19.1 Assembly sets.....	235
19.2 Antennas.....	235
19.3 Power supply.....	235
19.4 Adapter.....	235
19.5 Extension modules.....	236
19.6 Other accessories.....	236
<b>Chapter 20 Document history</b> .....	<b>237</b>
<b>Chapter 21 Glossary</b> .....	<b>245</b>
<b>Chapter 22 Contact information</b> .....	<b>247</b>

---

# Chapter 2 Declaration of Conformity

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

### EU-Konformitätserklärung

#### EU Declaration of Conformity / Déclaration de conformité UE

**Produktbezeichnung:** Stationäres, kompaktes, frei programmierbares Gerät zur Erfassung, Verarbeitung und Übertragung von Signalen und Geräteinformationen  
 Product: Stationäres, kompaktes, frei programmierbares Gerät zur Erfassung, Verarbeitung und Übertragung von Signalen und Geräteinformationen  
 Désignation du produit: Stationäres, kompaktes, frei programmierbares Gerät zur Erfassung, Verarbeitung und Übertragung von Signalen und Geräteinformationen

**Type :** myDatalogC33x 2G/3G/4G World **Gültig ab:** Rev. 1.2  
 Type code: Valid from:  
 Type: Valide à partir de:



**Hersteller:** Microtronics Engineering GmbH  
 Manufacturer: Hauptstrasse 7  
 Fabricant: A-3244 Ruprechtshofen

Das bezeichnete Produkt stimmt mit den folgenden Europäischen Richtlinien überein. The designated product is in conformity with the following european directives. Le produit décrit est conforme aux directives européennes suivantes.			
		Europäische Norm	
(2014/30/EU)	EMC Directive	EN61000-4-3 (testlevel 3)	
		EN55032 (class A)	
		EN61326-1	
(2014/35/EU)	LVD Directive	EN61010-1	
(2014/53/EU)	RED Directive	Safety & Health 3.1a	EN62368-1 EN62368-1+A11:2017 EN62311 EN62479
		EMC 3.1b	EN301489-1 V2.2.3 EN301489-52 V1.1.0
		Radio spectrum efficiency 3.2	EN301511 V12.5.1 EN301908-1 V13.1.1 EN301908-2 V13.1.1 EN301908-13 V13.1.1 EN300328 V2.1.1
(2015/863/EU)	RoHS Directive	Prevention 4.1	EN IEC 63000

Ruprechtshofen, den 22.09.2023

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

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



## Chapter 3 Specifications

Voltagesupply	9...32VDC (+/-10%)  Additional information is provided in "Technical details about the energy supply" on page 67.
Power consumption	typ. 5W (without sensors) max. 9W (without sensors)
Integrated rechargeable buffer battery	Li-Po rechargeable battery with 500mAh for: <ul style="list-style-type: none"> <li>• Reaction by application to power supply failure</li> <li>• Disconnection from mobile network in the event of a power supply failure</li> </ul> Additional information is provided in "Technical details about the integrated rechargeable buffer battery" on page 65.
Housing	Material: Lexan/Noryl Weight: 190g Degree of protection: IP20 / IP40 (connection area/operating area) Dimensions (WHD): Dimensions (WHD): 70 x 92 x 63mm (without antenna)
Operating temperature	-20...+60°C
Air humidity	15...90%rH non-condensing
Storage temperature	-40...+85°C
Charging temperature (rechargeable buffer battery)	0 ...+45°C
Display	RGB LED (freely useable, controlled by the device logic)
Operation	MDN Button (freely useable, evaluated by the device logic) Button to trigger a reset
System time	Hardware real-time clock with its own buffer battery (service life > 10 years) and automatic time synchronisation with the server.  Additional information is provided in "Technical details about the system time" on page 67.
Antenna connector	1 x SMA-F for mobile network 1 x SMA-F for WiFi

<p>Universal inputs</p>	<p>3 x analogue or digital</p> <p>Modes:</p> <ul style="list-style-type: none"> <li>• 0...20mA: Resolution 6,36µA, max. 23,96mA, load 96Ω</li> <li>• 4...20mA: Resolution 6,36µA, max. 23,96mA, load 96Ω</li> <li>• 0...2V: Resolution 610µV, max. 2,5V, load 10k086</li> <li>• 0...10V: Resolution 7,97mV , max. 32V, load 4k7</li> <li>• PWM: 1...99%, max. 100Hz, min. impulse length 1ms, max. 32V, load 4k7</li> <li>• Frequency: 1...1000Hz, max. 32V, 4k7</li> <li>• Digital: low &lt;0,99V, high &gt;2,31V, max. 32V, load 4k7</li> <li>• Counter: min. pulse length 1ms, max. 32V, load 4k7</li> </ul> <p>Additional information is provided in "Technical details about the universal inputs" on page 59.</p>
<p>Serial interface</p>	<p>1 x RS485 (2-wire)</p> <ul style="list-style-type: none"> <li>• Baud rate: 2400-115200</li> <li>• Stop bits: 1, 2</li> <li>• Parity: N, E, O</li> <li>• Data bits: 7, 8</li> <li>• Load resistance: Off, 120Ω</li> <li>• Pull up at RS485 B: Off, 390Ω</li> <li>• Pull down at RS485 A: Off, 390Ω</li> </ul> <p>1 x CAN / CAN FD</p> <ul style="list-style-type: none"> <li>• max. data transfer rate: <ul style="list-style-type: none"> <li>• CAN: 1Mbit/s</li> <li>• CAN FD: 8Mbit/s</li> </ul> </li> <li>• supported specification: <ul style="list-style-type: none"> <li>• CAN: V2.0B</li> <li>• CAN-FD: V1.0</li> </ul> </li> <li>• no galvanic isolation</li> <li>• Load resistance: Off, 120Ω , 2k</li> </ul> <p>1 x RS232 (4-wire)</p> <ul style="list-style-type: none"> <li>• Baud rate: 2400-115200</li> <li>• Stop bits: 1, 2</li> <li>• Parity: N, E, O</li> <li>• Data bits: 7, 8</li> <li>• Flow control: Off, RTS/CTS</li> </ul> <p>Additional information is provided in "Technical details about the RS485 interface" on page 60, "Technical details about the CAN interface" on page 61 and "Technical details about the RS232 interface" on page 63.</p>

Outputs	<p>2 x isolated switch contact</p> <ul style="list-style-type: none"> <li>• Galvanically isolated</li> <li>• Parallel connection of an optoswitch and relay</li> </ul> <p>Relay (high switching currents)</p> <ul style="list-style-type: none"> <li>• <math>U_{max}</math>: 32V</li> <li>• <math>I_{max}</math>: 2A</li> </ul> <p>Optoswitch (high switching frequencies)</p> <ul style="list-style-type: none"> <li>• <math>I_{max}</math>: 130mA</li> <li>• <math>U_{max}</math>: 32V</li> <li>• <math>R_{on}</math>: 35<math>\Omega</math></li> <li>• <math>f_{max}</math>: 1000Hz</li> </ul> <p>Additional information is provided in "Technical details about the outputs" on page 65.</p>
USB interface	<p>1 x mini-B USB 2.0 slave for the connection to a PC. The DeviceConfig configuration program must be installed on the PC or the web-based development environment rapidM2M Studio must be used to enable communication with the myDatalogC33x .</p> <p>Additional information is provided in "Technical details about the USB interface" on page 64.</p>
Data memory	<p>3 MB internal flash memory.</p> <p>The size of the data records is variable (max. 1024 Byte ) and is determined by the device logic created by the user. The system-related overhead is 11 Byte per data record.</p> <p>Additional information is provided in "Functionality of the internal data memory" on page 31.</p>
Configuration memory	<p>10 independent blocks each with 4000 Bytes</p>
Registration memory	<p>Flash: 4 blocks each with 1kB and pre-defined purposes for storing device-specific data</p> <p>RAM: 1 optional block with max. 1kB for storing application-specific data</p> <p>Additional information is provided in "Registration memory blocks" on page 37.</p>
Memory for the device logic binary	<p>8MB (uncompressed size)</p> <p>Additional information is provided in "Memory organisation" on page 33.</p>

Data transmission	<p>2G/3G/4G World (myDatalogC33x WIFI/2G/3G/4G World ):</p> <ul style="list-style-type: none"> <li>• 2G GPRS 900MHz / 1800MHz</li> <li>• 2G GPRS 850MHz / 1900MHz</li> <li>• UMTS B1, B2, B5, B8</li> <li>• LTE FDD B1, B2, B3, B4, B5, B7, B8, B12, B13, B18, B19, B20, B26, B28</li> <li>• LTE TDD B38, B39, B40, B41</li> </ul> <p>WiFi: <sup>1)</sup></p> <ul style="list-style-type: none"> <li>• Single band 2,4 GHz</li> <li>• IEEE 802.11 b/g/n</li> <li>• Channel bandwidth: 20MHz</li> <li>• Supported channels: 1-14</li> </ul> <p>Network interface:</p> <ul style="list-style-type: none"> <li>• 10/100 Ethernet</li> </ul>
SIM	The myDatalogC33x is equipped with an integrated SIM chip.

<sup>1)</sup> can also be used for communication with other devices and sensors (as long as not used as uplink)



# Chapter 4 General specifications

The information in this manual has been compiled with great care and to the best of our knowledge. The manufacturer, however, assumes no liability for any incorrect specifications that may be provided in this manual. The manufacturer is not responsible for direct, indirect, accidental or consequential damages which arise from errors or omissions in this manual even if advised of the possibility of such damages. In the interest of continuous product development, the manufacturer reserves the right to make improvements to this manual and the products described in it at any time and without prior notification or obligation.

**Note:** *The specifications in this manual are valid as of the versions listed on the front page. Revised versions of this manual, as well as software and driver updates are available in the service area of the myDatanet server.*

## 4.1 Translation

For deliveries to countries in the European Economic Area, the manual must be translated into the language of the respective country. If there are any discrepancies in the translated text, the original manual (German) must be referenced or the manufacturer contacted for clarification.

## 4.2 Copyright

The copying and distribution of this document as well as the utilisation and communication of its contents to others without express authorisation is prohibited. Contraventions are liable to compensation. All rights reserved.

## 4.3 General descriptive names

The use of general descriptive names, trade names, trademarks and the like in this manual does not entitle the reader to assume they may be used freely by everyone. They are often protected registered trademarks even if not marked as such.

## 4.4 Safety instructions

For the connection, commissioning and operation of the myDatalogC33x, the following information and higher legal regulations of the country (e.g. ÖVE), such as valid EX regulations as well as the applicable safety and accident prevention regulations for the respective application case must be observed.

Read this manual completely before unpacking, setting up or operating this device. Observe all hazard, danger and warning information. Non-observance can lead to serious injuries to the operator and/or damage to the device.

Ensure that the safety equipment of this measurement instrument is not impaired. Install and use the measurement system only in the manner and method described in this manual.


**Important note:** *The product is not approved for use outdoors as it is not protected from penetrating moisture and only provides minimal protection against the ingress of dust.*

---

#### 4.4.1 Use of the hazard warnings

 **DANGER:**  
*Indicates a potential or threatening hazardous situation that will result in death or serious injuries if not avoided.*

 **WARNING:**  
*Indicates a potential or threatening hazardous situation that can result in death or serious injuries if not avoided.*

 **CAUTION:**  
*Indicates a potential hazardous situation that can result in minor or moderate injuries or damage to this instrument.*


*Important note:* Indicates a situation that can result in damages to this instrument if it is not avoided. Information that must be particularly emphasised.


*Note:* Indicates a situation that does not result in any injury to persons.

*Note:* Information that supplements the specifications in the main text.

#### 4.4.2 General safety instructions

 **WARNING:**  
*Hazardous electric voltage can cause electric shock or burns. Always switch off all of the used power supplies for the device before installing it, completing any maintenance work or resolving any faults.*

 **WARNING:**  
*Ensure that the device is fully deactivated and cannot activate automatically when sending/returning it as air freight. Information on this is provided in chapter "Storage of the product" on page 26. If you have any unanswered questions, contact the manufacturer (see "Contact information" on page 247).*

 **WARNING:**  
*Never use this device in areas where the use of wireless equipment is prohibited. The device must not be used in hospitals and/or in the vicinity of medical equipment, such as heart pacemakers or hearing aids, as their functionality could be compromised by the mobile network modem contained in the device.*

 **WARNING:**  
*Never use this device in potentially explosive atmospheres and in the vicinity of highly combustible areas (fuel stations, storage areas for combustible material, chemical plants and detonation sites) or in the vicinity of flammable gases, vapours or dust.*

#### 4.4.3 Safety and preventative measures for handling mobile network modems

The following safety and preventative measures must be observed during all phases of installation, operation, maintenance or repair of a mobile network modem. The manufacturer is not liable if the customer disregards these preventative measures.

 **CAUTION:**  
*The mobile radio connection must not be used in hazardous environments.*

No guarantee of any kind, whether implicit or explicit, is given by the manufacturer and its suppliers for the use with high risk activities.

In addition to the following safety considerations, all directives of the country in which the device is installed must be complied with.

**Important note:** *No liability shall be assumed at any time and under no circumstances for connections via a mobile network modem for which wireless signals and networks are utilized. The mobile network modem must be switched on and operated in an area where sufficient signal strength is present.*

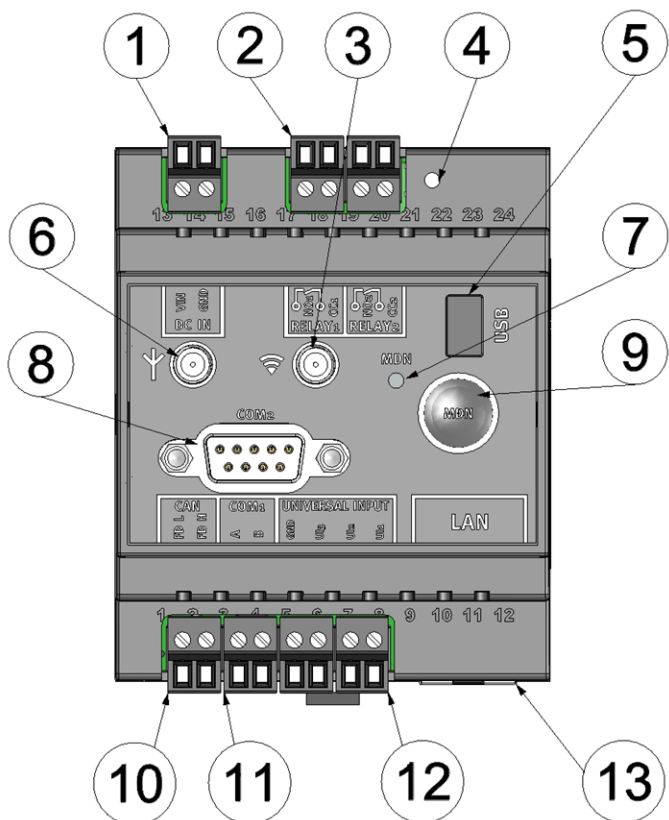
#### **4.4.3.1 Safety and precautionary measures for the mobile network modem installation**

- This device must only be installed by a trained technician who applies the recognised installation practices for a radio frequency transmitter including the correct grounding of external antennas.
- The device must not be operated in hospitals and/or in the vicinity of medical equipment such as heart pacemakers or hearing aids.
- The device must not be operated in highly flammable areas such as petrol filling stations, fuel storage sites, chemical factories and explosion sites.
- The device must not be operated in the vicinity of flammable gases, vapours or dusts.
- The device must not be subjected to strong vibrations or impacts.
- The mobile network modem can cause interferences if it is located in the vicinity of television sets, radios or computers.
- Do not open the mobile network modem. Any modification to the device is prohibited and will result in the operating licence being revoked.
- The use of GSM services (SMS messages/data communication/GPRS, etc.) may incur additional costs. The user is solely responsible for any resulting damages and costs.
- Do not install the device in any other way to the one described in the operating instructions. Improper use will invalidate the warranty.

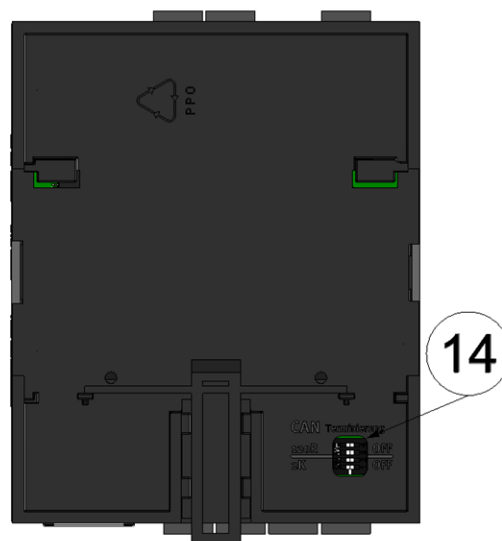
#### **4.4.3.2 Safety measures for installing the antenna**

- Only use antennas that are recommended or supplied by the manufacturer.
- The antenna must be installed at a distance of at least 20 cm from individuals.
- The antenna must not be extended outside protected buildings and must be protected against lightning strikes.
- The voltage supply must be switched off before replacing the antenna.

## 4.5 Overview



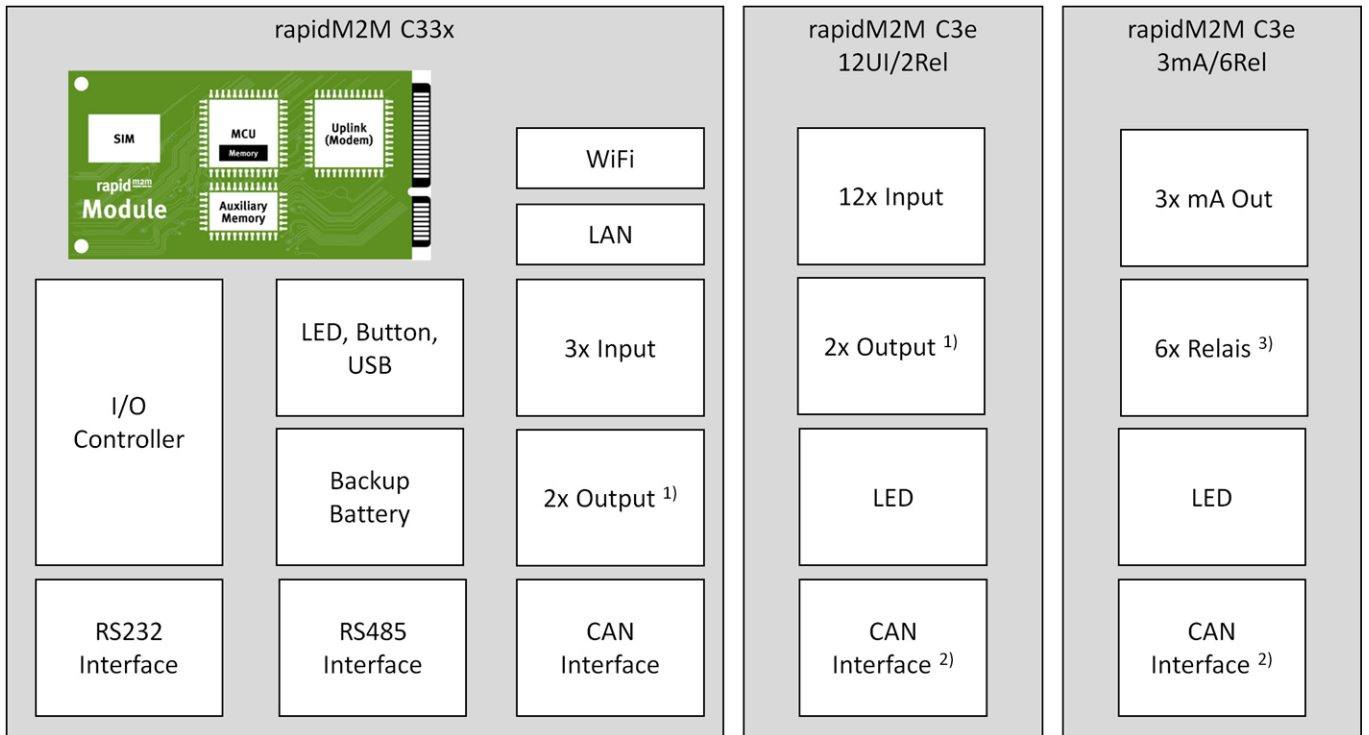
Front of the myDatalogC33x



Rear of the myDatalogC33x

<b>1</b> Supply (V IN, GND)	<b>8</b> Com2 (RS232)
<b>2</b> Relay 1-2	<b>9</b> MDN button (freely useable, evaluated by the device logic)
<b>3</b> Connector for the WiFi antenna	<b>10</b> CAN / CAN FD
<b>4</b> Reset button	<b>11</b> Com1 (RS485)
<b>5</b> Mini-B USB (only for debugging and device logic update)	<b>12</b> Universal input 1-3 (incl. GND)
<b>6</b> Connector for the mobile network antenna	<b>13</b> LAN interface
<b>7</b> RGB LED (freely useable, controlled by the device logic)	<b>14</b> Dip switch for activating/deactivating the load resistances for the CAN interface

## 4.5.1 System architecture



System architecture of the myDatalogC33x and the extension modules

- 1) Parallel connection of a relay (high switching currents) and an optoswitch (high switching frequencies)
- 2) Exclusively intended for communication with themyDatalogC33x
- 3) 3 of the relays are combined into a group with a common root



## 4.6 Intended use

The stationary, compact, freely programmable measurement instrument is designed for determining, processing and transferring measurement data acquired via various industrial interfaces. The device requires a continuous power supply. The measured and recorded data is stored on a non-volatile memory medium. This stored data is sent via the mobile network, WiFi or LAN to a central server for further processing. The device is equipped with an integrated SIM chip for establishing a mobile connection. The maximum permissible limit values specified in chapter "Specifications" on page 13 must be observed. The manufacturer shall not be liable for any operational cases that deviate from these limit values and have not been approved by the manufacturer in writing.

**Note:** *This device is exclusively intended to be used for the purposes as described before. Any other use or use beyond what is specified or a modification of the device shall be deemed to be not for the intended purpose and is not permitted without the express written consent of the manufacturer. The manufacturer shall not be held liable for any damages that may result from such unauthorised use or modification. The operator alone bears the associated risk.*

**Note:** *The manufacturer is not liable for data loss of any kind.*

**Note:** *The integrated SIM chip provides a mobile communications connection to a variety of international service providers. In order to be able to utilise all functions of the device, you must ensure that the device is located in the service area of one of these service providers. You can find a list of all supported countries and associated service providers under [www.microtronics.com/footprint](http://www.microtronics.com/footprint). A Managed Service contract with Microtronics Engineering GmbH is required for use of the mobile data transmission (see [www.microtronics.com/managedservice](http://www.microtronics.com/managedservice)). This includes the provisioning of the mobile communications connection via the network of the service provider included in the above-mentioned list.*

## 4.7 General product information

The device is a compact, stationary, freely programmable device that can be used for a variety of control and regulating tasks in addition to determining, processing and transferring measurement data.

The following interfaces are available for recording measurement data:

- 3 x universal inputs that can be operated in various analogue and digital modes
- 1 x RS485 interface
- 1 x CAN / CAN FD interface (not galvanically isolated)
- 1 x RS232 interface
- WiFi interface (as long as it is not used as uplink)

In addition, 2 isolated switch contacts are also available for the output of regulating and control commands. The isolated switch contacts are the parallel connection of a relay (high switching currents) and an optoswitch (high switching frequencies). In the event of a power failure the integrated buffer battery allows an appropriate application-specific response to be implemented or enables correct deregistration from the mobile radio network. Thanks to the hardware real-time clock with its own buffer battery the system time still continues to run even when the device is switched off, which means a valid time base is immediately available following recommissioning.

The user can create their own application via the rapidM2M Studio (see "rapidM2M Studio" on page 99). During development the part of the application that needs to be installed on the device (i.e. the device logic) is loaded into the myDatalogC33x via the USB interface. For applications that are provided via the rapidM2M

---

Store installation of the device logic is performed via the mobile network , WiFi or LAN connection in the course of connecting the site with the myDatalogC33x . The device logic enables access to the serial interfaces (RS232, CAN / CAN FD and RS485), thus providing the user with the option to connect to almost all of the devices and sensors that are compatible with these interfaces and to implement the corresponding communication protocols.

The myDatalogC33x provides the user with a memory area for their data (3 MB ) as well as 10 independent memory blocks each with 4000 Bytes for the configuration data. In addition to the 4 registration memory blocks each with 1kB , that are saved in the flash, the myDatalogC33x has another one that can optionally be initialised via the "rM2M\_RegInit()" function and that is saved in the RAM. Its size can be specified during initialisation, although it is limited to a maximum of 1kB . The registration memory blocks are assigned to predefined purposes and are designed for storing device-specific data (see "Registration memory blocks" on page 37).

Recorded data can be sent via the mobile network, WiFi or LAN to a central myDatamet server for further processing. The device is equipped with an integrated SIM chip for establishing a mobile connection. The user holds the responsibility for initialising the connection to the central myDatamet server(see "rM2M\_TxStart()"). However, the system automatically synchronises the configuration, registration and measurement data with the server.



## 4.8 Device labelling

The specifications in this user manual apply exclusively to the myDatalogC33x device type. The type plate is located on the right side of the device and contains the following specifications:

- Type designation
- Serial number
- MAC address of the LAN interface
- MAC address of the WiFi interface
- Item number
- Voltage supply specifications
- Week and year of production
- Country list profile of the SIM chip
- Environmental conditions during operation
- Protection class
- Hardware revision
- Name and address of the manufacturer
- Logo for the EU WEEE Directive
- CE marking

The correct specification of the type designation and serial number is important for all queries and spare part orders. Only then can we process requests promptly and properly.



Type plate myDatalogC33x



**Note:** This symbol indicates the country list profile (see [www.microtronics.com/footprint](http://www.microtronics.com/footprint)) of the SIM chip installed in the device.

**Note:** These operating instructions are part of the device and must be available to the user at all times. The safety instructions contained therein must be observed.



**WARNING:**

**It is strictly prohibited to disable the safety equipment or modify its mode of operation.**

## 4.9 Installation of spare and wear parts

Be advised that spare and accessory parts that have not been supplied by the manufacturer have also not been inspected or approved by the manufacturer. The installation and/or use of such products can possibly have a negative impact on the specified constructional properties of the device. The manufacturer shall not be liable for any damages that arise from the use of non-original parts and non-original accessory parts.

---

## 4.10 Storage of the product

To safeguard the myDatalogC33x, ensure that all relevant data was transferred to the myDatanet server. If necessary, initiate a transmission directly on the device using the MDN-button, if you have included this in your device logic, and then check again to see whether all of the relevant data has now been transferred. If your device does not include the option to initiate the transmission of temporarily stored measurement data, you may have to wait until the next scheduled data transmission for all of the data to be sent to the myDatanet server. This particularly applies to the "interval" connection type (see "rM2M\_TxSetMode()"). If the "Interval & wakeup" connection type has been selected, you can initiate the transmission via the myDatanet server. Then check again to make sure all of the relevant data has been transferred. With the "Online" connection type, the determined measurement data is immediately transferred to the myDatanet server. The data on the server is always up-to-date and the device can be switched off at any time. Then disconnect the device from the supply voltage. If possible, switch off the supply voltage before disconnecting the cables from the VIN and GND terminals (see "Connecting the sensors, actuators and power supply" on page 46). The remaining cables and the antenna can then be removed. Ensure that the myDatalogC33x is completely deactivated before you store it in the original packaging. To do so press the reset button directly on the device, unless you have provided a routine in your device logic for controlled shut-down of the system following disconnection of the supply voltage.

The configuration and most recently determined data are retained. The system time also continues to run thanks to the hardware real-time clock equipped with its own buffer battery. This means that a valid time basis is available immediately when recommissioning (see "Technical details about the system time" on page 67).

## 4.11 Warranty

The device has been functionally tested before delivery. If it is used as intended (see "Intended use" on page 23) and the operating instructions, the applicable documents (see "Applicable documents" on page 69) and the safety notes and instructions contained therein, are observed, no functional restrictions are to be expected and perfect operation should be possible.

**Note:** Please also note in this regard the next chapter "Disclaimer" on page 27.

**Note: Limitation of warranty**

*In the event of non-compliance with the safety instructions and instructions in this document, the manufacturer reserves the right to limit the warranty.*

## 4.12 Disclaimer

The manufacturer assumes no liability

- for damages owing to a **change** of this document. The manufacturer reserves the right to change the contents of this document and this disclaimer at any time and without any notice.
- for damages to persons or objects resulting from **failure to comply** with applicable **regulations**. For connection, commissioning and operation of the devices/sensors all available information and higher local legal regulations (e.g. in Austria ÖVE guidelines) such as applicable Ex regulations as well as safety requirements and regulations in order to avoid accidents shall be adhered to.
- for damages to persons or objects resulting from **improper use**. For safety and warranty reasons, all internal work on the instruments beyond from that involved in normal installation and connection, must be carried out only by qualified Microtronics personnel or persons or companies authorised by Microtronics .
- for damages to persons or objects resulting from the use of instruments in technically **imperfect** condition.
- for damages to persons or objects resulting from the use of instruments **not in accordance with the requirements**.
- for damages to persons or objects resulting from **failure to comply** with **safety information** contained within this instruction manual.
- for missing or incorrect measurement values or resulting consequential damages due to **improper installation**.

## 4.13 Obligation of the operator



**WARNING:**

*In the EEA (European Economic Area), the national implementation of the framework directive (89/391/EEC) as well as the associated specific directives and from these in particular, the directive (2009/104/EC) about the minimum safety and health requirements for use of work equipment by workers at work, each in their respective version are to be complied with.*

The operator must obtain the local operating licence and the associated documents.

In addition, the operator must comply with the local legal requirements for

- the safety of the personnel (accident prevention measures),
- the safety of the equipment (protective equipment and maintenance),
- the product disposal (waste disposal law),
- the material disposal (waste disposal law),
- the cleaning (cleaning agents and disposal) and
- the environmental protection amendments.

Before commissioning, the operator must ensure that the installation and commissioning – provided these were performed by the operator himself – are in compliance with the local regulations.

---

## 4.14 Personnel requirements

Installation, commissioning and maintenance may only be completed by personnel who meet the following conditions:

- Qualified specialist personnel with the relevant training
- Authorised by the facility operator

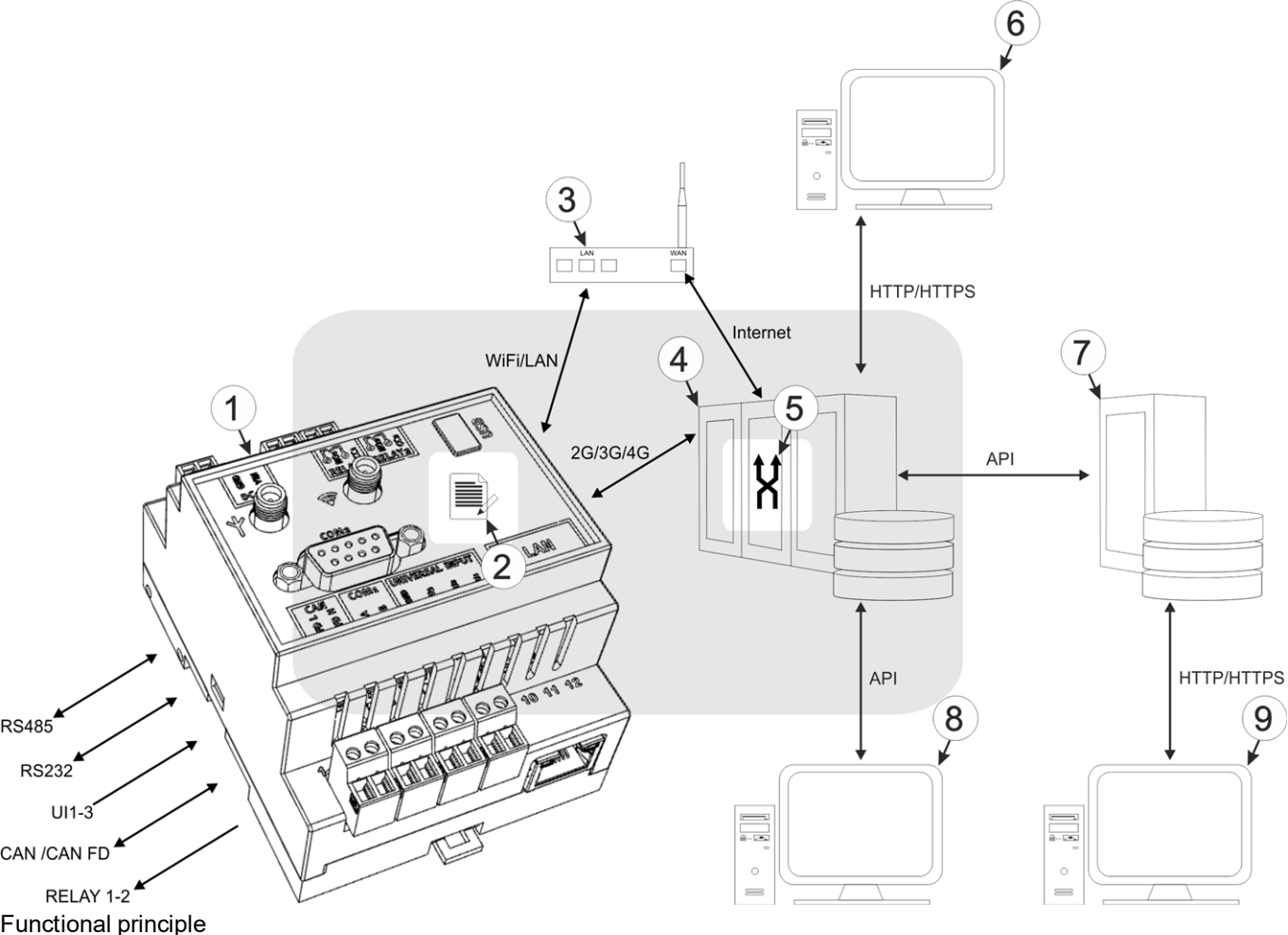
**Note: Qualified personnel**

*In the context of these instructions and the warnings on the product itself, individuals responsible for the setup, installation, commissioning and operation of the product must have gained relevant qualifications relating to their activities, including, for example:*

- *Training, instruction and authorisation to activate/deactivate, ground and label electric circuits and devices/systems in accordance with the standards of safety engineering.*
- *Training or instruction on the maintenance and use of suitable safety equipment in accordance with the standards of safety engineering.*
- *First aid training*

# Chapter 5 Functional principle

In the graphic below, all of the components that are part of the myDatanet are illustrated in grey. All other components must be provided/created by the customer.



<b>1</b>	myDatalogC33x with integrated managed service SIM chip (including data transmission)
<b>2</b>	Application created by the customer (device logic) that collects and records the data (see "Device Logic" on page 105)
<b>4</b>	myDatanet server to which the data is transferred
<b>5</b>	Data Descriptor defined by the customer that enables the use of the measurement data and configurations generated by the application (device logic) in connection with the interface of the myDatanet server (see "Data Descriptor " on page 207)
<b>6</b>	Client that accesses the interface of the myDatanet server via the web browser
<b>7</b>	Customer-specific server that provides clients with their own interface. The customer-specific server obtains the data via the API interface of the myDatanet server (see "API" on page 219).
<b>8</b>	Client, on which a PC program is running, that obtains its data via the API interface of the myDatanet server (see "API" on page 219)
<b>9</b>	Client that accesses the interface of the customer-specific server via the web browser

---

Functions and components provided by myDatanet :

- myDatalogC33x

Programmable (see "Device Logic" on page 105), stationary device with integrated memory and standardised industrial interfaces (UI1-3, RS485, CAN / CAN FD, RS232, isolated switch contact 1-2) to connect machines, sensors and actuators to the myDatanet server (2G/3G/4G/WiFi/LAN)

- Managed Service

Managed Service is the basis for operating the devices and provides a wide range of services. Managed Service includes updates for device firmware, mobile data transmission on a global scale and free support - providing you with one contact person for the entire solution.

- myDatanet server

Database for saving the measurement data and configurations. Data is either accessed via the API (see "API" on page 219) or the web interface of the server.

Functions and components provided by the customer

- Machines, sensors or actuators

Machines, sensors or actuators that include interfaces that are compatible with the specifications listed in the chapter "Specifications" (see "Specifications" on page 13)

- Application (device logic)

The firmware of the myDatalogC33x only manages the synchronisation of the measurement data and configurations between the myDatalogC33x and myDatanet server. The application created by the customer must record the measurement values and create the data blocks that are to be saved. The data blocks, on the other hand, are stored by the firmware of the myDatalogC33x (see "rM2M\_RecData()"). The time of the synchronisation and the type of connection must also be determined by the application created by the customer. Both of the API functions "rM2M\_TxStart()" and "rM2M\_TxSetMode()" are provided for this purpose.

- Data Descriptor

The basic function of the myDatanet server is limited to synchronisation of the measurement data channels ("histdata0" - "histdata9") and configuration blocks ("config0" - "config9") between the myDatalogC33x and server. The Data Descriptor defined by the customer must divide the measurement data channels and configuration blocks into the individual data fields. An explanation is provided in the chapter "Data structure" on page 207.

- Router for connecting the myDatalogC33x to the Internet (optional)

This means that in addition to the mobile network connection, the WiFi or LAN connection is also available as a communication channel for the connection to the myDatanet server.

- Customer-specific server with web interface for the clients (optional)

It can be used to create an individual web interface for the clients. Using this method, the data is read out of the myDatanet server via the API interface (see "API" on page 219) by the customer-specific server.

## 5.1 Recommended procedure

### 5.1.1 Development of M2M/IoT application

It is recommended to start with the definition of the Data Descriptor (see "Data Descriptor " on page 207) when developing a M2M/IoT application. It specifies the various data structures (measurement data, configurations, etc.) that are valid for the Device Logic as well as the myDatanet server. The definitions of the Data Descriptor also apply for accessing the data of the myDatanet server via the API.

The information type should be taken into consideration when assigning the data to the relevant containers ("histdata0" - "histdata9" or "config0" - "config9"). The "histdata0" - "histdata9" containers should be used for time series. If there is measurement data that will be generated frequently and data that will only be generated rarely, it is recommended to use two different containers (e.g. "histdata0" for the frequently generated and "histdata1" for the rarely generated). Similar also applies to the configuration data, for which the "config0" - "config9" containers are provided. When it comes to the configuration data, it is recommended to take a grouping based on logical context into consideration in addition to the frequency of change.

**Note:** *A well thought out selection of the containers helps to reduce the required data volume and associated costs.*

## 5.2 Functionality of the internal data memory

Structure	Circular buffer
Total size	3 MB
Number of sectors	8
Sector size	393.216 Byte

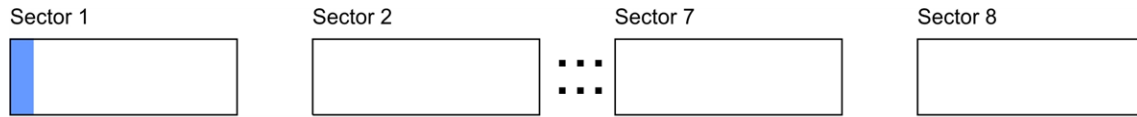
The internal data memory of the myDatalogC33x is designed as a circular buffer with 8 sectors. If the entire memory (3 MB ) is full, the sector with the oldest data is deleted fully before new data can be saved in this sector again. This means that the internal data memory comprises at least 2,625 MB of valid data and a maximum of 3 MB .

For this reason, it is recommended to coordinate the data transmission and record interval in such a way that a maximum of 2,625 MB has to be recorded between two transmissions. If it can be expected that individual transmissions fail due to poor network coverage, this must also be taken into consideration when calculating the data volume to be saved. Additionally, it must be noted that the system-related overhead is 11 Byte per data record and that the first 8 Byte of each sector are reserved for the internal memory management. The 11 Byte overhead already includes the timestamp, so it does not have to be taken into account when calculating the size of the entire record. If there is not enough free space in a sector to save the entire data record, the data record is written to the next sector. This means that a data record is not written over the sector limits.

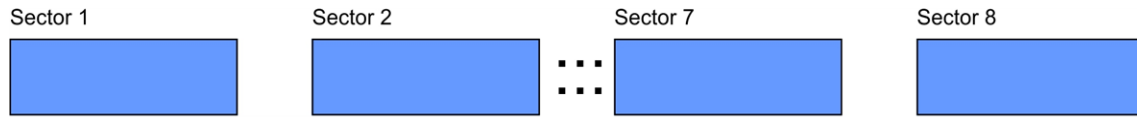
**Note:**

*Additional explanation regarding the functionality of the circular buffer*

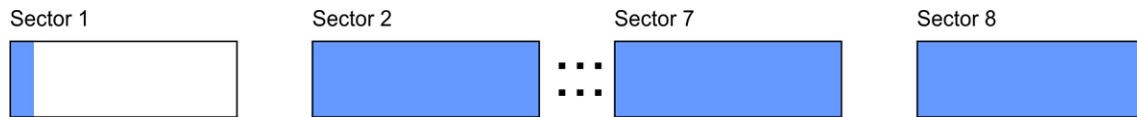
*Data memory after the first data is recorded:*



*Data memory once 3 MB has been recorded*



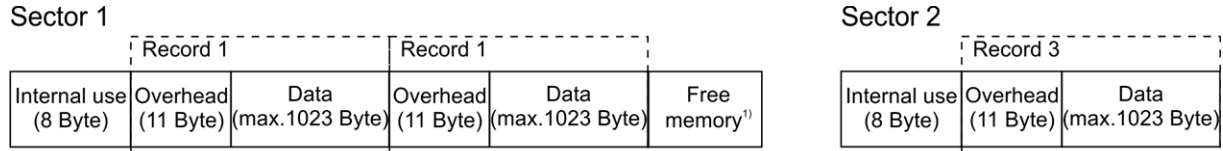
*Data memory if further data is recorded once 3 MB has already been recorded*



**Note:**

*Additional explanation on calculating the data volume to be saved:*

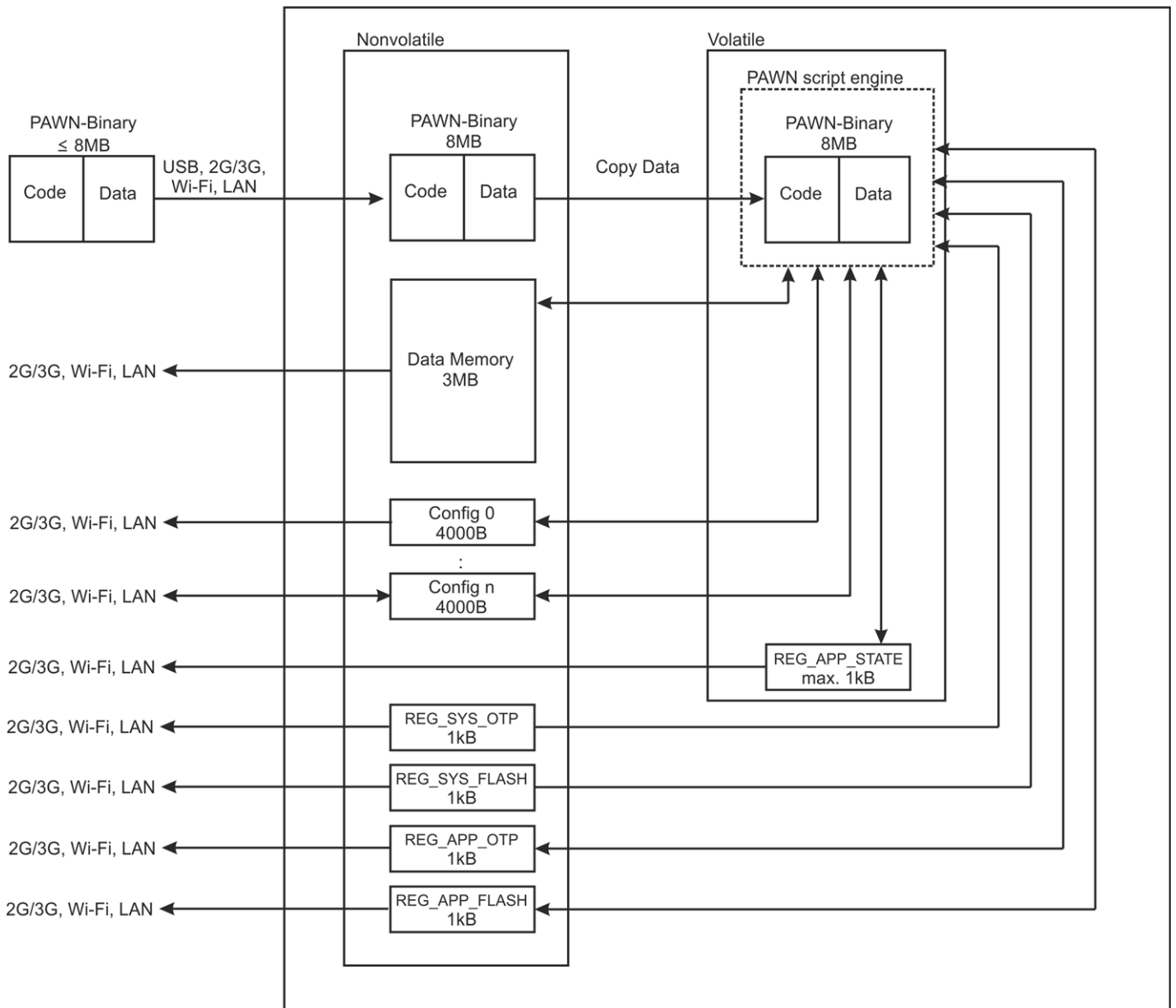
*To provide a clear and simple overview, the following example assumes that the sectors can only record two complete data records.*



<sup>1)</sup> Free memory in sector 1 is not enough to record a full data record (overhead + data).



## 5.3 Memory organisation



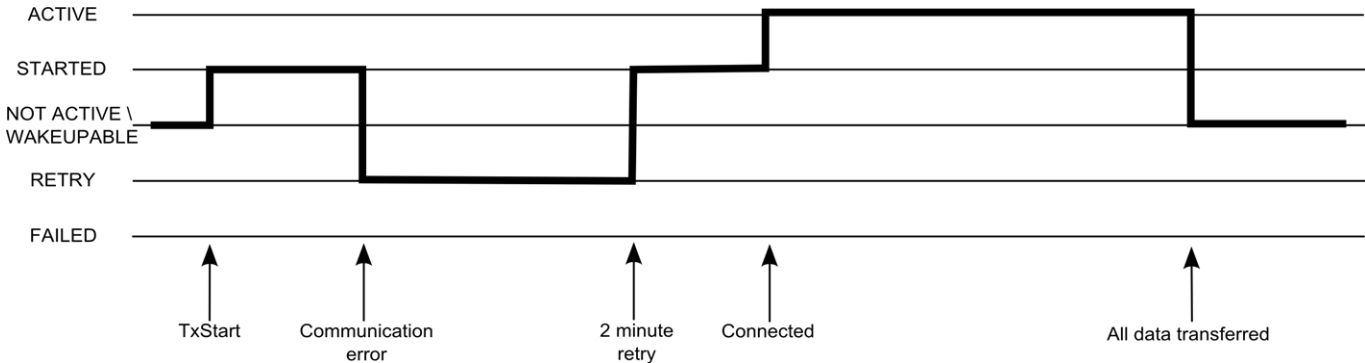
Organisation of the myDatalogC33x memory

1) This memory block is only available if it was initialised via the "rM2M\_RegInit()" function.

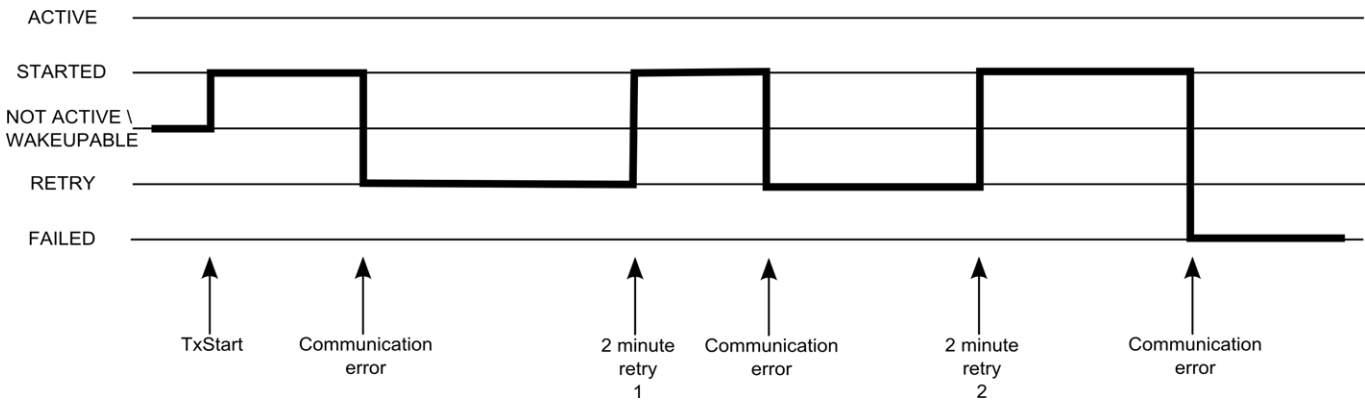
The decompressed size of the PAWN binary that is to be loaded into a myDatalogC33x must not exceed 8MB. In order to reduce the data volume when transferring the PAWN binary into the myDatalogC33x the PAWN binary can be compressed using the compiler instruction (`#pragma amxcompress <0-3>`). The PAWN binary (up to 8MB), 10 configuration blocks (4000 Byte each), 4 registration memory blocks (1kB each) and measurement data (3 MB) are stored in the flash memory of the myDatalogC33x. To execute via the PAWN script engine, the PAWN binary is copied to the RAM and decompressed if necessary. The registration memory blocks that can optionally be initialised using the function "rM2M\_RegInit()" (e.g. REG\_APP\_STATE) are also stored in the RAM.

## 5.4 Procedure in case of connection aborts

If the connection is terminated, another attempt to establish a connection is made after 2min. for all connections, except for "online" mode. Up to 2 attempts to establish a connection are made.



Connection could be established during first retry.



Connection could not be established despite 2 retries.

### ACTIVE

*Connection to the myDatanet server established. Data is being transmitted.*

### STARTED

*Connection establishment initiated.*

### NOT ACTIVE

*The system is waiting for the next connection establishment to be initiated. The last connection attempt was successful and all of the data was transmitted.*

### WAKEUPABLE

*The modem is logged into the GSM network and the system waits for the next connection attempt to be initiated. The last connection attempt was successful and all of the data was transmitted. As the modem is logged into the GSM network, the connection establishment can also be initiated via the myDatanet server (see "myDatanet Server Manual " 805002).*

### RETRY

*The system waits 2min. until the next attempt to establish a connection.*

### FAILED

*The system is waiting for the next connection establishment to be initiated. During the last connection attempt no data or not all of the data was transmitted.*

**Note:** Depending on the type of communication error, the system may be restarted (e.g. to reinstall the SIM chip) before the "FAILED" status is set.

The current connection status can be read out at any time via the "rM2M\_TxGetStatus()" function.

### 5.4.1 Connection abort in "online" mode

A single direct connection attempt is made if the connection is lost in "online" mode. If it is not possible to establish a connection, 2 further attempts of the standard retry sequence are made at intervals of 2min. . If the connection was not established during the last retry, the device remains offline until the next connection attempt is triggered via the "rM2M\_TxStart()" function.

### 5.4.2 Connection abort during a Device Logic download

The device reacts to a connection abort during the Device Logic download with the standard retry sequence (2 connection attempts at intervals of 2min. ). In addition to this, the "SCRIPT\_ERR, SCRIPT DOWNLOAD ERROR" log entry is entered in the device log as soon as the device has detected the connection abort. As the existing Device Logic is located in the RAM, it can continue to be executed until the next PowerOn. It can no longer be executed following a PowerOn and the "SCRIPT\_ERR, NO SCRIPT" error is entered in the device log. The reason for this is that the area in the flash memory where the Device Logic is located is erased at the start of the Device Logic download.

## 5.5 Timeout monitoring in online mode

In online mode, the myDatalogC33x sends a keep alive ping to the myDatanet server by default at an interval of 15 min. and 3 sec. (i.e. every 903 sec.). This enables the server to detect whether the connection to the myDatalogC33x is still available. To be able to detect interruptions to the connection more promptly, the standard interval for the keep alive ping can be adjusted via the "rM2M\_SetTCPKeepAlive()" function.

The "Bidirectional alive ping" can be activated on the server to ensure that the myDatalogC33x can also promptly detect an interruption to the connection. This bidirectional alive ping can be activated globally for the complete server, for a specific customer or for a single site (see "myDatanet Server Manual " 805002). If the bidirectional alive ping is enabled, the myDatanet server sends a corresponding response to every keep alive ping from the device (keep alive response). Following receipt of the first keep alive response, the myDatalogC33x will expect to receive a regular keep alive response within 10 sec. of the keep alive ping. If the keep alive response fails to appear three times in a row, an attempt is initially made to re-establish the communication without completely disconnecting the connection (i.e. by only reinitialising the connection on a TCP level only). Only once this has not worked will the myDatalogC33x disconnect the connection to the myDatanet server completely and will immediately establish the connection again. The recording of the round trip time [ms] is also activated upon receipt of the first keep alive response. This means that the time until the keep alive response has been received by the server is measured for every subsequent keep alive ping. The "rM2M\_TxItfGetStats()" function can be used to read the last determined "Round trip time" of the system.

## 5.6 Automatic selection of the GSM network

The GSM network to which the device should register must be selected, as the myDatalogC33x is equipped with a SIM chip that provides a mobile connection via a variety of international service providers (see [www.microtronics.com/footprint](http://www.microtronics.com/footprint) ). This is completed automatically by the device.

---

## 5.7 Determining the GSM/UMTS/LTE signal strength

The internal update rate of the measurement value for the GSM/UMTS/LTE signal strength is dependent on the type of connection selected via the "rM2M\_TxSetMode()" function:

- Interval: Updated during connection establishment
- Interval & Wakeup: Updated every 30 seconds
- Online: Updated every 5 seconds

The "rM2M\_GSMGetRSSI()" function can be used to read the last determined value from the system.

## 5.8 Determining the GSM position data

An internal flag is set by the firmware every 24h , which ensures that the GSM position data is also determined the next time the "rM2M\_TxStart()" function is called up. The positioning can however also be suppressed by setting the "RM2M\_TX\_SUPPRESS\_POSUPDATE" flag when calling up the function. It is also possible to trigger the determination of the GSM position data by setting the "RM2M\_TX\_POSUPDATE" flag when calling up "rM2M\_TxStart()". If "Interval & Wakeup" connection mode was activated on the myDatalogC33x , the previously described internal flag is set by the firmware and the connection establishment is triggered by the receipt of a Wakeup SMS (i.e. via the myDatagnet server), the determination of the GSM position data is definitely executed and cannot be suppressed. The GSM position data cannot be generated if "online" connection mode is active.

## 5.9 Error handling

The following transmission mechanisms have been integrated in the firmware to ensure that problems with the Device Logic can be diagnosed and resolved remotely. In the event that there is no Device Logic, a connection to the myDatagnet server is established every 24h . This backup interval is set to 1h if an existing Device Logic has been deactivated due to an error being detected by the system. The "Interval & wakeup" connection type is activated in both cases, which enables a connection to be initiated via the interface of the myDatagnet (see "myDatagnet Server Manual " 805002).

## 5.10 Registration memory blocks

In addition to 4 1kB blocks, that are saved in the flash, another one, that is saved in the RAM, can optionally be initialised via the "rM2M\_RegInit()" function. Its size can be specified during initialisation, although it is limited to a maximum of 1kB. The registration memory blocks provide the option of storing device-specific data and synchronising it with the myDatanet server. The blocks only differ with regard to their access options and storage location. This results in predefined intended uses that are described in the following table:

Memory block	Access	Memory	Purpose
System-specific data			
REG_SYS_OTP <sup>1)</sup>	<b>readable:</b> Device Logic, myDatanet server	FLASH	System information that is written once as part of the production process
REG_SYS_FLASH <sup>1)</sup>	<b>readable:</b> Device Logic, myDatanet server	FLASH	System information that must be able to be changed during operation
Application-specific data			
REG_APP_OTP	<b>readable:</b> Device Logic, myDatanet server <b>writable:</b> Device Logic	FLASH	Application-specific information that is written once as part of the production process (recommendation, writing it multiple times is not prevented by the firmware)
REG_APP_FLASH	<b>readable:</b> Device Logic, myDatanet server <b>writable:</b> Device Logic	FLASH	Application-specific information that must be able to be changed during operation
Application-specific, volatile data			
REG_APP_STATE <sup>2)</sup>	<b>readable:</b> Device Logic, myDatanet server <b>writable:</b> Device Logic	RAM	Application-specific information that must be able to be changed during operation and that does not require non-volatile storage in the flash (e.g. current device status).

<sup>1)</sup> Writing data in these two memory blocks is reserved for the manufacturer.

<sup>2)</sup> This memory block is only available if it was initialised via the "rM2M\_RegInit()" function.

**Note:** It is also possible to write in the memory blocks as part of the production process via the local interfaces (USB and both UART). However, an agreement must be reached with the manufacturer to receive information about this (see "Contact information" on page 247).

The "rM2M\_RegGetString()", "rM2M\_RegGetValue()", "rM2M\_RegSetString()", "rM2M\_RegSetValue()", "rM2M\_RegDelValue()" and "rM2M\_RegDelKey()" functions are available for accessing the registration memory blocks.

### 5.10.1 REG\_APP\_OTP

By saving the "Product Identity Profile" (PIP) in this registration memory block, the functions described in the following can be initiated on the myDatanet server. The PIP consists of the following fields:

#### pipCustomer

*Name of the customer to whom the site should be assigned [2-50 characters].*

---

**pipCtx**

*Name of the site that should be created/used [2-50 characters].*

**pipAppId**

*ID of the IoT application based on which the site should be created [max. 50 characters].*

**pipAppVer (optional)**

*Version of the Device Logic currently installed on the device (e.g. 7) [Integer].*

**pipCtxAutocreate (optional)**

*Indicates whether the site (if it does not exist yet) should be created ("0" or "1" must be saved as the string)*

- "0": creation of a new site is not permissible
- "1": new site can be created (default)

If the myDatanet server receives a PIP, the two basic scenarios are differentiated:

- There is no site with the name specified in the "pipCtx" field:

The new site is only created if the customer and application template were found on the myDatanet server and "pipCtxAutocreate=1" or the field is missing.

- A site with the name specified in the "pipCtx" field was found on the server:

In this case, the "pipCtxAutocreate" and "pipCustomer" fields are not relevant. The device is assigned to the site, even if it is located within a different customer, if the application ID in the "pipAppId" field and that of the found site match. The device is moved to the relevant customer for this purpose. The allocation of the existing device is deallocated if a device is already assigned to the site. The existing device is moved to the customer's pool.

## 5.11 File transfer

It is possible to register up to 60 files for the file transfer (see "FT\_Register()"). A callback function, that should be called up when a file transfer command is received, must be transferred to the "FT\_Register()" function during this process (see "Callback functions" on page 176). The callback function must be able to handle all file transfer commands (see "File transfer commands" in chapter "Constants" on page 176). The file properties must also be set via the "FT\_SetProps()" function as part of the registration process. If a file should no longer be available for the file transfer, it can be removed from the registration using the "FT\_Unregister()" function.

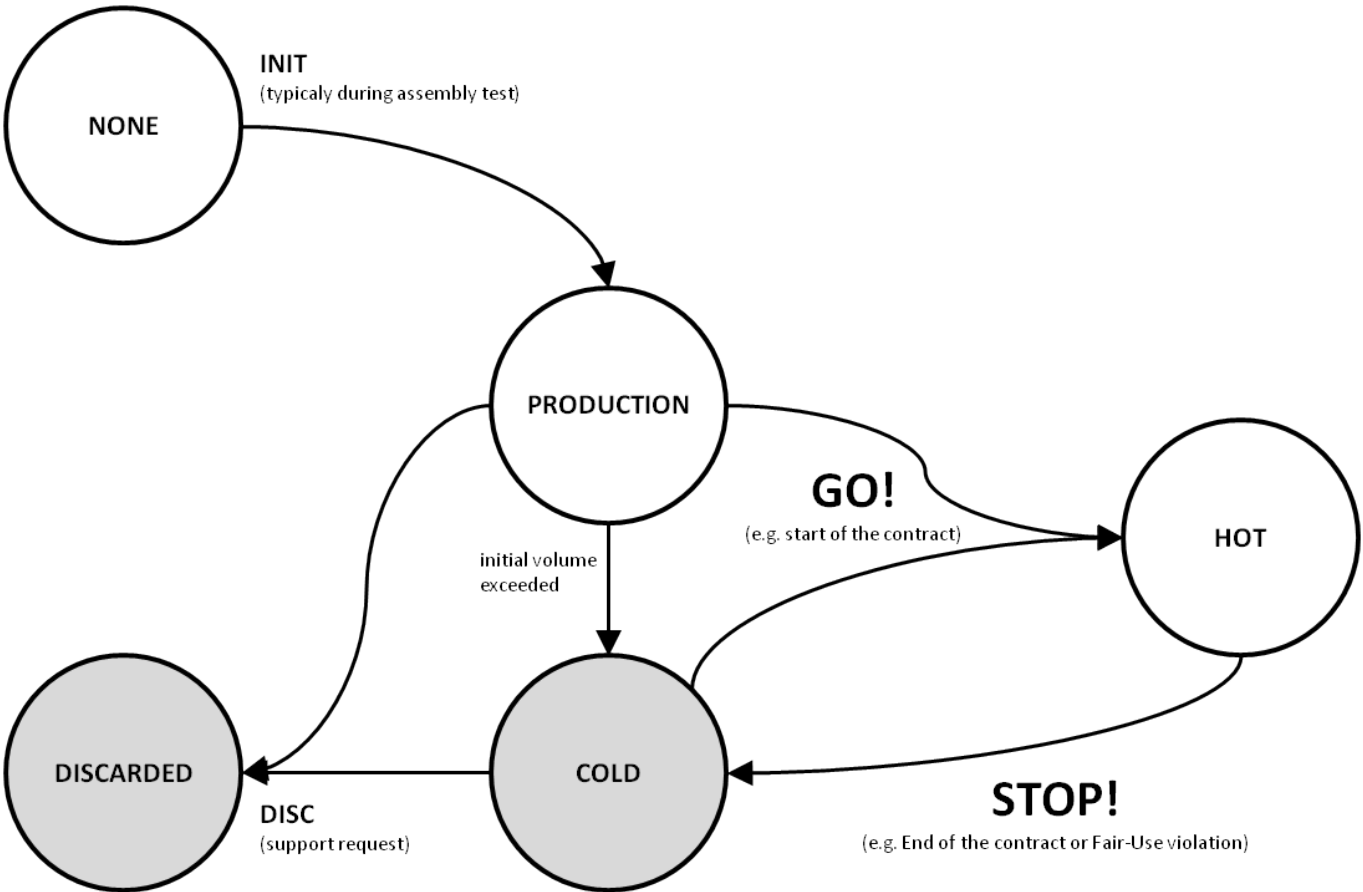
Upon receipt of a file transfer command, a session is started that is automatically terminated after 15sec. if the Device Logic is not handling the command correctly. Sessions can only be active one after another to prevent any conflicts.

### 5.12 Meaning of the SIM state

The device receives information about the permissible use of the SIM chip from the myDatenet server. The following states are defined for this SIM status:

SIM state	Transmission via Device Logic	Explanation
RM2M_SIM_STATE_NONE	Yes	Initial state
RM2M_SIM_STATE_PRODUCTION	Yes	New device is in stock
RM2M_SIM_STATE_HOT	Yes	Valid contract
RM2M_SIM_STATE_COLD	No	End of contract or violation of fair use policy
RM2M_SIM_STATE_DISCARDED	No	Device decommissioned

As the previous table illustrates, it is not possible to trigger the connection by Device Logic for the "RM2M\_SIM\_STATE\_COLD" and "RM2M\_SIM\_STATE\_DISCARDED" states. In this case, the functions "rM2M\_TxStart()" and "rM2M\_TxSetMode()" return error code "ERROR\_SIM\_STATE". Contact the manufacturer, to switch a device that is in the "RM2M\_SIM\_STATE\_COLD" state back to the "RM2M\_SIM\_STATE\_HOT" state (see "Contact information" on page 247). The SIM state can be read out at any time using the "rM2M\_GSMGetInfo()" function. An entry is also added to the device log every time the SIM state changes (e.g. SIM\_STATE, HOT).



State diagram of the SIM states





# Chapter 6 Storage, delivery and transport

## 6.1 Inspection of incoming deliveries

Check the shipment immediately upon receipt to ensure it is complete and intact. Immediately report any discovered transport damages to the delivering carrier. Also notify Microtronics Engineering GmbH in writing about this without delay. Report any incompleteness of the delivery to the responsible representative or directly to the company headquarters of the manufacturer within two weeks (see "Contact information" on page 247).

**Note:** Any claims received thereafter will not be accepted.

## 6.2 Scope of supply

**Note:** An antenna which is absolutely necessary for operation (see "Antennas" on page 235) is not part of the standard scope of delivery and must be ordered separately.

The standard scope of delivery of the myDatalogC33x WIFI/2G/3G/4G World (301331) includes:

- rapidM2M M23x WIFI/2G/3G/4G World <sup>1)</sup>

<sup>1)</sup> The rapidM2M module (rapidM2M M23x WIFI/2G/3G/4G World ) was already inserted in the myDatalogC33x WIFI/2G/3G/4G World during production.

Additional accessories such as assembly sets, antennas, power supply unit, etc. depending on the order. Please check these against the delivery slip.

## 6.3 Storage

The following storage conditions must be observed:

myDatalogC33x	Storage temperature	-40...+85°C
	Humidity	15...90%rH

Store the measurement technology so that it is protected against corrosive and organic solvent vapours, radioactive emissions and strong electromagnetic radiation.

## 6.4 Transport

Protect the myDatalogC33x against heavy shocks, bumps, impacts or vibrations. The original packaging must always be used for transport.

---

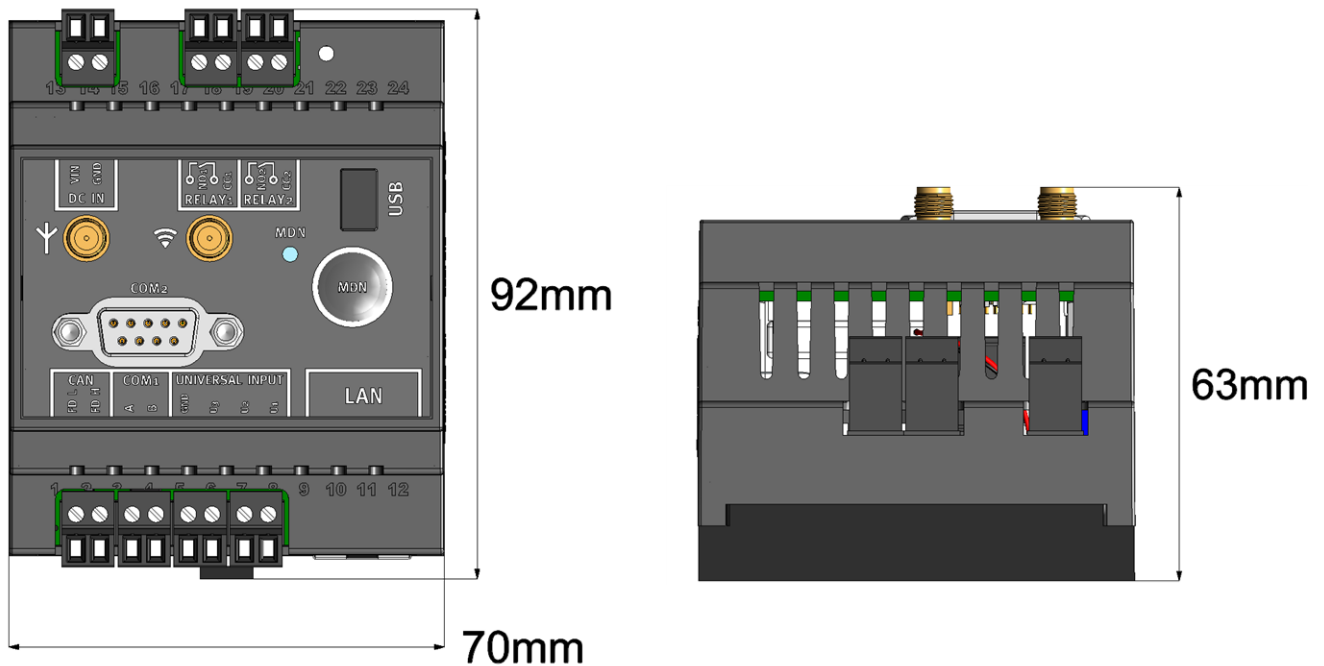
## 6.5 Return

Every return must be accompanied by a fully field-out return form. This return form is available in the service area of the myDatanet server. An RMA number is mandatory for any returns and can be obtained from the Support & Service Centre (see "Contact information" on page 247). The return shipment of the myDatalogC33x must occur in the original packaging and with freight and insurance paid to Microtronics Engineering GmbH (see "Contact information" on page 247). Insufficiently cleared return shipments will otherwise not be accepted!

# Chapter 7 Installation

**Important note:** To prevent any damage to the device, the work described in this section of the instructions must only be performed by qualified personnel.

## 7.1 Dimensions



Dimensions: Width and height

Dimensions: Depth

## 7.2 Installing the myDatalogC33x

**Important note:**

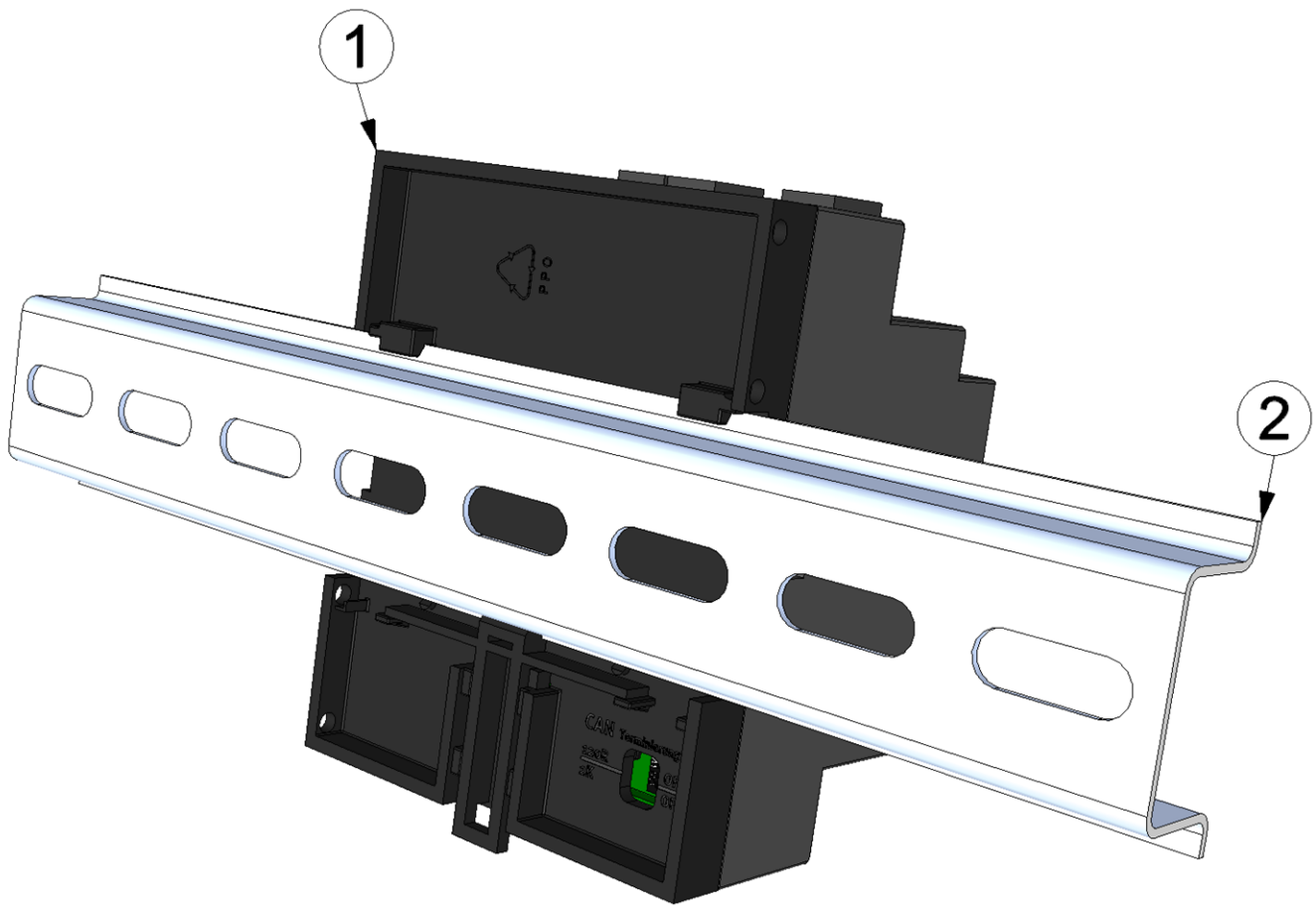
- Ensure installation is completed correctly.
- Comply with existing legal and/or operational directives.
- Improper handling can cause injuries and/or damage to the devices.
- The myDatalogC33x is not approved for use in closed channels.

The installation site must be selected according to specific criteria. The following conditions must be avoided in any case:

- Direct sunlight
- Direct weather exposure (rain, snow, etc.)
- Objects that radiate intense heat (maximum ambient temperature:  $-20\dots+60^{\circ}\text{C}$ )
- Objects with a strong electromagnetic field (frequency converter or similar)
- Corrosive chemicals or gases
- Mechanical impacts
- Direct installation on paths or roads
- Vibrations
- Radioactive emissions

**Note:** Approx. 2-5 cm of space must be left above and below the device for the cable connections. The antenna connections are located on the front of the device. The space required depends on the antennas used. Further information regarding the installation dimensions can be found in the relevant sub-chapter.

## 7.2.1 Top-hat rail assembly



Top-hat rail assembly

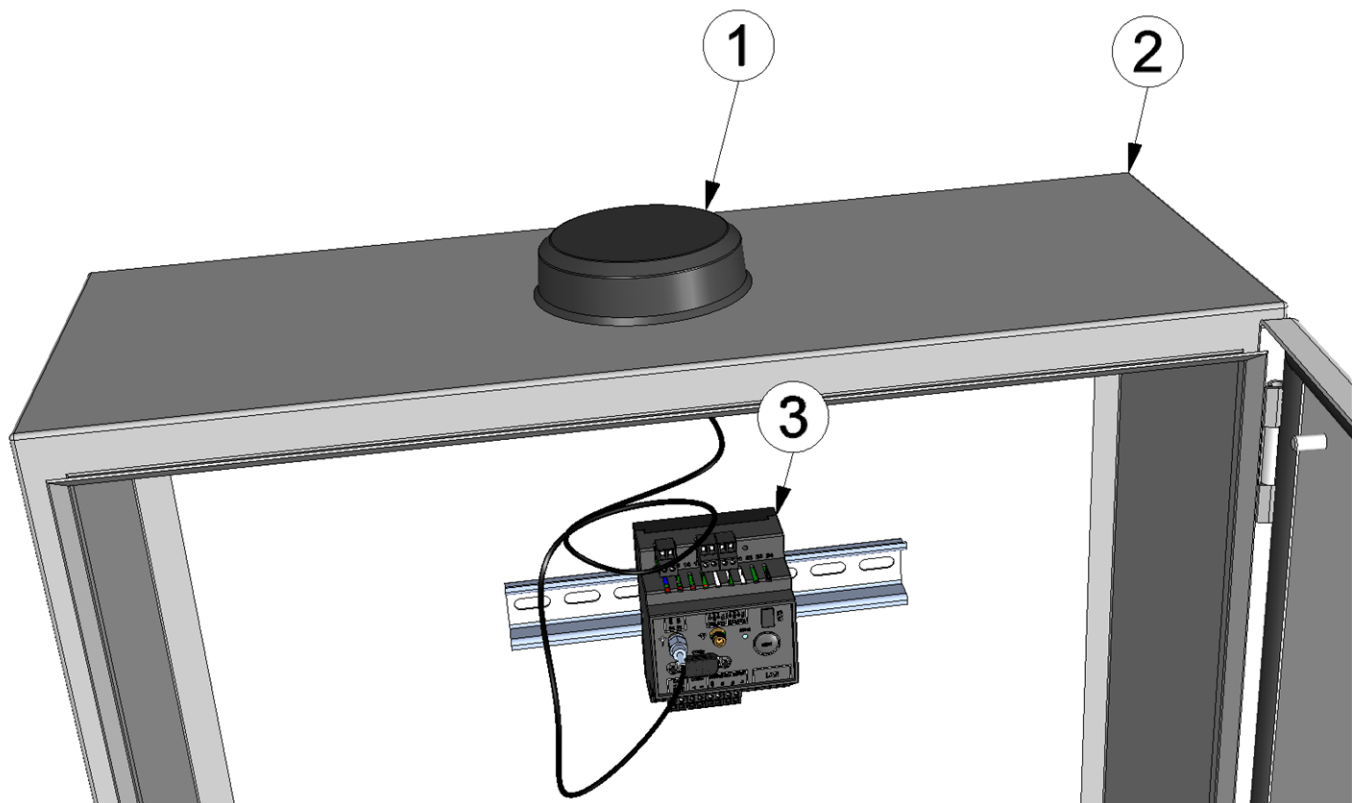
1 myDatalogC33x	2 Top-hat rail
-----------------	----------------

1. Place the myDatalogC33x on to the top edge of the top-hat rail. Turn slightly around the horizontal axis so that the myDatalogC33x clicks into the top-hat rail (see Figure "Top-hat rail assembly" on page 44).

## 7.2.2 Assembly in a control cabinet

The Rod antenna multi band 2G/3G SMA-M (301075) is not suitable for assembly within a control cabinet as the mobile network signal is shielded by the metal of the cabinet. In this case, the manufacturer recommends using the Flat antenna Disc SMA-M 2,5m (206.816) or Flat Antenna Disc US SMA-M 2,5m (206.818) that are available as accessories.

**Important note:** The installation of the Flat antenna Disc SMA-M 2,5m or Flat Antenna Disc US SMA-M 2,5m is only possible up to a wall thickness of 3mm .



Control cabinet with mounted Flat antenna Disc SMA-M 2,5m (206.816)

1 Flat antenna Disc SMA-M 2,5m (206.816)	3 myDatalogC33x
2 Control cabinet	

---

## 7.3 Safety instructions for cabling

**Important note:** To avoid any damage, always switch off the voltage supply to the device when performing electrical connections.

When connections are made to the myDatalogC33x, the following warnings and information must be observed, in addition to the warnings and information found in the individual chapters on the installation. Further safety information is included in "Safety instructions" on page 17.

### 7.3.1 Information on preventing electrostatic discharges (ESD)

**Important note:** Maintenance procedures that do not require the device to be connected to the power supply should only be performed once the device has been disconnected from the mains power supply to minimise hazards and ESD risks.

The sensitive electronic components inside the device can be damaged by static electricity, which can impair the device performance or even cause the device to fail. The manufacturer recommends the following steps to prevent any damage to the device caused by electrostatic discharges:

- Discharge any static electricity present on your body before handling the electronic components of the device (such as circuit boards and components attached thereto). To do this, you can touch a grounded metallic surface such as the housing frame of a device or a metal pipe.
- Avoid any unnecessary movements to prevent the build-up of static charges.
- Use antistatic containers or packaging to transport components that are sensitive to static.
- Wear an antistatic wristband that is grounded via a cable to discharge your body and keep it free of static electricity.
- Only touch components that are sensitive to electric charges in an antistatic working area. If possible, use antistatic mats and work pads.

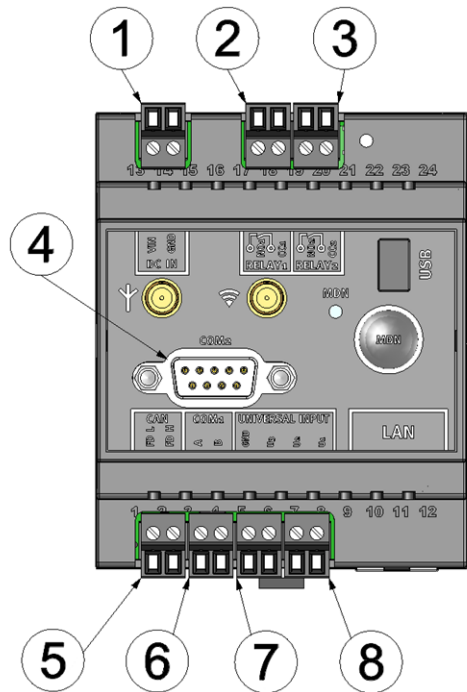
## 7.4 Electrical installation

**Important note:** Only qualified personnel should undertake the installation described in this chapter of the operating instructions to avoid any damage to the device.

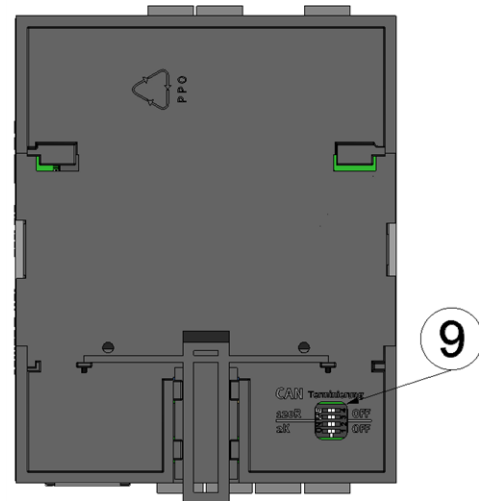
### 7.4.1 Connecting the sensors, actuators and power supply

**Important note:**

- All wiring work must be performed in the de-energised state.
- Ensure installation is completed correctly.
- Comply with existing legal and/or operational directives.
- Improper handling can cause injuries and/or damage to the instruments.
- Run all data and power cables so that they do not pose a trip hazard and ensure that cables do not have any sharp bends.



Connection of the sensors, actuators and power supply (front side)



Connection of the sensors, actuators and power supply (rear side)

1 Supply (V IN, GND)	6 Com1 (RS485)
2 Relay 1	7 Ground and universal input 1
3 Relay 2	8 Universal input 2-3
4 Com2 (RS232)	9 Dip switch for activating/deactivating the load resistances for the CAN interface
5 CAN / CAN FD	

#### DC IN

VIN	Supply voltage: 9...32VDC (+/-10%), max. 9W
GND	Ground

#### RELAY1<sup>1)</sup>

NO1	Operating contact of the isolated switch contact 1 (normally open)
CC1	Root for the isolated switch contact 1

<sup>1)</sup> Parallel connection of a PhotoMOS relay (high switching frequencies) and a mechanical relay (high switching currents)

#### RELAY2<sup>1)</sup>

NO2	Operating contact of the isolated switch contact 2 (normally open)
CC2	Root for the isolated switch contact 2

<sup>1)</sup> Parallel connection of a PhotoMOS relay (high switching frequencies) and a mechanical relay (high switching currents)

---

**COM2**

1	NC
2	RXD line of the RS232 interface
3	TXD line of the RS232 interface
4	NC
5	Ground
6	NC
7	RTS line of the RS232 interface
8	CTS line of the RS232 interface
9	5V <sup>1)</sup> , max. 750mA

<sup>1)</sup> Supply voltage (reserved for extensions)

**CAN**

FD L	CAN Low line of the CAN interface
FD H	CAN High line of the CAN interface

**COM1**

A	RS485 A
B	RS485 B

**UNIVERSAL INPUT**

GND	Ground
UI3	Universal input 3
UI2	Universal input 2
UI1	Universal input 1

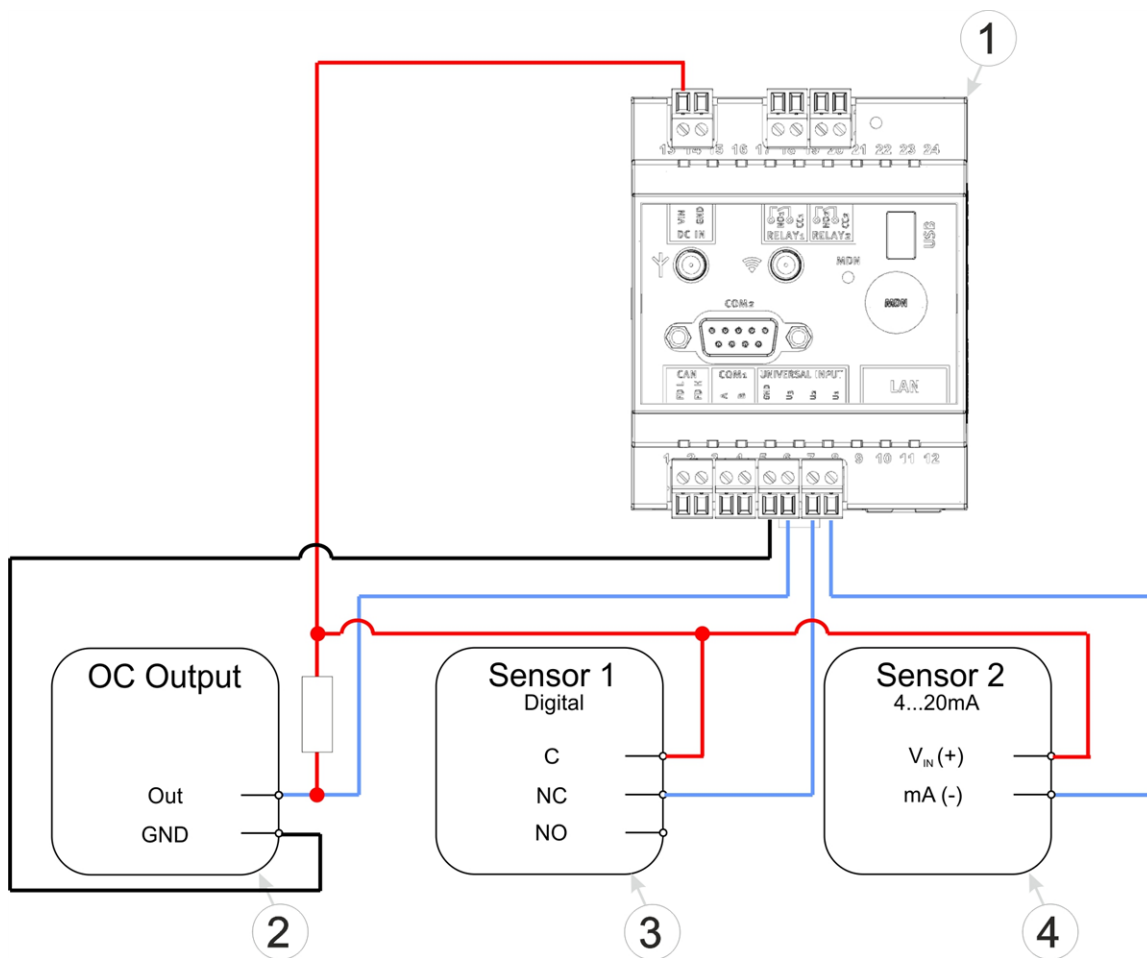
1. Connect your sensors and actuators with the inputs and outputs. Ensure that no current is present! Ensure the supply cables for the myDatalogC33x are in a de-energised state when being connected to the supply connectors.
2. Connect the antenna (see "Connection of the GSM antenna" on page 50).
3. Switch on the 9...32VDC supply voltage of the myDatalogC33x .

The following step is not mandatory.

4. Check whether the connection to the myDatanet has worked correctly (see "Testing communication with the device" on page 70).

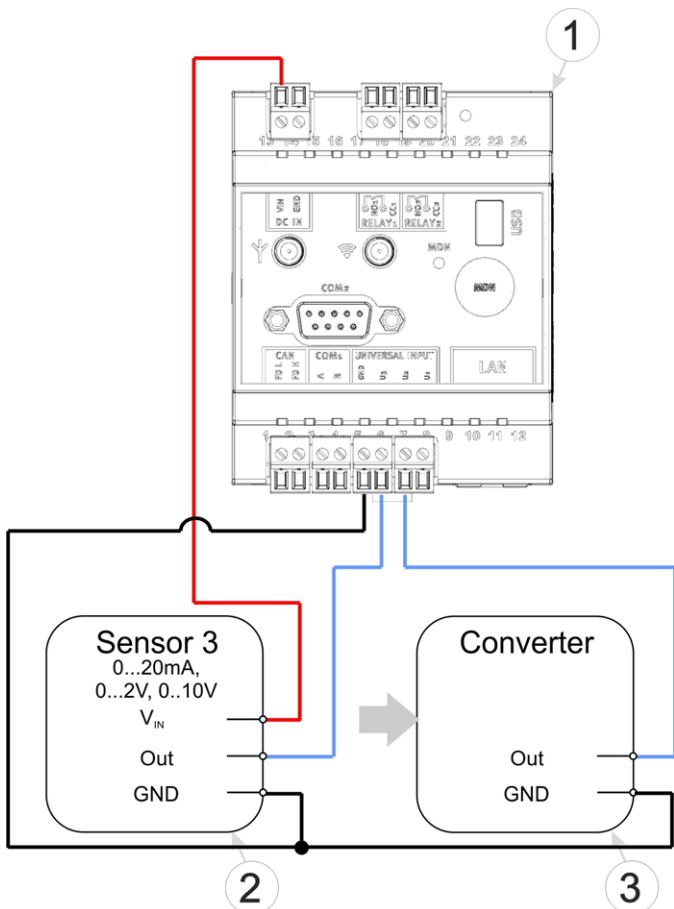


## 7.4.1.1 Connection examples



Connection examples (OC output, digital, 4...20mA)

1 myDatalogC33x	4 Isolated relay contact
2 Sensor with open collector output	4 2-wire mA sensor



Connection examples (0...20mA, 0...2V, 0...10V, converter)

1 myDatalogC33x	3 Signal converter, isolating transducer
2 3-wire mA sensor or 3-wire U-sensor	

## 7.4.2 Connection of the GSM antenna

**Important note:** To ensure the correct functionality, only use antennas that are supplied by the manufacturer.

The standard antenna is directly attached to the antenna connector (see "Overview" on page 20) of the myDatalogC33x. In the event of a low radio signal strength, you can use the Dome antenna multi band SMA-M 3m (301212).

If the distance between the antenna position and the myDatalogC33x is too great, you can use a 2.5m Extension cable for antenna SMA-M/SMA-F 2,5m (206.807).

1. Ensure that the myDatalogC33x is de-energised.
2. If you need an antenna extension, connect this to the antenna first.
3. Connect the antenna extension or antenna directly to the antenna connector of the myDatalogC33x (see "Overview" on page 20).

**Important note:** Do not apply too much force when tightening the antenna. Do not use any tools to tighten the antenna or antenna extension; only tighten it manually.

4. Switch the voltage supply of the myDatalogC33x back on.

The following step is not mandatory.

5. Check whether the connection to the myDatanet has worked correctly (see "Testing communication with the device" on page 70).

### 7.4.3 Connecting the extension modules

***Important note:***

- *All wiring work must be performed in the de-energised state.*
- *Ensure installation is completed correctly.*
- *Comply with existing legal and/or operational directives.*
- *Improper handling can cause injuries and/or damage to the instruments.*
- *Run all data and power cables so that they do not pose a trip hazard and ensure that cables do not have any sharp bends.*

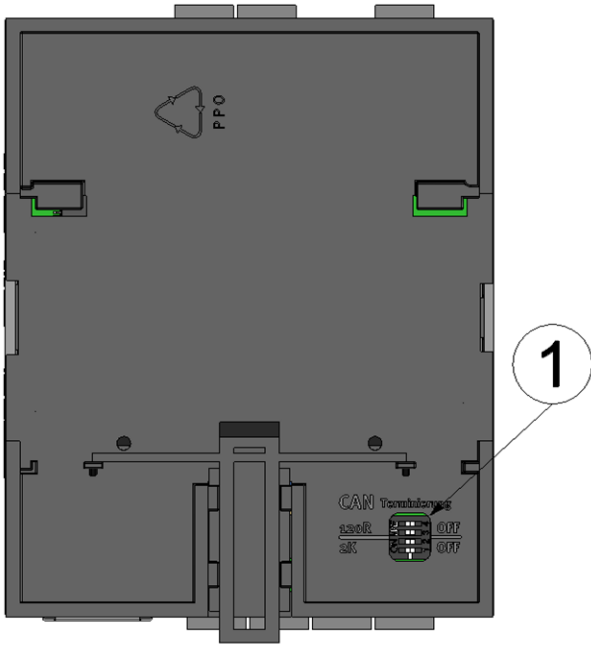
A list of compatible extension modules is provided in the chapter "Extension modules" on page 236. It is possible to connect up to 10 extension modules to the myDatalogC33x via a CAN bus. The total length of the CAN bus must not exceed 20m . If the CAN interface of the myDatalogC33x is used to connect extension modules, it is no longer available for measurement and control tasks (i.e. for the direct connection of sensors and actuators). In this type of application, no other bus participants except for the extension modules and the myDatalogC33x should be connected to the CAN bus (i.e. a separate, encapsulated CAN bus is required for the connection of extension modules).

### 7.4.3.1 CAN bus without branch lines

**Important note:** All wiring work must be performed in the de-energised state.

1. Set the load resistances for the CAN interface on all of the bus participants (extension modules and myDatalogC33x ) based on the position they will have on the bus. The 120Ω load resistances (S3 and S4 of the dip switch) must be activated on the first and last bus participant.

**Note:** Setting the load resistances before installation of the devices is recommended as the dip switches to activate/deactivate the load resistances for the CAN interface are located on the rear of the device.

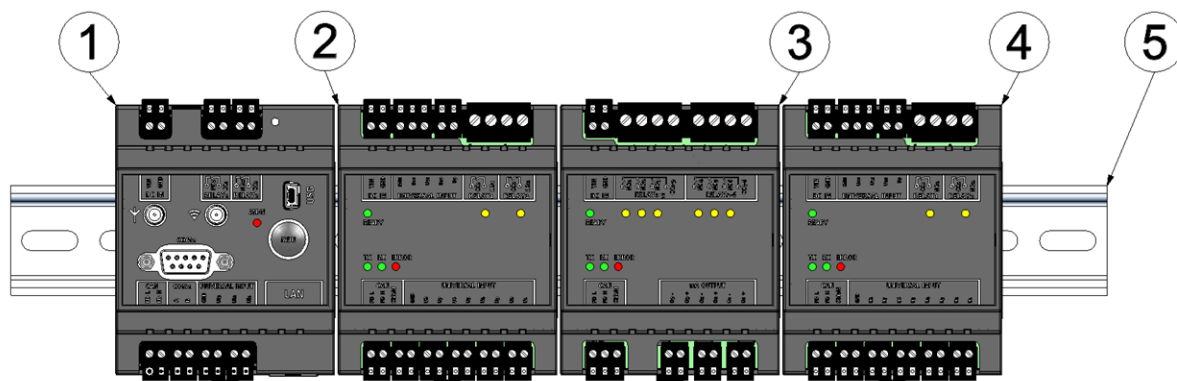


Rear of the myDatalogC33x or of an extension module

- |  |
|--|
| <b>1</b> Dip switch for activating/deactivating the load resistances for the CAN interface |
|--|

2. Position the myDatalogC33x and the extension modules in the final installation position (e.g. next to one another on a top-hat rail). Information regarding the correct installation of the myDatalogC33x is provided in chapter "Installing the myDatalogC33x" on page 43. Information on installing the extension modules is provided in the manual of the relevant extension module.

**Note:** When selecting the installation position for the individual devices, ensure that the total length of the CAN bus 20m must not be exceeded



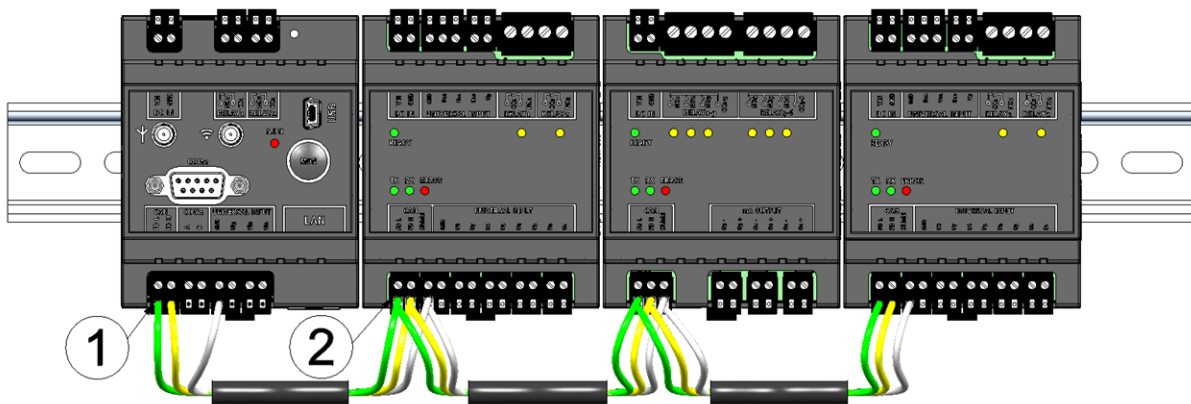
Top-hat rail with myDatalogC33x and extension modules attached

1 myDatalogC33x (120Ω load resistance activated)	4 myDatalogC3e 12UI/2Rel (120Ω load resistance activated)
2 myDatalogC3e 12UI/2Rel	5 Top-hat rail
3 myDatalogC3e 3mA/6Rel	

3. Connect the CAN interface of the myDatalogC33x with those of the extension modules. All of the "FD L" terminals and all of the "FD H" terminals must be connected with one another during this process. Ensure that no current is present!

**Note:** Use a shielded cable to connect the CAN interfaces. The "Shield" terminal is available on the extension modules to connect the cable shield. The cable shield on the myDatalogC33x must be connected to the GND terminal. Use a twin wire end sleeve or distributor terminal if you want to connect several cables to the GND terminal.

**Note:** As there is only one terminal for each signal on the myDatalogC33x and on the extension modules, you should use twin wire end sleeves if you want to use more than one extension module.

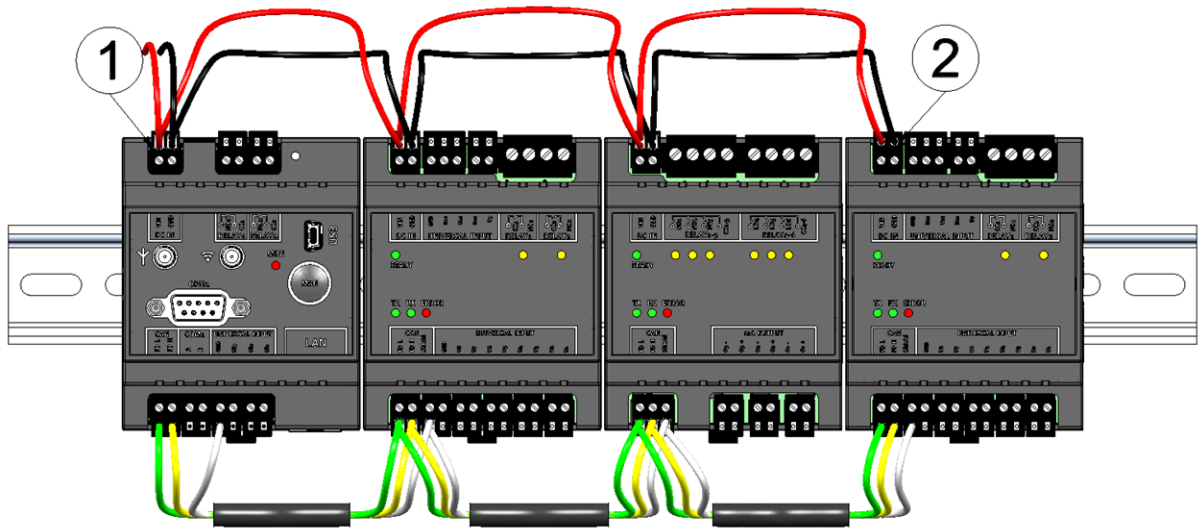


Connecting the cables of the CAN interface

1 Wire end sleeve	2 Twin wire end sleeve (wire end sleeve for two cables)
-------------------	---

4. Connect your sensors and actuators with the inputs and outputs of the myDatalogC33x myDatalogC3xx and the extension module. Ensure that no current is present!

- Connect the cables for the power supply of the myDatalogC33x and the extension modules with the VIN and GND terminals. Ensure that no current is present when establishing the connection. In this case, use twin wire end sleeves if more than one cable should be connected per terminal.



Connecting the power supply

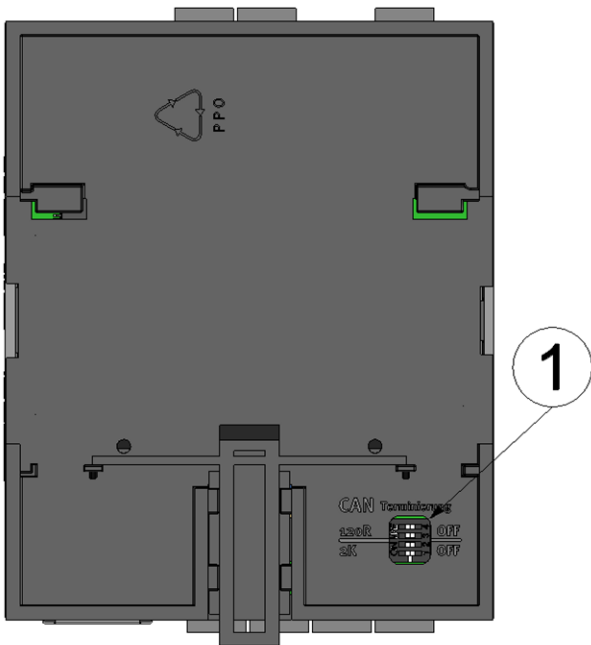
<b>1</b> Twin wire end sleeve	<b>2</b> Wire end sleeve
-------------------------------	--------------------------

### 7.4.3.2 CAN bus with branch line

**Important note:** All wiring work must be performed in the de-energised state.

1. Set the load resistances for the CAN interface on all of the bus participants (extension modules and myDatalogC33x ) based on the position they will have on the bus. The 120Ω load resistances (S3 and S4 of the dip switch) must be activated on the first and last bus participant. On bus participants that are connected to the bus via the branch line, the 2k load resistance (S1 and S1 of the dip switch) must be activated if the branch line is longer than 0,3m .

**Note:** Setting the load resistances before installation of the devices is recommended as the dip switches to activate/deactivate the load resistances for the CAN interface are located on the rear of the device.



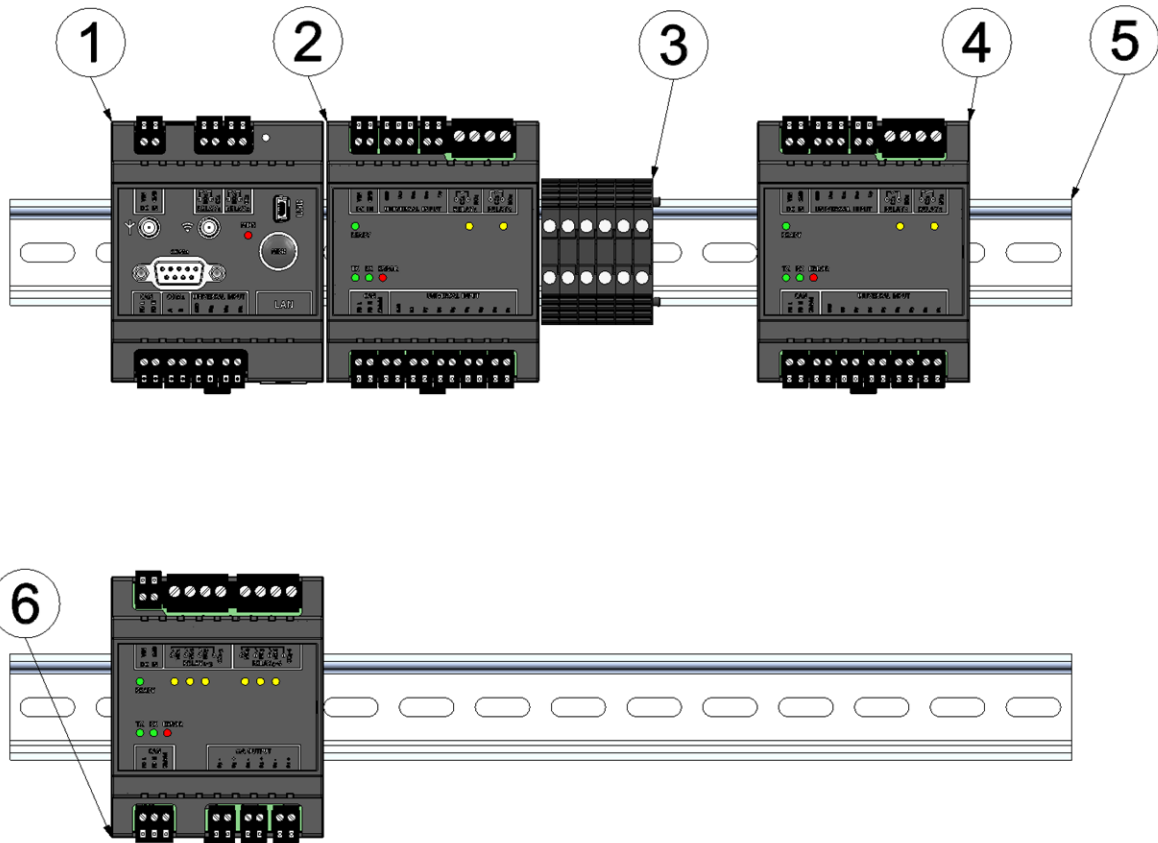
Rear of the myDatalogC33x or of an extension module

- |  |
|--|
| <b>1</b> Dip switch for activating/deactivating the load resistances for the CAN interface |
|--|



2. Position the myDatalogC33x and the extension modules in the final installation position (e.g. installed on several top-hat rails). Information regarding the correct installation of the myDatalogC33x is provided in chapter "Installing the myDatalogC33x" on page 43. Information on installing the extension modules is provided in the manual of the relevant extension module.

**Note:** When selecting the installation position for the individual devices, ensure that the total length of the CAN bus 20m must not be exceeded



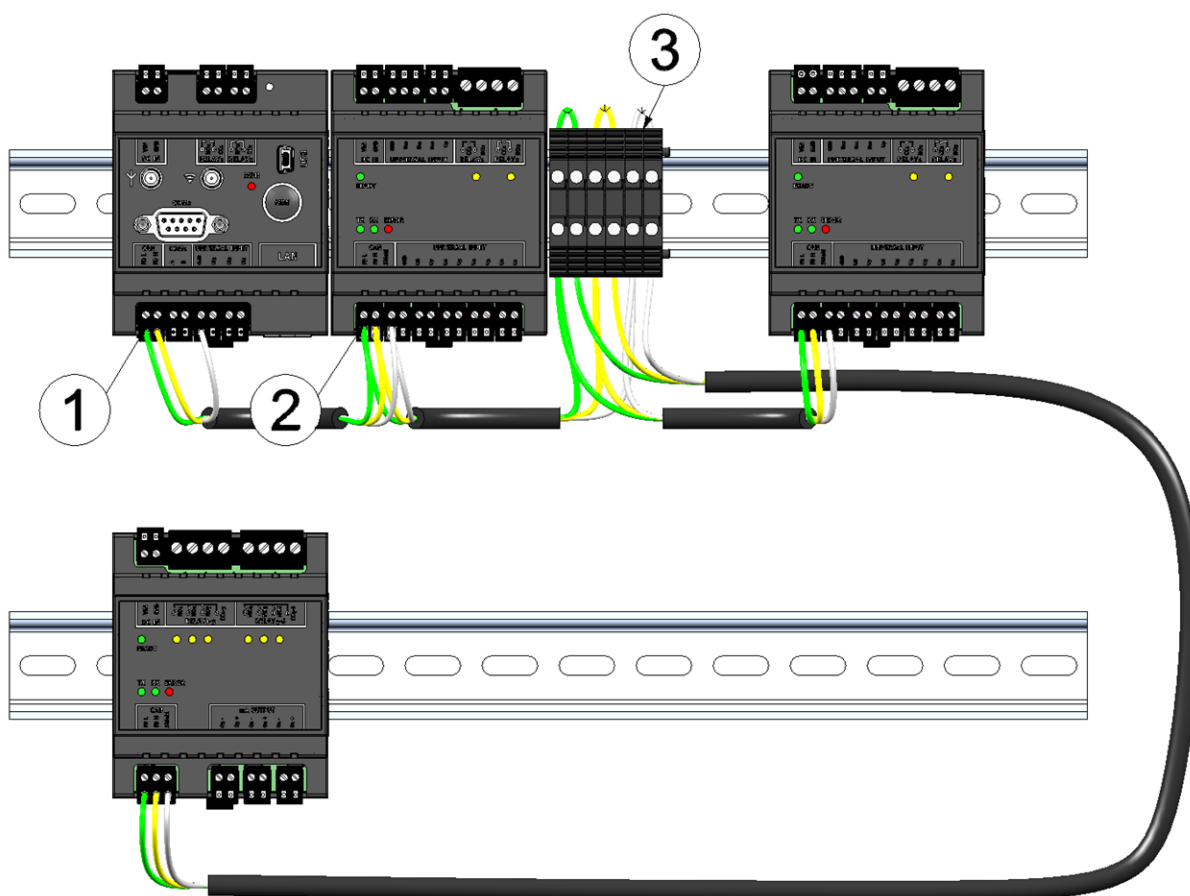
Top-hat rails with myDatalogC33x and extension modules attached

1 myDatalogC33x (120Ω load resistance activated)	4 myDatalogC3e 3mA/6Rel (2k load resistance activated)
2 myDatalogC3e 12UI/2Rel	5 Top-hat rail
3 Distributor terminal	6 myDatalogC3e 12UI/2Rel (120Ω load resistance activated)

- Connect the CAN interface of the myDatalogC33x with those of the extension modules. All of the "FD L" terminals and all of the "FD H" terminals must be connected with one another during this process. Ensure that no current is present!

**Note:** Use a shielded cable to connect the CAN interfaces. The "Shield" terminal is available on the extension modules to connect the cable shield. The cable shield on the myDatalogC33x must be connected to the GND terminal. Use a twin wire end sleeve or distributor terminal if you want to connect several cables to the GND terminal.

**Note:** As there is only one terminal for each signal on the myDatalogC33x and on the extension modules, you should use twin wire end sleeves or distributor terminals if you want to use more than one extension module.

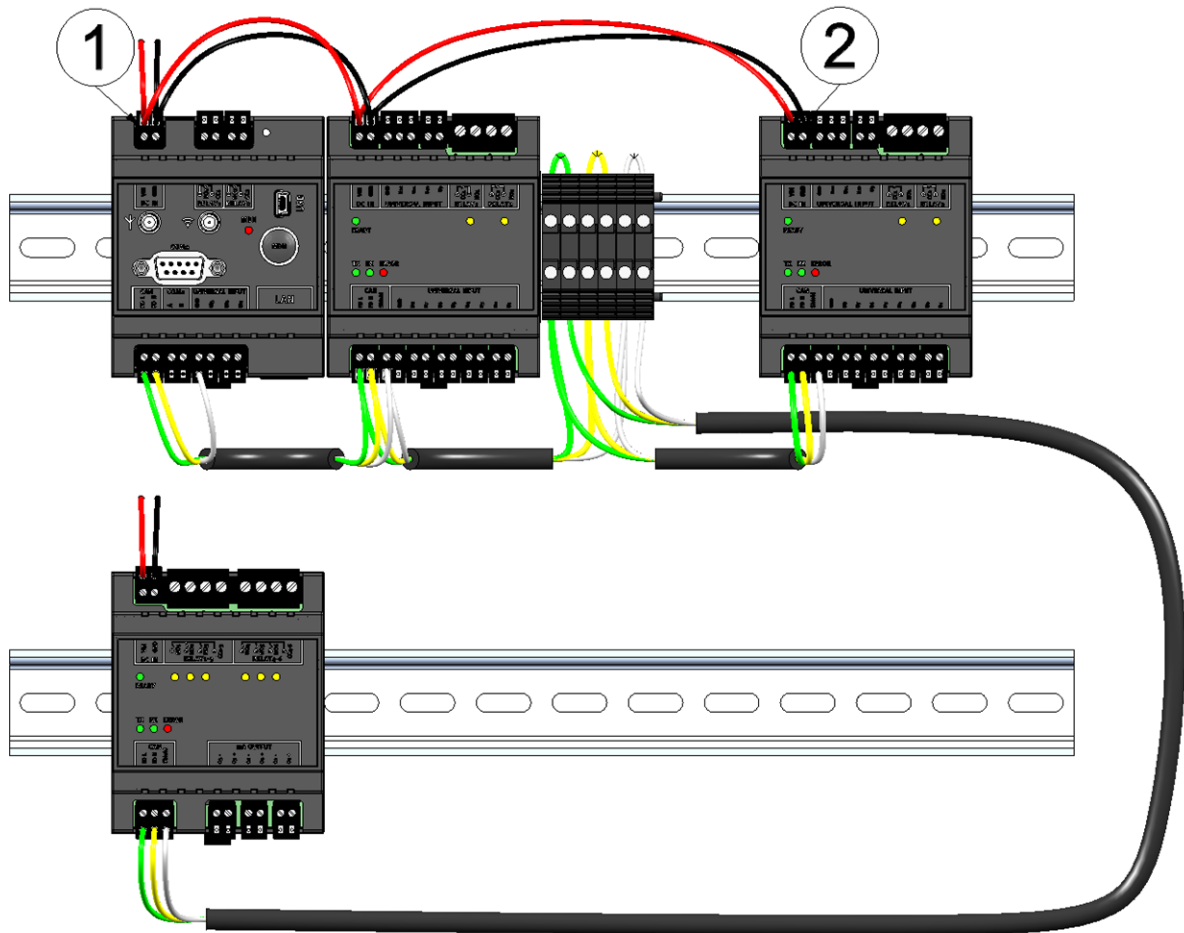


Connecting the cables of the CAN interface

1 Wire end sleeve	3 Twin wire end sleeve (wire end sleeve for two cables)
2 Distributor terminal	

- Connect your sensors and actuators with the inputs and outputs of the myDatalogC33x and the extension module. Ensure that no current is present!

- Connect the cables for the power supply of the myDatalogC33x and the extension modules with the VIN and GND terminals. Ensure that no current is present when establishing the connection. In this case, use twin wire end sleeves or distributor terminals if more than one cable should be connected per terminal.



Connecting the power supply

1 Twin wire end sleeve

2 Wire end sleeve

## 7.4.4 Technical details about the universal inputs

**Note:** The universal inputs are not galvanically isolated.

### 7.4.4.1 0/4...20mA mode

**Note:** Above 23,96mA, the relevant input becomes highly resistive (safety shutdown to prevent damage to the universal input).

Resolution	6,36 $\mu$ A
$I_{max}$	23,96mA
Load	96 $\Omega$

---

#### 7.4.4.2 0...2V mode

Resolution	610 $\mu$ V
U <sub>max</sub>	2,5V
Load	10k086

#### 7.4.4.3 0...10V mode

Resolution	7,97mV
U <sub>max</sub>	32V
Load	4k7

#### 7.4.4.4 Standard digital modes (PWM, frequency, digital, counter)

General	U <sub>max</sub>	32V
	Low	<0,99V
	High	>2,31V
	Load	4k7
PWM	Measurement range	1...99%
	f <sub>max</sub>	100Hz
	Minimum pulse length	1ms
Frequency	Measurement range	1...1000Hz
Counter	Minimum pulse length	1ms

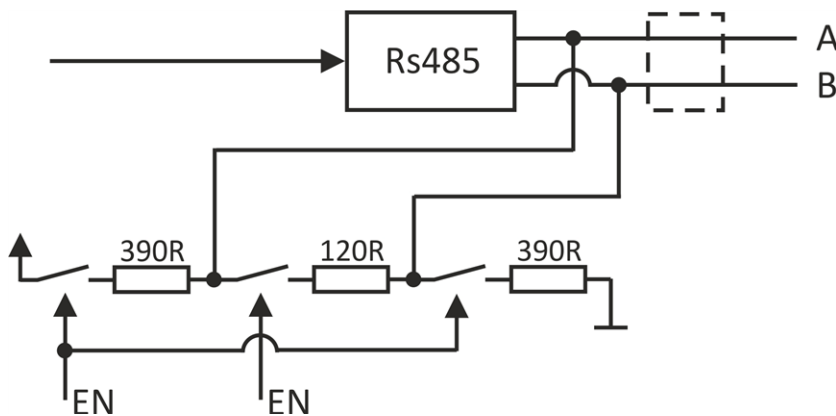
#### 7.4.5 Technical details about the RS485 interface

**Note:** The RS485 interface corresponds to standard EIA-485.

The RS485 interface of the myDatalogC33x includes an input common mode range that covers the full area specified for RS485 (-7V...+12V). Higher voltages damage the interface. Differential signals of more than +/- 200mV within the specified input common mode range are detected correctly. In send mode, the output signal is in the range of 1,5...3,3V .

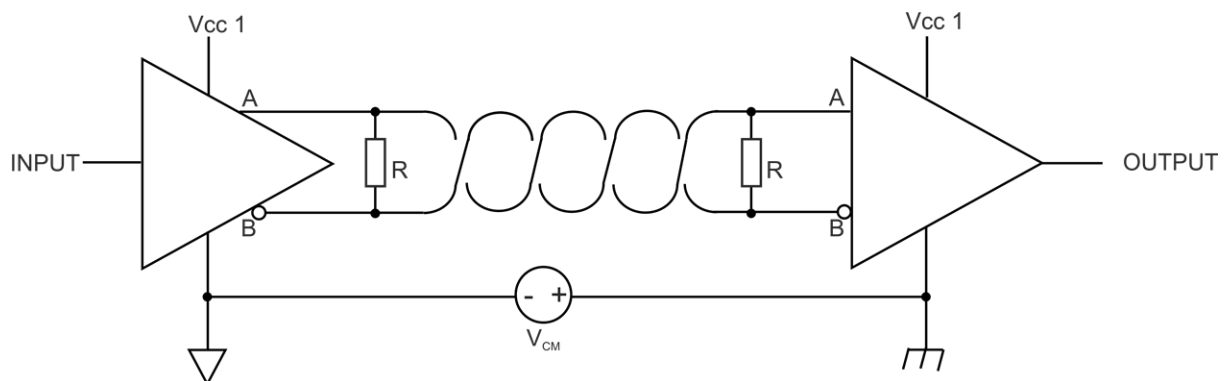
Baud rate	2400-115200
Stop bits	1, 2
Parity	N, E, O
Data bits	7, 8
Load resistance	Off
	120 $\Omega$
Terminal resistance	Off
	390 $\Omega$

The 120Ω load resistance between RS485 A and B can be activated via the device logic. The clamp resistances (pull up to RS485 A and pull down to RS485 B) can also be activated via the device logic.



Schematic diagram of the switchable load resistances

**Note:** Additional explanation regarding the connection of two RS485 bus participants



Schematic diagram: Connection of two RS485 bus participants

A problem occurs if there is no connection between the GND potentials of the sender and recipient. A common mode voltage ( $V_{CM}$ ) occurs in this case. The GND potential difference must not exceed max.  $\pm 7V$ . Higher voltages will damage the interface. Temporary overvoltages (ESD, EFT and surge) are, however, absorbed by protective circuits.

**Note:** The common mode input voltage range of  $-7V \dots +12V$  specified for the RS485 is determined from the max. permissible GND potential difference ( $\pm 7V$ ) and the max. permissible output voltage range of  $0 \dots 5V$  for RS485.

## 7.4.6 Technical details about the CAN interface

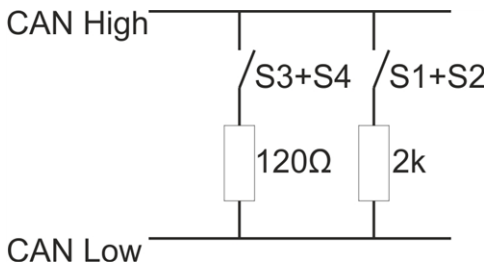
**Note:** The CAN interface of the myDatalogC33x is compatible with standard ISO-11898, including the requirements for 24 V.

Supported CAN specification	CAN: V2.0B CAN FD: V1.0
Data transfer rate	CAN: max. 1Mbit/s/ CAN FD: max. 8Mbit/s

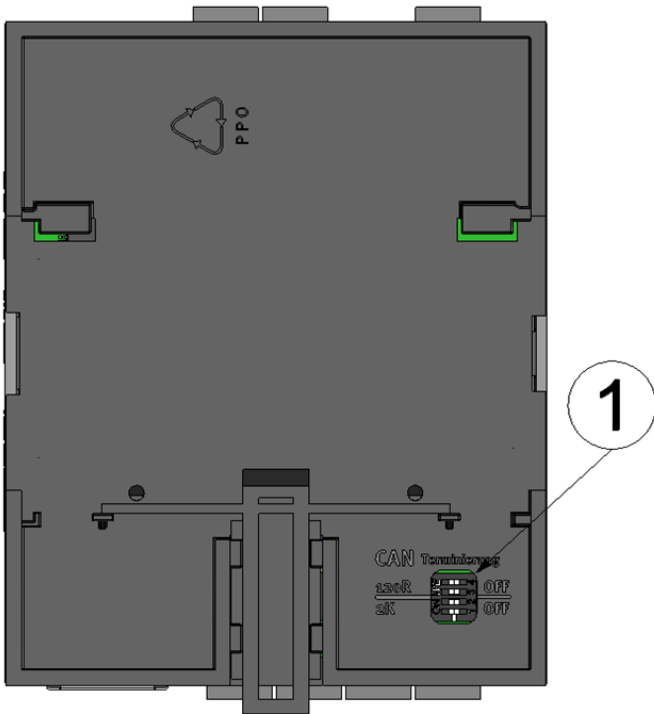
The CAN interface does not have any galvanic isolation between the CAN bus and CAN controller. However, the output drivers of the CAN interface are protected against overloading and are not damaged by a short circuit. It is possible to connect the myDatalogC33x to the CAN bus at the end as well as via the branch lines. If the myDatalogC33x is connected at the end of the bus instead of the prescribed 120Ω load resistance, then the 120Ω resistance integrated in the device must also be activated via S3 and S4 of the dip switch. When establishing a connection via the branch lines, it may be necessary to activate the 2k resistance via S1 and S2 of the dip switch based on the length of the line and the selected data transfer rate. Neither of the two resistances are active in the delivered condition. The dip switch to activate/deactivate the load resistances is located on the rear of the myDatalogC33x .

**Dip switches SW1**

S1 and S2	2k load resistance between CAN high and CAN low
S3 and S4	120Ω load resistance between CAN high and CAN low



Schematic diagram of the switchable load resistances



Position of the dip switch

<b>1</b> Dip switch for activating/deactivating the load resistances for the CAN interface
--

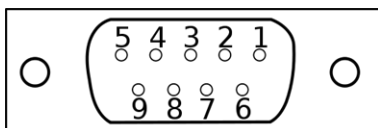
## 7.4.7 Technical details about the RS232 interface

**Note:** The RS232 interface of the myDatalogC33x is compatible with standard TIA/EIA-232-F.

The output drivers are protected against overloading and are not damaged by a short circuit to the GND or +/- 15 V. The inputs are equipped with a 5 k $\Omega$  load resistance.

Baud rate	2400-115200
Stop bits	1, 2
Parity	N, E, O
Data bits	7, 8
Flow control	Off RTS/CTS

The direction of the signals corresponds to that of a DCE (e.g. modem).



9-pole Sub-D(f)

### Assignment of the Sub-D connector

Pin	Signal	Type
1	NC	
2	RXD	O (low: -5,4V ; high: 5,4V )
3	TXD	I (low: <0,8V; high: >2V)
4	NC	
5	GND	
6	NC	
7	RTS	I (low: <0,8V; high: >2V)
8	CTS	O (low: -5,4V ; high:5,4V )
9	5V <sup>1)</sup> max. 750mA	

<sup>1)</sup> Supply voltage (reserved for extensions)

If your sensor also comprises a SUB-D(f) connector, you can use the Gender changer 9-pin D-Sub male/male (206.684) provided as an accessory. If the connection properties (transmission direction of the individual signal lines) of your sensor also correspond to that of a DCE (e.g. modem), you can use the Null modem adapter 9-pin D-Sub female/male (206.686) that is available as an accessory.

---

## 7.4.8 Technical details about the USB interface

The connection to a PC is established via the USB slave interface. It is only designated for the communication with the web-based development environment rapidM2M Studio or the DeviceConfig configuration program. It is not possible to access the USB interface via the device logic. A detailed description of the rapidM2M Studio web-based development environment is provided in chapter "rapidM2M Studio " on page 99. Explanations regarding the functionality of the DeviceConfig configuration program is provided in chapter "DeviceConfig " on page 79.

Access to the web-based development environment rapidM2M Studio is included in the Microtronics Partner Program, for which you can register free of charge at the following address:

**<https://partner.microtronics.com>**

The DeviceConfig configuration program can be downloaded free of charge from the following website:

**[www.microtronics.com/deviceconfig](http://www.microtronics.com/deviceconfig)**

***Important note:*** *If the antenna of the device is earthed or connected to the ground potential of another object (e.g. installation on a control cabinet), remove the antennas before you connect the device with the USB interface of a PC. Otherwise, this can cause a potential displacement between the ground of the antenna and the ground of the PC, which could damage the USB interface of the device.*

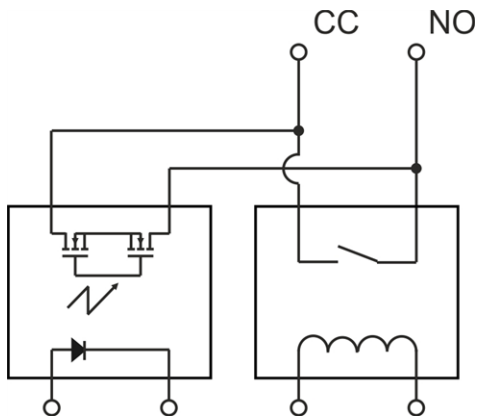


## 7.4.9 Technical details about the outputs

### 7.4.9.1 Isolated switch contact (NO, CC)

**Important note:** The user must ensure that the current on the isolated switch contact does not exceed 2A (relay mode) or 130mA (optoswitch mode).

In idle state, the make contact of the relay or the optoswitch is open (normally open).



Equivalent circuit diagram for the isolated switch contact

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

### 7.4.10 Technical details about the integrated rechargeable buffer battery

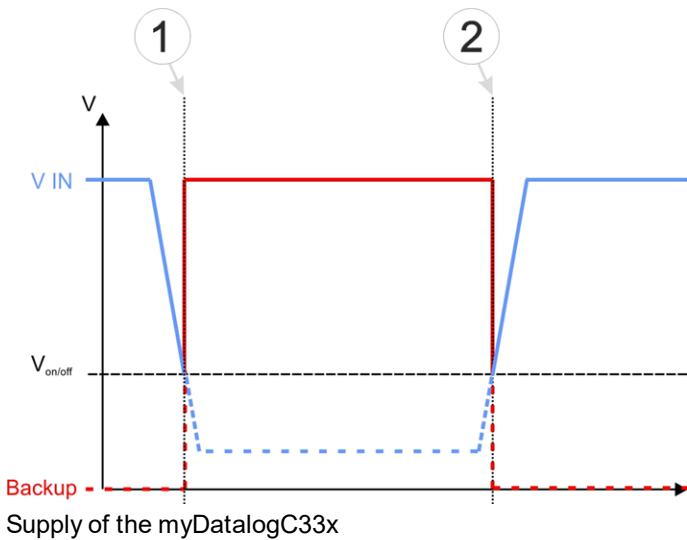
The integrated rechargeable buffer battery enables an application-specific reaction (e.g. issue a message and disconnection from mobile network) to the failure of the power supply. As soon as the supply voltage falls below 8,65V , the power supply is switched to the rechargeable buffer battery. If the supply voltage increases to above 8,65V again, the supply is switched from the rechargeable buffer battery back to the supply voltage.

The following components fail as soon as the myDatalogC33x is only supplied by the rechargeable buffer battery:

- CAN interface (communication with the extension modules is thus also no longer possible).
- Control of the relays (i.e. the closed-circuit contacts switch to "NO" idle mode). The control of the optoswitch is not affected by this.
- 5V supply to COM2 (RS232 interface)

A hardware-regulated controller ensures that the rechargeable buffer battery is only charged when the ambient temperature is in the permitted range (0 ... +45°C ).

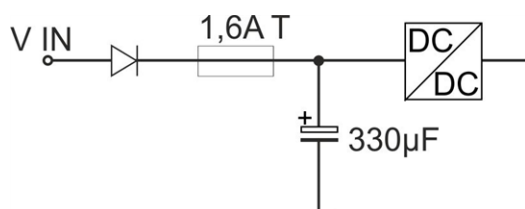
V IN	Supply voltage: 9...32VDC (+/-10%)
V <sub>on/off</sub>	Threshold for switching between normal operation and the rechargeable buffer battery supply: 8,65V
Backup	Voltage of the rechargeable buffer battery



1	<ul style="list-style-type: none"> <li>• The power supply is switched to the rechargeable buffer battery.</li> <li>• The device continues to operate normally (except for the components that fail when supplied by the rechargeable buffer battery).</li> </ul>
2	<ul style="list-style-type: none"> <li>• The supply voltage exceeds the threshold for switching to normal operation again.</li> <li>• The myDatalogC33x resumes normal operation again.</li> </ul>

**Important note:** If the myDatalogC33x is not shut down in a controlled way via the device logic following the execution of the application-specific reaction (e.g. issue a message and disconnection from mobile network) to the failure of the power supply, the device logic continues to be executed until the integrated rechargeable buffer battery is completely discharged. This operating state should be avoided as the integrated rechargeable buffer battery could be damaged by being completely discharged.

### 7.4.11 Technical details about the energy supply



Schematic diagram of the energy supply

V IN	9...32VDC (+/-10%)
Power consumption <sup>1)</sup> (without sensors)	typ. 5W max. 9W
Input capacity	330µF
Fuse	1,6A T
Reverse voltage protection	Yes

<sup>1)</sup> applies to ongoing operation. A current peak is caused by the input capacity at the time of activation.

The myDatalogC33x is equipped with a relatively large input capacity (330µF ) to ensure reliable switchover to the integrated rechargeable buffer battery in the event of a supply voltage failure. When selecting the power supply please ensure that it is able to supply the required initial current. A selection of compatible power supply units is included in the chapter "Power supply" on page 235. The supply voltage input is also equipped with a diode to protect against polarity reversal and a 1,6A T fuse.

### 7.4.12 Technical details about the system time

The myDatalogC33x is equipped with a hardware real-time clock that has its own buffer battery with an expected service life of >10 years. The system time continues to run even if the power supply unit is removed. This means that following recommissioning, valid time stamps for the measurement and log data can be generated immediately. Additionally, the system time is synchronised with the server each time a connection to the myDatatnet server is established.



# Chapter 8 Initial Start-Up

## 8.1 User information

Before you connect the myDatalogC33x and place it into operation, you must observe and comply with the following user information!

This manual contains all information that is required for using the device.

Is intended for technically qualified personnel who have the relevant knowledge and experience in the area of measurement technology.

Read this manual carefully and completely in order to ensure the proper functioning of the myDatalogC33x .

Contact Microtronics Engineering GmbH(see "Contact information" on page 247) if anything is unclear or if you encounter difficulties with regard to installation, connection or configuration.

## 8.2 Applicable documents

In addition to this operating instructions, additional instructions or technical descriptions may be required for the installation, commissioning and operation of the entire system.

These instructions are enclosed to the respective additional devices or sensors or are available for download on the Microtronics website.

## 8.3 General principles

The entire measurement system may only be placed into operation after completion and inspection of the installation. Study the manual thoroughly before placing into operation to prevent faulty or incorrect configuration.

Utilise the manual to familiarise yourself with the operation of the myDatalogC33x and the input screens of the myDatanet server before you begin with the configuration.

## 8.4 Commissioning the system

**Note:** It is recommended that the myDatalogC33x is first placed into operation in the office before mounting the device permanently at the place of use. During this process, you should create a site for the later operation on the myDatanet server (see "Creating the site" on page 95) and determine a site configuration (including data descriptor and device logic) (see "Site configuration" on page 74). If you create the site based on an IoT application (see "myDatanet Server Manual " 805002), the data descriptor and device logic are taken from the IoT application and do not need to be defined separately. Take the opportunity to get to know the functions of the device in a stable environment. You can also use suitable test signals to simulate the sensors to establish the optimum configuration of the myDatalogC33x prior to its actual first use. This reduces the amount of time required for on-site installation to a minimum.

---

The following work should be completed in the office before you go to the future location of the device:

1. If necessary, create a customer on the myDatanet server (see "myDatanet Server Manual " 805002).
2. Within the selected customer, create a site/application for operation on the myDatanet server (see "Creating the site" on page 95).

**Note:** A "rapidM2M " type site or a site based on an IoT application that is compatible with the "rapidM2M " site type must be created to operate the myDatalogC33x .

3. Configure the created site/application according to your requirements (see "Site configuration" on page 74). If the site was not created based on an IoT application, you must determine the data descriptor and device logic via the "Control" configuration section (see "Control" on page 75).
4. Connect the antenna (see "Connection of the GSM antenna" on page 50).
5. Trigger a connection establishment so that the configuration of the site/application is transferred to the myDatalogC33x . If no device logic has been loaded into the device yet then you can achieve this by establishing the power supply (see "Connecting the sensors, actuators and power supply" on page 46). If a device logic has already been loaded into the device, execute the operations provided in the device logic to trigger the establishment of a connection.

**Note:** You can also skip this step, as a connection must be established during the installation on site, which transfers the configuration settings to the myDatalogC33x at the same time.

6. Then disconnect the cables of the supply voltage from the device preferably in a de-energised state. Ensure that the myDatalogC33x is completely deactivated. To do so press the reset button directly on the device, unless you have provided a routine in your device logic for controlled shut-down of the system following disconnection of the supply voltage.
7. Remove the antenna again.

The following tasks are to be completed on site, directly at the deployment site of the device:

8. Complete all of the steps detailed in the chapter "Connecting the sensors, actuators and power supply" on page 46.
9. Check whether the connection to the myDatanet server has worked correctly (see "Testing communication with the device" on page 70).

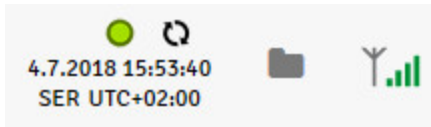
## 8.5 Testing communication with the device

1. Create a site/application for operation on the myDatanet server (see "Creating the site" on page 95).

**Note:** A "rapidM2M " type site or a site based on an IoT application that is compatible with the "rapidM2M " site type must be created to operate the myDatalogC33x .

2. Configure the created site/application according to your requirements (see "Site configuration" on page 74). If the site was not created based on an IoT application, you must determine the data descriptor and device logic via the "Control" configuration section (see "Control" on page 75).
3. Connect the antenna (see "Connection of the GSM antenna" on page 50).

4. Establish a connection. If no device logic has been loaded into the device yet then you can achieve this by establishing the power supply (see "Connecting the sensors, actuators and power supply" on page 46). If a device logic has already been loaded into the device, execute the operations provided in the device logic to trigger the establishment of a connection.
5. Wait until the measurement instrument list indicates that the device is connected to the server (rotating arrows).



With the exception of the "Online" connection type (see "rM2M\_TxSetMode()"), the time during which the myDatalogC33x is connected to the server is very short. It can therefore also be checked whether the time stamp of the last connection (under the green status symbol) has been updated.

The following steps are only necessary, if you simultaneously want to test the measurement value acquisition and data transmission.

6. Complete all of the steps detailed in the chapter "Connecting the sensors, actuators and power supply" on page 46. This includes connecting the sensors.

**Important note:** All wiring work must be performed in the de-energised state.

7. You can use the "Reports" of the myDatatnet server to check the data transmission (see "myDatatnet Server Manual " 805002). The configuration of the Data Descriptor (see "Data Descriptor " on page 207) is required for this purpose.
8. Once you have completed the necessary preparations, initiate a transmission directly on the device if you have included this in your device logic. If you have not included an option to trigger a transmission, wait for the next scheduled data transmission.
9. Evaluate the incoming data.



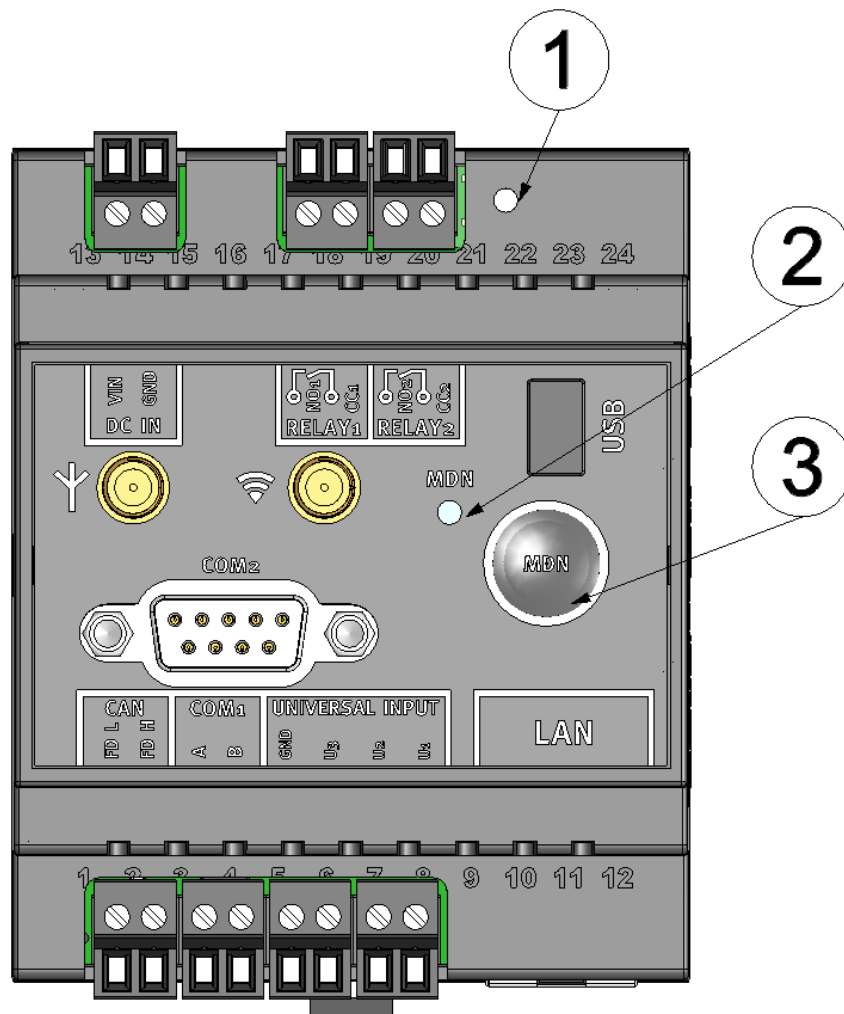


# Chapter 9 User interfaces

The configuration of the myDatalogC33x is carried out via the web interface on the myDatatnet server (see "User interface on the myDatatnet server" on page 74), which your responsible sales partner will provide to you.

## 9.1 User interface on the myDatalogC33x

### 9.1.1 Operating elements



Operating elements

<p><b>1</b> Reset button</p>	<p><b>3</b> MDN button (freely useable, evaluated by the device logic)</p>
<p><b>2</b> RGB LED (freely useable, controlled by the device logic)</p>	

---

## 9.2 User interface on the myDatanet server

### 9.2.1 Site configuration

*Note: Depending on the respective user level, some of the configuration fields mentioned in the following sub-chapters may be hidden. In this case, please contact the administrator of the myDatanet server.*

Click on the name of the site in the list of sites to open the input screen for configuring the site (see "myDatanet Server Manual " 805002).

#### 9.2.1.1 Site

##### Customer

*Specifies to which customer the site is assigned*



##### symbol

*Assign site to another customer*

##### Name

*Site designation (not relevant for the device or data assignment) [2-50 characters]*

##### Device S/N

*Serial number of the device that is linked to the site (device assignment!)*

##### Application

*Name of the IoT application based on which the site was created*

##### Application version

*Version number of the IoT application that is currently installed on the site. If the version number of the site is not the same as the version number of the device logic installed on the device then the version number of the device logic installed on the device is displayed in addition to the version number of the site.*

##### Tags

*List of tags that are already assigned to the site. This assignment can be cancelled by clicking on the cross next to the title of the tag. The input screen for assigning tags can be opened by clicking on the plus symbol.*

#### 9.2.1.2 Comments

##### Comments

*Free comment field (is also displayed below the device type in the site/application list)*

### 9.2.1.3 Control

**Note:** This configuration section is not visible if this site was created based on an IoT application (see "myDatanet Server Manual " 805002).

Device logic type	Off	Device logic deactivated	
	Pawn	Activates device logic processing	
Device logic source	Pawn source code	Device logic	Input window for editing the device logic that is loaded into the myDatalogC33x (see "Device Logic" on page 105)
	Upload a compiled script	File upload	Selection of the device logic binary file (*.amx) that is uploaded to the myDatanet server and is loaded into the myDatalogC33x during the next connection. The file path is only displayed as long as the input screen for configuring the site has not been closed.
Data descriptor	Input window for configuring the Data Descriptor (see "Data Descriptor " on page 207)		

### 9.2.1.4 Configuration 0 - Configuration 9

**Note:** These configuration sections are only visible if the logical structure of the corresponding configuration data block was defined using the Data Descriptor (see "Data Descriptor " on page 207). The name of the configuration section is also defined via the Data Descriptor .

These configuration sections ensure that the parameters from the customer's freely definable, independent memory blocks can be edited and displayed via the interface of the myDatanet server. For this purpose, the logical structure of the configuration data blocks must be defined with the help of the Data Descriptor (see "Data Descriptor " on page 207).

### 9.2.1.5 Alarm settings

Acknowledgement	Standard	The global server setting is used to determine whether alarms must be acknowledged automatically or manually.
	automatic	Alarms are acknowledged automatically as soon as all of the messages have been sent. If SMS that have a tariff with a delivery confirmation function have also been sent, acknowledgement is provided after delivery confirmation.
	manual	Alarms must be acknowledged by the user.
Transfer volume	Standard	The setting for the transfer volume alarm is taken from the global server settings.
	off	The transfer volume alarm is deactivated.
	individual	The level at which the transfer volume alarm should be triggered can be entered in the adjacent field in KiB.
Offline alarm after	alarm in the event that the device does not report for longer than the set time (00:00 alarm deactivated).	
Title user alarm 1	Freely selectable title for user-defined alarm 1. If the user-defined alarm 1 is triggered by a device connected to the site, the server will use this text to signal the alarm. The same applies to user-defined alarm 2 and 3.	
Title user alarm 2	Freely selectable title for user-defined alarm 2	
Title user alarm 3	Freely selectable title for user-defined alarm 3	

### 9.2.1.6 Basic settings

Time zone	Regional settings (not relevant for raw measurement data as this is stored in UTC)	
Daylight saving time	Configuration for automatic time adjustment	
	Standard	The configuration for the time adjustment is adopted by the global server setting.
	Off	Automatic time adjustment deactivated
	USA	Predefined setting for the American area
	EU	Predefined setting for the European area
Default report	Selection of the report that is loaded by clicking on the device link in the maps	
	Off	No report is loaded.
	"Name of a report"	The selected report is loaded.
Report template	Selection of the report template used to display the data when clicking on the symbol to display the measurement data, which is located in the list of sites/applications. Only the report templates in which the site/application type of the first wild card is compatible with the site/application that is currently being edited are displayed in the dropdown list. The symbol to display the measurement data is only displayed in the list of sites/applications if a report template has been selected.	
	(not assigned)	The symbol to display the measurement data is not displayed in the list of sites/applications.
	"Name of a report template"	Name of the report template used to display the measurement data
Change log configuration	Selection of which changes to the configurations should be logged	
	web api	Changes that were implemented via the server interface or REST-API are logged.
	web device api	Changes that were implemented via the server interface, by the device itself or the REST-API are logged.

### 9.2.1.7 FTP export settings

**Note:** This configuration section is only visible if the "FTP Agent Extended" licence for the myDatanet server has been enabled.

FTP export profile	off	FTP export deactivated
	"Name of an FTP export profile"	List with the FTP export profiles that were created on the myDatanet server (for creating an FTP export profile, see "myDatanet Server Manual " 805002).
Settings of the selected profile	Shows an overview of the most important parameters of the selected FTP export profile	
FTP directory	Makes overwriting the standard directory of the selected FTP export profile possible [0-100 characters]	
Last export	Time stamp of the last FTP export	

## 9.2.2 Device configuration

**Note:** Several of the configuration fields in the following sub chapters may possibly be hidden depending on the respective user level. In this case, contact the myDatanet server administrator.

You can reach the input screen for configuring the device by clicking on the serial number in the list of sites/applications (see "myDatanet Server Manual " 805002) or by clicking on the device name in the device name list (see "myDatanet Server Manual " 805002).

### 9.2.2.1 Comments

#### Comments

Free comment field (is also displayed below the site name in the site/application list)

### 9.2.2.2 Measurement instrument

Customer	Name of the customer to whom the measurement instrument is assigned
Tags	List of the tags that are already assigned to the measurement instrument. This assignment can be cancelled by clicking on the cross next to the title of the tag. The input screen for assigning the tags is opened by clicking on the plus symbol. This enables existing tags to be assigned and new tags to be created.
Serial number	Serial number of the instrument
Instrument class	The instrument class of the site and instrument must match for an instrument to be able to be connected to a site. Once the instrument has been created via the server interface, the instrument class can only be changed up until the first connection of the instrument to the server. If an instrument class, that does not match the actual class of the instrument, is selected when the instrument is created it is automatically corrected during the first connection.
Telephone number	Telephone number of the SIM card. The control SMS messages (e.g. wakeup) are sent to this number. Format: +43555837465
Instrument flags	Additional information regarding the instrument class (for internal use)
Firmware version	Current software version installed on the measurement controller
Last connection	In each case, the last time stamp of the affected operation
Last wakeup	
Last disconnection	
Last transmission error	
Last Aloha connection	
Wakeup SMS count	Number of wakeup SMS sent to this device since the last connection. This counter is reset at/during each successfully established connection.

Device Logic sync	Productive	If the Device Logic installed on the device and saved on the server do not match, the Device Logic saved on the server is loaded in to the device.
	Development (sync)	The Device Logic on the device and server are synchronised. The one with the latest time stamp is transferred to the other one.
	Development (no sync)	The Device Logic on the device and server are not synchronised.
Firmware update	Off	Firmware update is deactivated.
	On	As soon as a new version of the selected firmware type is available, this is installed immediately.
	Even if tag is missing	Firmware is also transferred to the device if the device has not transmitted the current firmware version to the server (NOT RECOMMENDED!).
	Allow downgrade	Facilitates the installation of an older firmware version than the one on the device (NOT RECOMMENDED!)
	Once	Performs a single firmware update. If no new firmware is available or the firmware was installed successfully, the firmware update is automatically switched to "OFF".
	Ignore	The firmware update is deactivated and no information is provided about available firmware updates.
Firmware type	Released	Only firmware versions that have successfully undergone internal and field testing are installed (this practically eliminates malfunctions).
	Release candidate	Only firmware versions that have successfully undergone internal testing are installed (malfunctions cannot be excluded).
	Beta release	Even firmware versions that have not successfully undergone all of the internal tests are installed (malfunctions may occur).
Identification	String specifying the hardware platform implemented in the device and the corresponding hardware version (i.e. the rapidM2M module identification).	
Prod. rev.	Product rev. of the myDatalogC33x	

### 9.2.2.3 GPRS

#### SIM tariff

*Selected SIM tariff*

---

# Chapter 10 DeviceConfig

## 10.1 General

The DeviceConfig configuration program can be downloaded free of charge from the following website:

**[www.microtronics.com/deviceconfig](http://www.microtronics.com/deviceconfig)**

The tool is used for configuration, maintenance, fault analysis and synchronisation purposes. It is compatible with all myDatanet devices equipped with a USB interface, wireless M-bus interface or Bluetooth Low Energy.

The requirements regarding configuration and maintenance vary depending on the type of device. To ensure simple and intuitive operation, the user interface of the DeviceConfig therefore automatically adjusts to the relevant device that is connected. In addition to the standard functions, the tool also supports device-specific processes (e.g. calibration, zero point adjustment).

The DeviceConfig enables you to complete the following tasks:

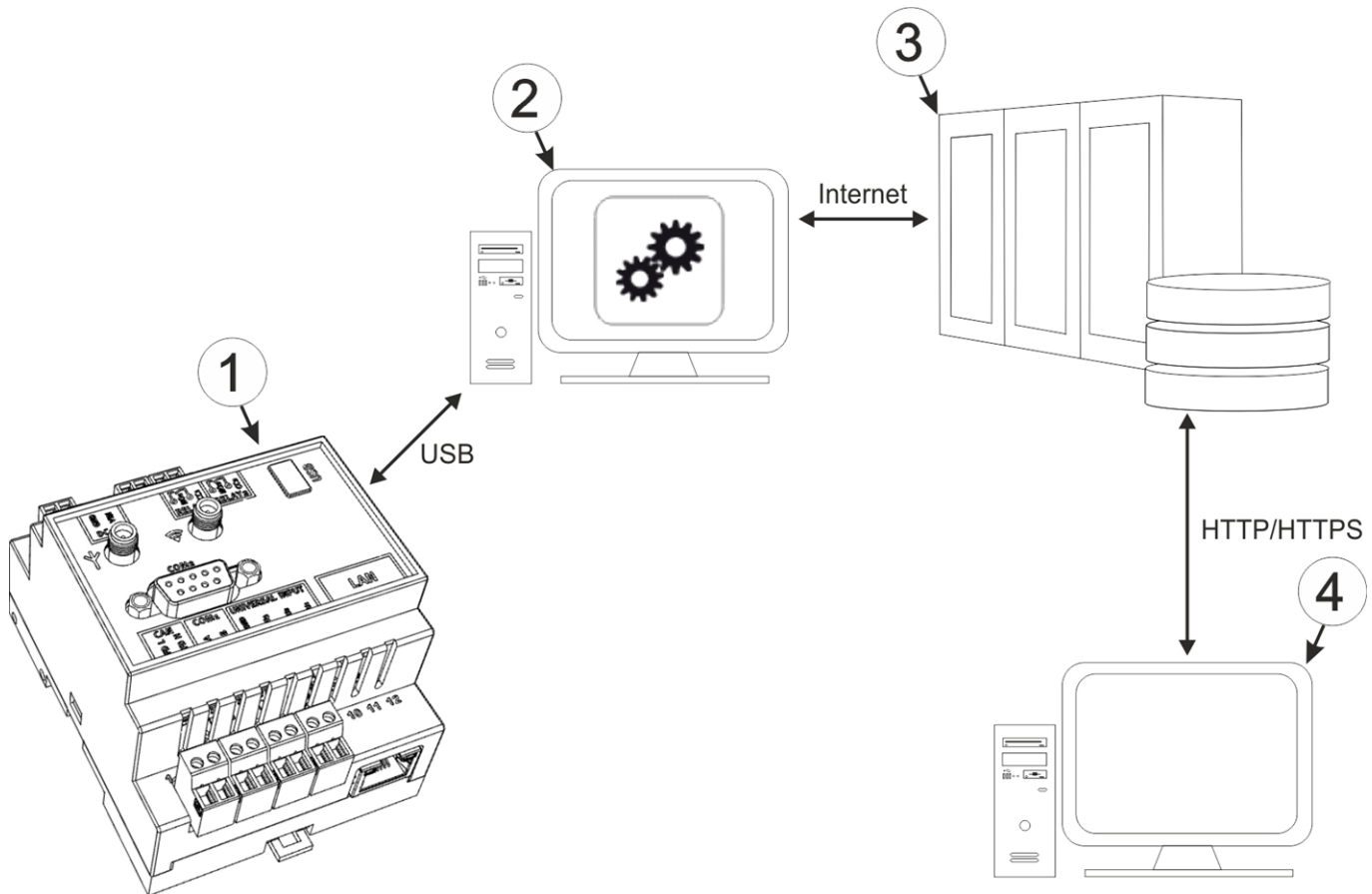
- Synchronisation of measurement data and configuration between device and server (specifically for devices without GSM/GPRS modem)
- Basic configuration of the device (e.g. measurement and transmission cycle)
- Read out and analysis of the device log
- Calibration, trimming and zero point adjustment (special knowledge and password required)
- Update the firmware

## 10.2 Prerequisites

Interfaces	1 x USB
Operating system	Win XP Windows Vista Windows 7 Windows 8 Windwos 10
Internet connection	Recommended
Required disk space	approx. 50 MB

## 10.3 Functional principle

The following description specifically refers to the use of the configuration program DeviceConfig in conjunction with the myDatalogC33x .



Functional principle

<b>1</b> myDatalogC33x	<b>3</b> myDatanet server
<b>2</b> PC with the DeviceConfig configuration program installed	<b>4</b> Client that accesses the interface of the myDatanet server via the web browser

**Important note:** The USB interface is a service interface that must be protected from contamination when not in use by the sealing plug included in the package.

The configuration program DeviceConfig communicates directly with the myDatalogC33x via a USB connection. The functions provided with the DeviceConfig configuration program include:

- Read out and analysis of the device log (see ""Log" tab" on page 87)
- Update the firmware(see ""Firmware" tab" on page 89)



---

## 10.4 Installation

The following chapter describes the installation process in Windows 7.

1. Execute the "*InstDeviceConfig.exe*" file to start the installation process.

**Note:** Only connect the device or USB BLE-Adapter (300685) to your PC once the installation process has completed as the required drivers are only installed during this process.

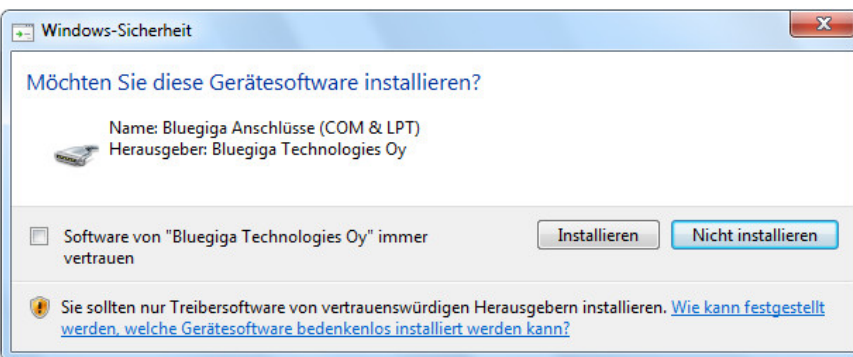


DeviceConfig setup wizard

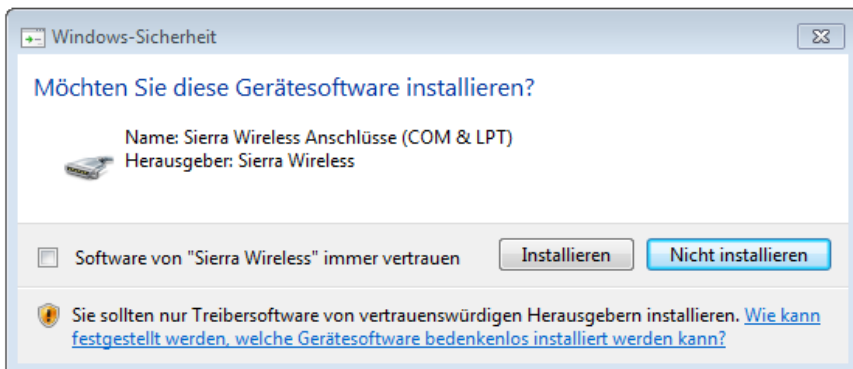
2. Follow the instructions of the setup wizard until the following screen is displayed. The following drivers must be installed to ensure correct operation.



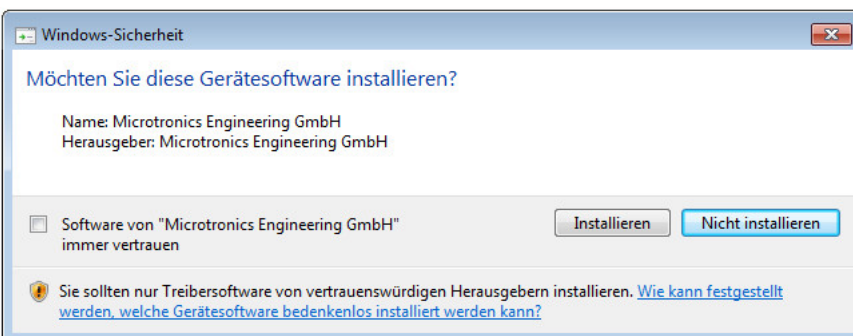
Installation of the USB drivers for the devices



Installation of the drivers for the USB BLE-Adapter

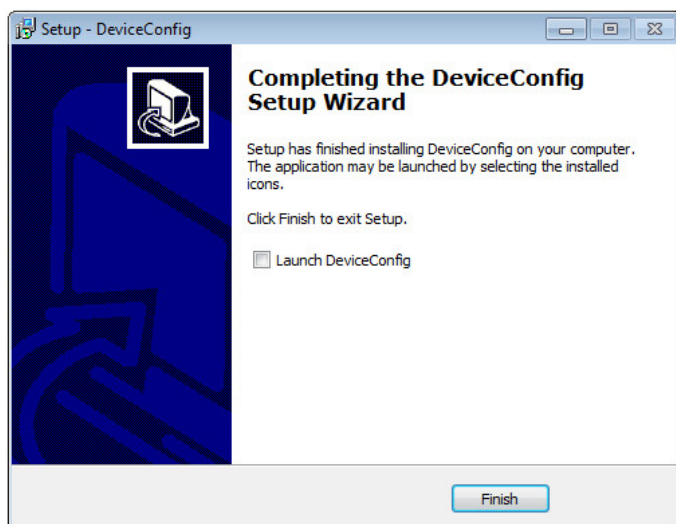


Installation of the USB drivers for the devices on a M1 basis



Installation of the USB drivers for the devices on a M2/M3 basis

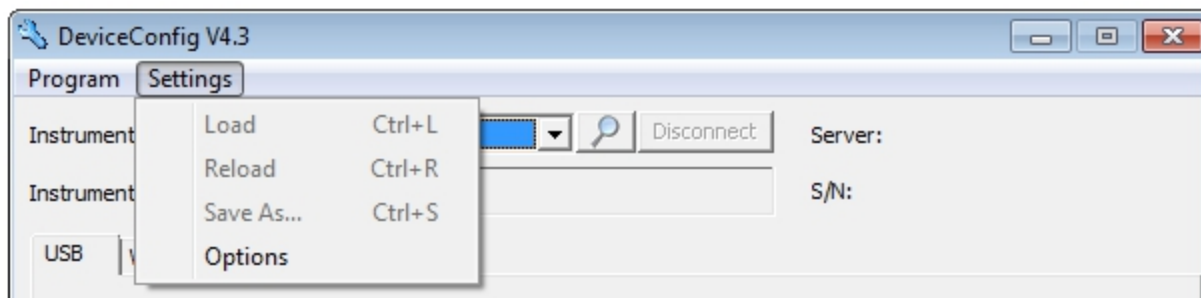
- 
3. Once the following screen is displayed, close the installation process by clicking on the "Finish" button.



Complete the setup

## 10.5 Menu of the DeviceConfig

### 10.5.1 Settings

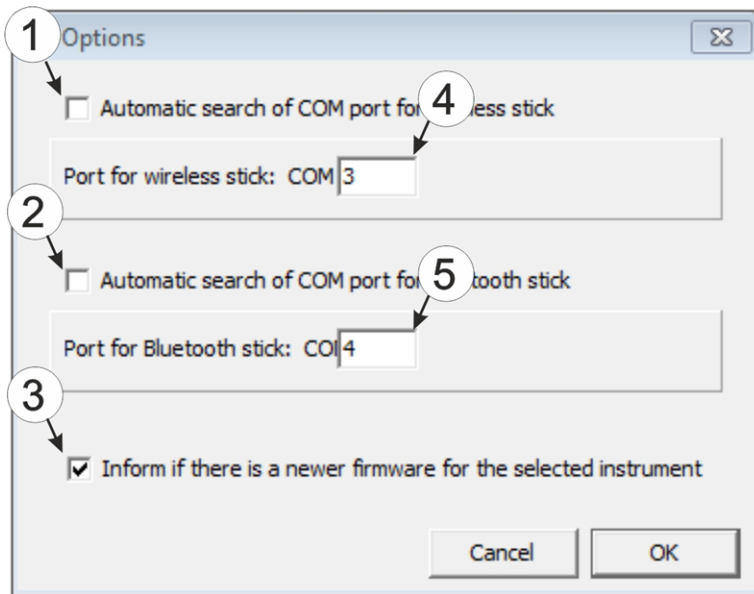


"Settings" menu item

#### 10.5.1.1 Options

The settings for the COM ports to which the USB radio transmitter (206.657) or the USB BLE-Adapter (300685) are connected can be specified and the automatic search for the available firmware versions can be activated or deactivated via the "Settings -> Options" menu item.

The USB radio transmitter (206.657) is required for myDatanet devices that are connected to the PC via a wireless M-bus, while the USB BLE-Adapter (300685) is required for devices that are connected to the PC via Bluetooth Low Energy. Information on whether your device supports one of these connection methods is provided in the user manual for the respective device.

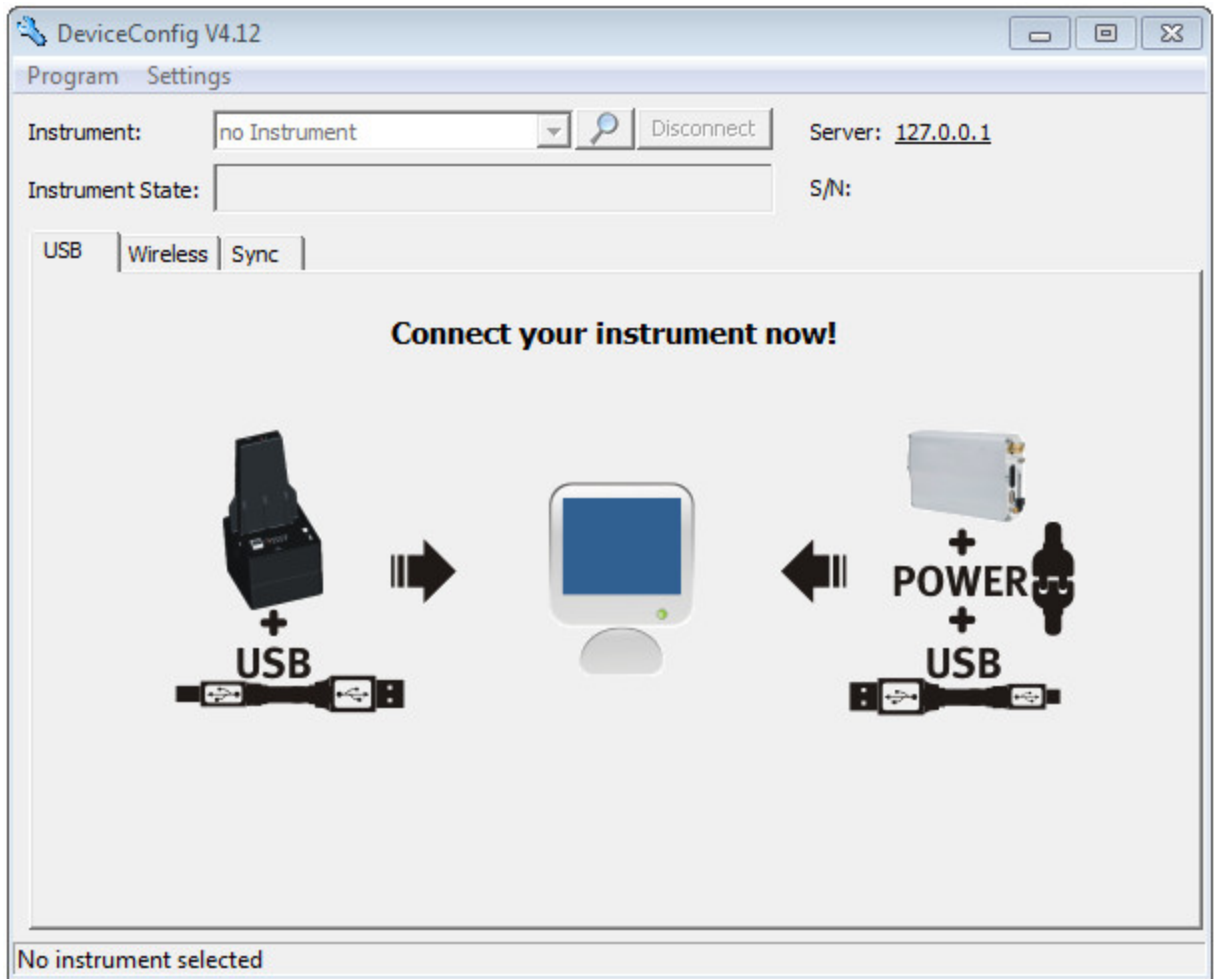


"Settings -> Options" menu item

<p><b>1</b> Activates/deactivates the automatic search for the USB radio transmitter (206.657) on all of the available COM ports</p>	<p><b>4</b> COM port that is connected with the USB radio transmitter (206.657) (only visible when the automatic search is deactivated)</p>
<p><b>2</b> Activates/deactivates the automatic search for the USB BLE-Adapter (300685) on all of the available COM ports</p>	<p><b>5</b> COM port that is connected with the USB BLE-Adapter (300685) (only visible when the automatic search is deactivated)</p>
<p><b>3</b> Activates/deactivates the automatic search for available firmware versions</p>	

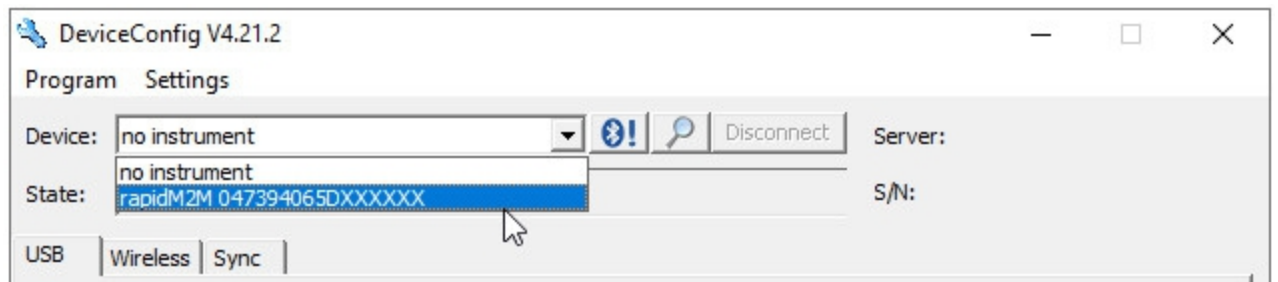
## 10.6 Connecting a Device via USB

1. Start the DeviceConfig configuration program.



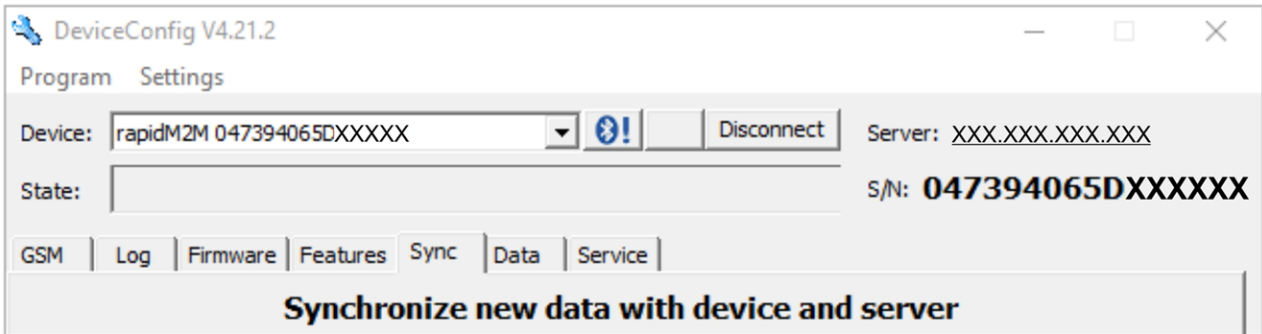
DeviceConfig

2. Connect the myDatalogC33x to the PC using a USB cable.
3. Select your device based on the serial number from the list of devices found.



List of devices found

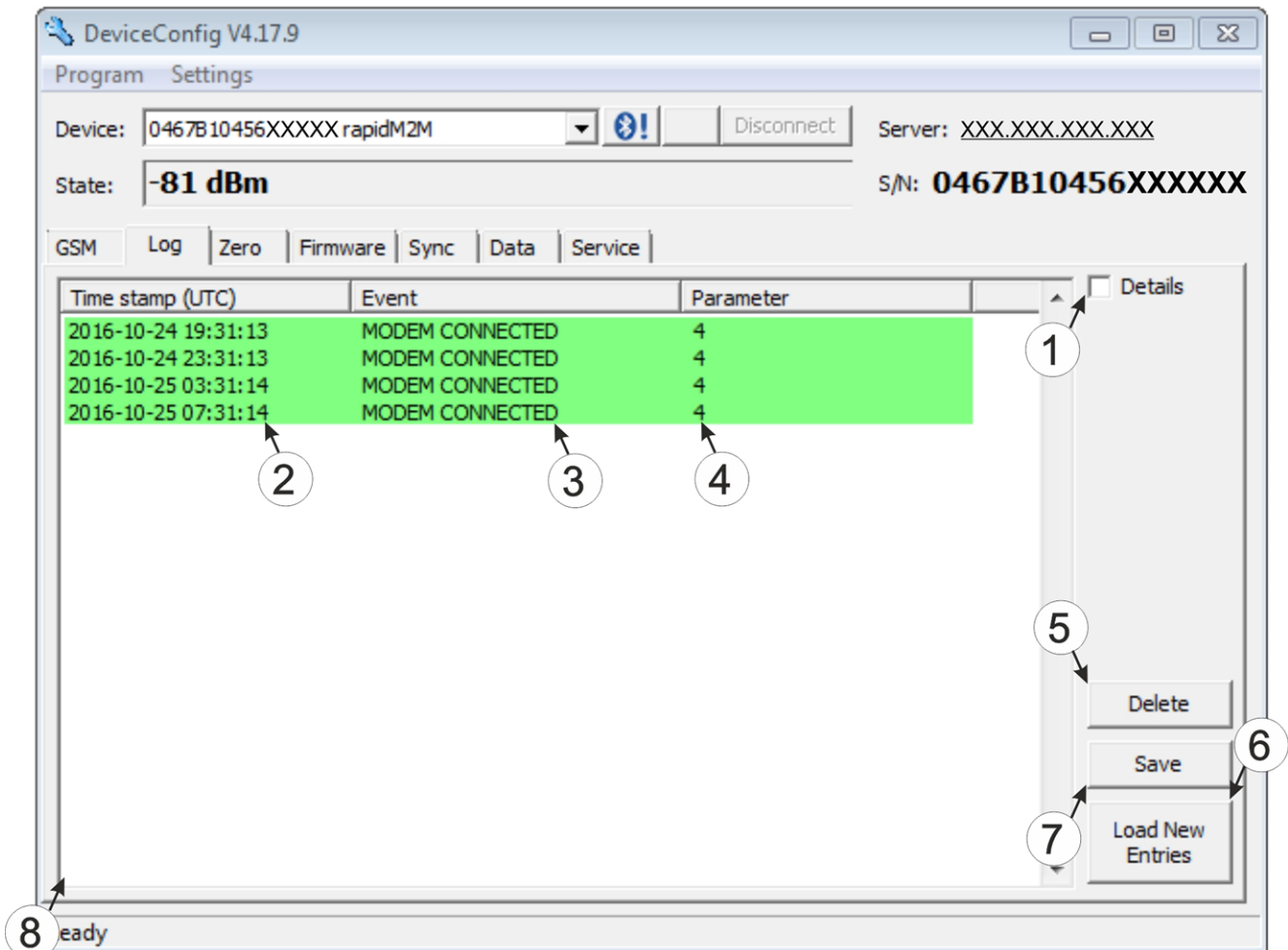
- 
4. Wait until the DeviceConfig has received the configuration of the device. Depending on the device, additional tabs may be displayed.



"Sync" tab when actively connected to the myDatalogC33x

## 10.7 "Log" tab

This tab is designed to manage log entries. It enables the entries to be loaded from the myDatalogC33x , to be saved as a \*.tsv file and entries to be deleted from the memory of the myDatalogC33x .

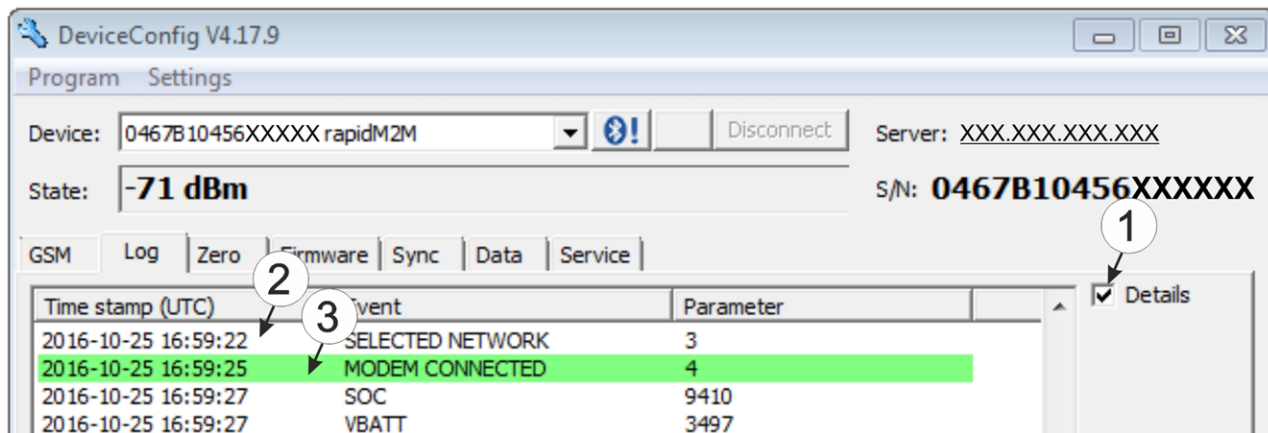


"Log" tab

1 Activates the detailed display of the log entries	5 Deletes the log entries from the memory of the device
2 Time stamp of the log entry	6 Loads the log entries from the device
3 Log entry	7 Saves the loaded log entries as a tsv file
4 Parameter of the log entry	8 Window to display the loaded log entries

The coloured highlighting indicates how crucial the log entry is. The white, informative log entries are only displayed when the detailed display of the log entries is activated (see ""Log" tab with detailed view activated" on page 88).

Colour	Evaluation
White	Information regarding the current operating state
Green	
Light blue	
Blue	
Purple	
Grey	
Yellow	Uncritical error
Red	Critical error



"Log" tab with detailed view activated

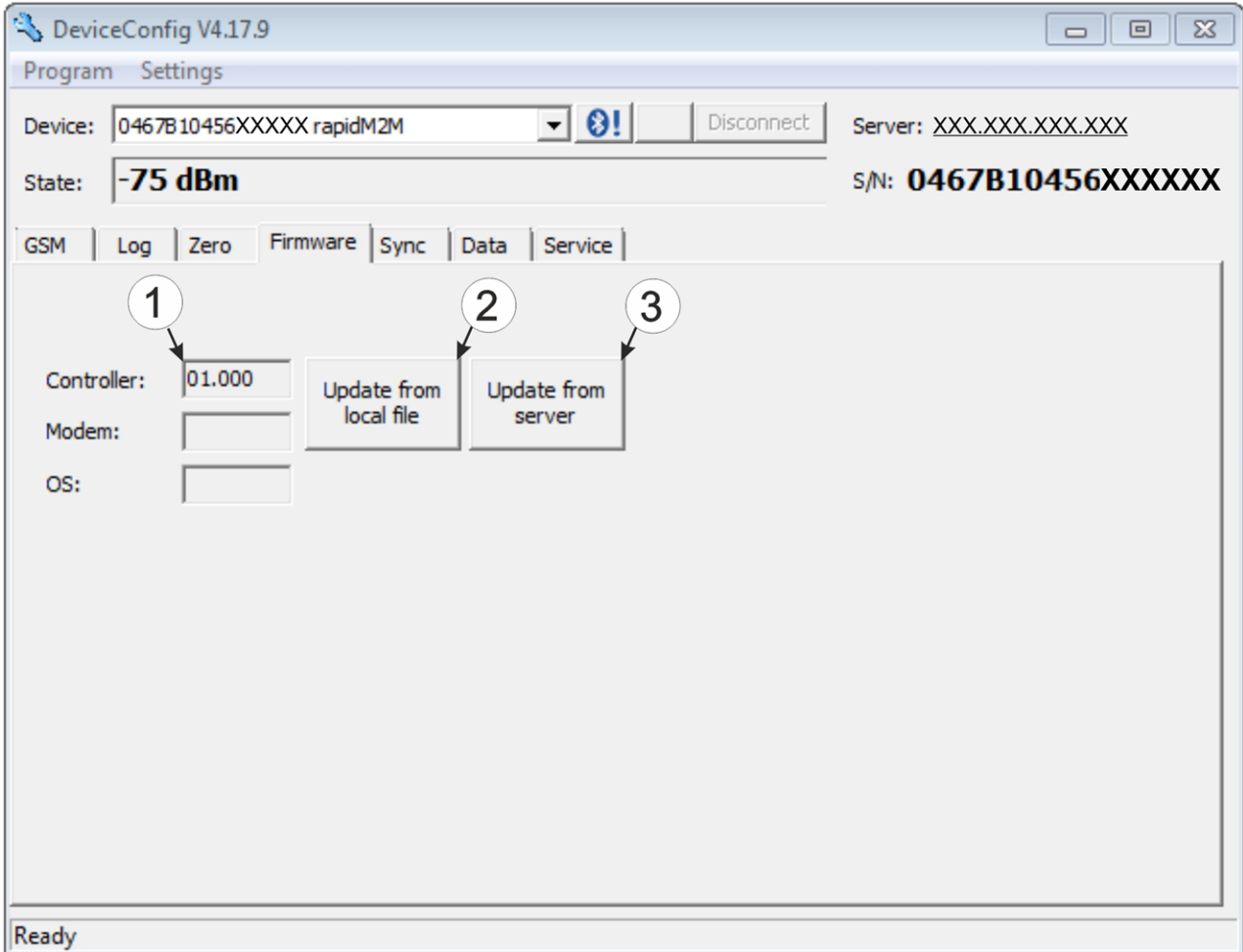
1	Activates the detailed display of the log entries	3	Log entry that is always displayed
2	Informative log entry that is only visible if the detailed display is activated		



## 10.8 "Firmware" tab

This tab enables firmware to be installed directly via the USB interface or the Bluetooth Low Energy interface. There are two available ways to update the firmware:

- Using a previously downloaded firmware package
- By directly loading from the myDatenet server



"Firmware" tab

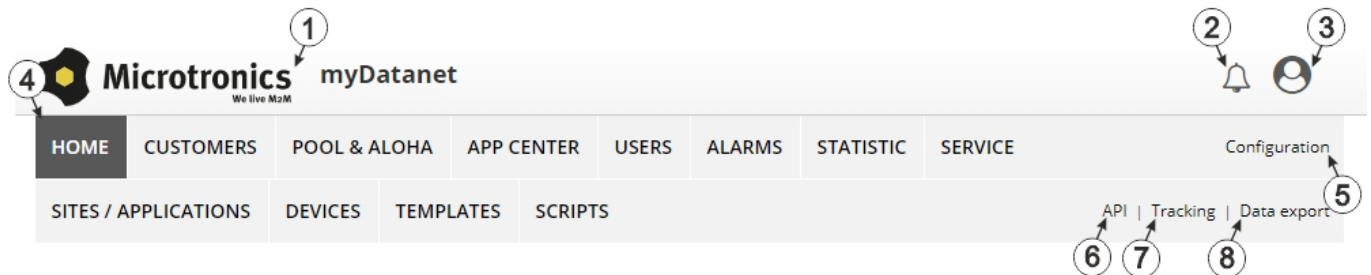
<p><b>1</b> Currently installed software version</p>	<p><b>3</b> The firmware is loaded directly from the server and installed on the device.</p>
<p><b>2</b> Button to install a previously downloaded firmware package</p>	



# Chapter 11 myDatagnet server

**Note:** All of the screenshots show version 50v007 of the myDatagnet server using the standard colour scheme. Newer versions may include minor changes to the appearance of the server.

## 11.1 Overview



Overview of the myDatagnet server

<b>1</b> Freely selectable logo	<b>5</b> Opens the screen to input the global settings for the server
<b>2</b> Opens the window in which the notifications created by the system and intended for the currently logged-in user are summarized	<b>6</b> Opens the rapidM2M Playground
<b>3</b> Displays the menu for adjusting the user settings and for logging out the currently active user	<b>7</b> Switches to the "Data exports" area to configure the data export. This tab is only visible if at least the licence for one export variant is available.
<b>4</b> Tabs to switch between the individual server areas	<b>8</b> Opens the input screen to upload a XML file. This tab is only visible if the licence for the XML import is available.

### 11.1.1 Explanation of the symbols



Adds a new entry to the current list (reports, sites, users, etc.).



Deletes the adjacent element (report, site, user, etc.) from the list.



Calls up the input screen to edit the adjacent element (report, site, user, etc.).

## 11.2 "Customer" area

HOME HEALTH **CUSTOMERS** POOL & ALOHA APP CENTER USERS ALARMS STATISTIC SERVICE Configuration

SITES / APPLICATIONS DEVICES TEMPLATES SCRIPTS API | Tracking | Data export

**1 Overview**

**2 + Customers 3**

2015 Training **8**

Pages: **1** (Total 1)

**4** **5** **6** **7** **9** **10** **11**

!! Training Comment

1234 City Street 1

Overview of the "Customer" area

**1** Area where an image file can be displayed as a "Map" and/or the OpenStreetMaps map can be displayed

The sites can be manually placed on the image file used as a "map".

In the OpenStreetMaps map, the sites are only displayed once GPS coordinates have been assigned to the site.

**2** Adds a new customer

<p><b>3</b> List of tags that are assigned to at least one of the customers displayed in the list of customers. If the list of customers was limited by the search field or selection of a tag, this is taken into consideration when creating the list of tags. A cross is added to the end of the list of tags as soon as the list of customers is limited by the selection of a tag. Clicking on this cross will reset the selection of all tags and the restriction is cancelled.</p> <p>By clicking on one of the tags with the left mouse button only those customers who have been assigned the corresponding tag are displayed in the list of customers and the selected tag is highlighted in colour.</p> <p>By clicking on one of the tags with the right mouse button all of the customers who have been assigned the corresponding tag are hidden, the selected tag is highlighted in colour and the title of the tag is crossed out.</p> <p>Clicking the same mouse button again will remove the restriction.</p>
<p><b>4</b> Opens the input screen for configuring the customer</p>
<p><b>5</b> Deletes the customer</p>
<p><b>6</b> Comment that can be entered in the configuration of the customer</p>
<p><b>7</b> If a default report was defined, the default report is accessed by clicking on the name of the customer. Otherwise the "Sites / Applications" area at customer level is opened by clicking on the name of the customer (see ""Sites / Applications" area at customer level" on page 94 or "Reports" on page 95).</p>
<p><b>8</b> Search field to filter the customer list</p>
<p><b>9</b> Customer's address that can be entered via the input screen for configuring the customer</p>
<p><b>10</b> Symbol via which a OpenStreetMaps map, on which the sites are displayed, can be loaded. (see "Map view" on page 95)</p>
<p><b>11</b> Symbol via which an image file can be loaded on to the server as an "Overview map"</p> <p>To remove the "Map" again, open the upload dialogue again and click on "Submit" without selecting an image file beforehand.</p>

## 11.3 "Sites / Applications" area at customer level

SITES / APPLICATIONS | DEVICES & ALOHA | USERS | ALARMS | STATISTIC | SERVICE

SITES / APPLICATIONS TAGS | DEVICES TAGS API | Data export

1 **Overview**

2 **Reports** 5 6

Report 1 Pages: 1 (Total 1)

Report 1

Channel 1 Site 1	Channel 2 Site 1	Int. Temp Site 1
-0,3	-0,3	<b>22,7</b> °C

3 **Sites / Applications** CONNECTION APPLICATION

Filter: off | Order: Name | Page Length: 12

Austria

Site Pages: 1 (Total 2)

	<b>Site 1</b> <small>4-Channel Data Logger: 047394065Dxxxxxx (9.9.2020 - 1.9.2022)</small>	<span style="color: green;">●</span> <small>1.9.2022 09:45:48 SER UTC+02:00</small>	<small>01:49</small>
	<b>Site 2</b> <small>4-Channel Data Logger: 048A880857xxxxxx (9.9.2020 - 1.9.2022)</small>	<span style="color: green;">●</span> <small>1.9.2022 09:42:01 SER UTC+02:00</small>	<small>01:45</small>

Overview of the "Sites / Applications" area at customer level

- 1** Area where an image file can be displayed as a "Map" and/or the OpenStreetMaps map can be displayed  
 The sites can be manually placed on the image file used as a "map".  
 In the OpenStreetMaps map, the sites are only displayed once GPS coordinates have been assigned to the site.

<b>2</b>	List of reports (see "Reports" on page 95)
<b>3</b>	List of sites/applications (see "Site" on page 74)
<b>4</b>	Symbol that represents a site on the "Map"
<b>5</b>	Symbol via which a OpenStreetMaps map, on which the sites are displayed, can be loaded. (see "Map view" on page 95)
<b>6</b>	Symbol via which an image file can be loaded on to the server as a "Map"  To remove the "Map" again, open the upload dialogue again and click on "Submit" without selecting an image file beforehand.

### 11.3.1 Reports

The reports provide a variety of options to display graphs of the data on the web interface of the myDatenet server or to download the data from the myDatenet server. Detailed instructions on creating and handling the reports is provided in myDatenet Server Manual (805002).

### 11.3.2 Map view

The map view provides an overview of the geographic position of the sites. Detailed instructions on operating and configuring map view are provided in myDatenet Server Manual (805002).

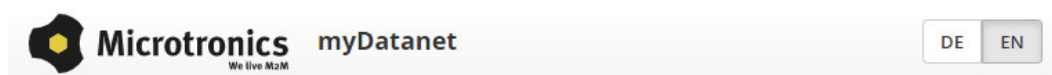
## 11.4 Recommended procedure

### 11.4.1 Creating the site

**Note:** Some of the fields mentioned in the following chapters may be hidden depending on the respective user level. In this case, please contact the administrator of the myDatenet server.

Detailed instructions on creating a new site are provided in myDatenet Server Manual (805002).

1. Log in via the web interface on the myDatenet server. You will receive the web address from your responsible sales partner.



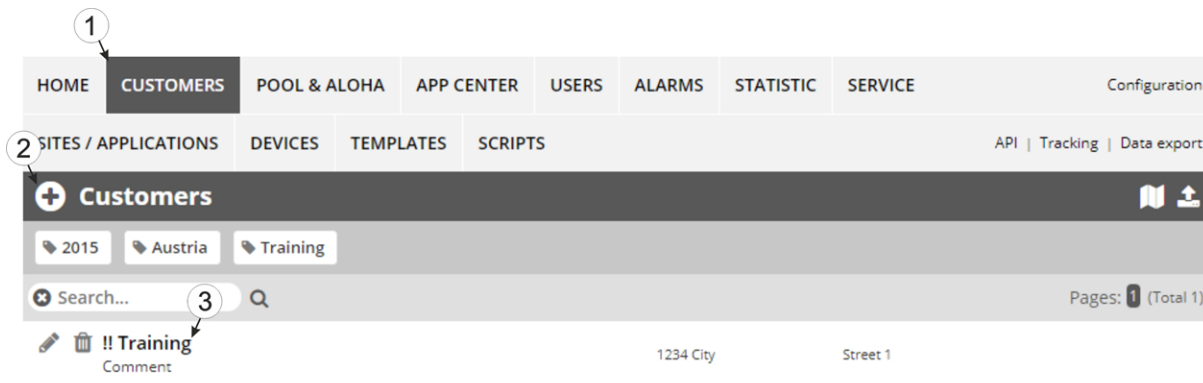
user name

password

LOG IN

Login form of the myDatenet server

- Click on the "Customer" menu item of the myDatanet server to call up the list of available customers. Select an existing customer or create a new customer.

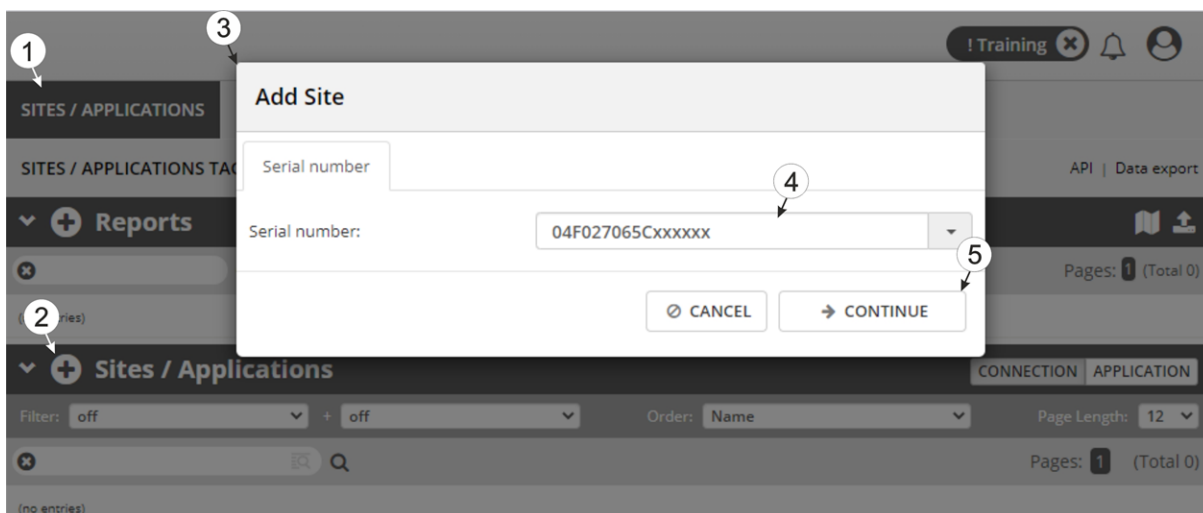


Selecting the customer

1 Menu item to call up the list of customers	3 List of available customers
2 Creating a new customer	

- Click on the "Sites / Applications" menu item of the myDatanet server to call up the list of existing sites / applications. Open the input window for creating a new site by clicking the "Add new site / application" symbol, enter the serial number of your device in the appropriate field and then click the "Continue" button.

**Note:** The serial number is on the type plate of the device (see "Device labelling" on page 25)

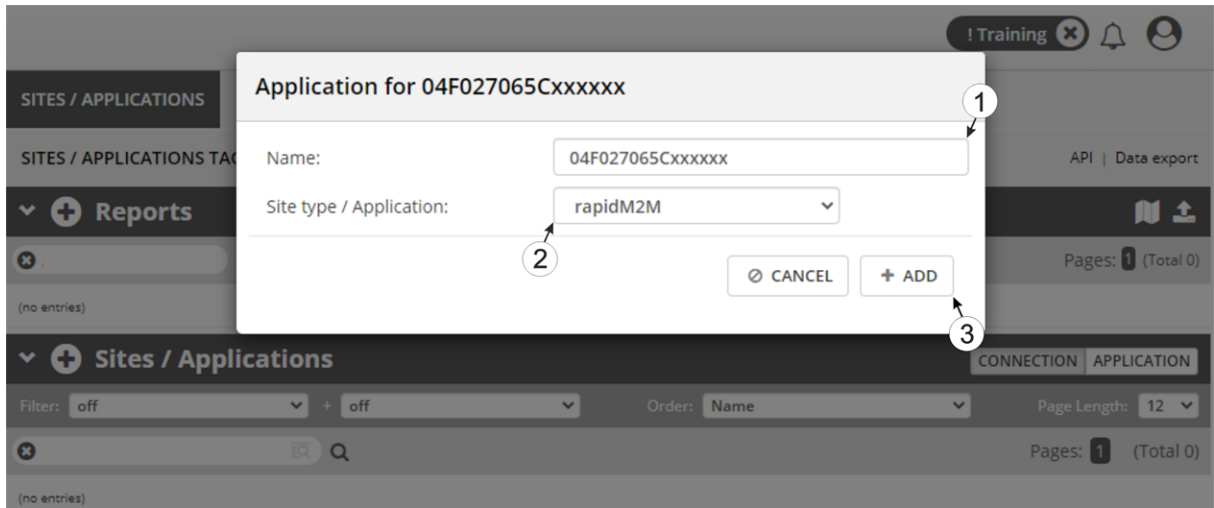


Creating the site

1 Menu item to call up the list of existing sites / applications	4 Field for entering the serial number
2 "Add new site / application" symbol	5 "Continue" button
3 Input window for creating a new site	



4. If necessary, change the suggested name of the site, select the desired site type or the desired application from the drop-down list and then click the "Add" button.



Completing site creation

<b>1</b> Name of the site (freely selectable)	<b>3</b> "Add" button
<b>2</b> Drop-down list of available applications, templates and site types	



---

# Chapter 12 rapidM2M Studio

*Note: The web-based development environment rapidM2M Studio is being developed continuously which can lead to slight changes to the appearance of the program compared to the screenshots used in this manual.*

## 12.1 General

Access to the web-based development environment rapidM2M Studio is included in the Microtronics Partner Program, for which you can register free of charge at the following address:

**<https://partner.microtronics.com>**

It is a web-based IDE that is designed to support customers with the creation of IoT applications for the myDatalogC33x . This covers the entire development process - from editing the source code, to testing as part of the creation process to publishing the finished IoT application in the rapidM2M Store . All elements which make up an IoT application are summarised in a single project. The elements are:

- **Device logic:** intelligence installed locally on the myDatalogC33x
- **Backend logic:** intelligence installed on the myDatagnet server
- **Data descriptor:** describes the structure of the data (measurement data, configurations, etc.), that is exchanged between myDatalogC33x , myDatagnet server and external systems (e.g. front ends connected via REST API).
- **Portal view:** Simple front end that is supplied by the myDatagnet server (e.g. for fast prototype development and/or provision of administrative data)

In addition to the dashboard (see "Project dashboard" on page 101) for managing the projects, the rapidM2M Studio consists of two main interfaces:

- **CODEbed:** Editing and compiling the source codes (see "CODEbed" on page 102)
- **TESTbed:** Testing the IoT application in conjunction with a locally connected device and the associated back end i.e. the myDatagnet server (see "TESTbed" on page 103)

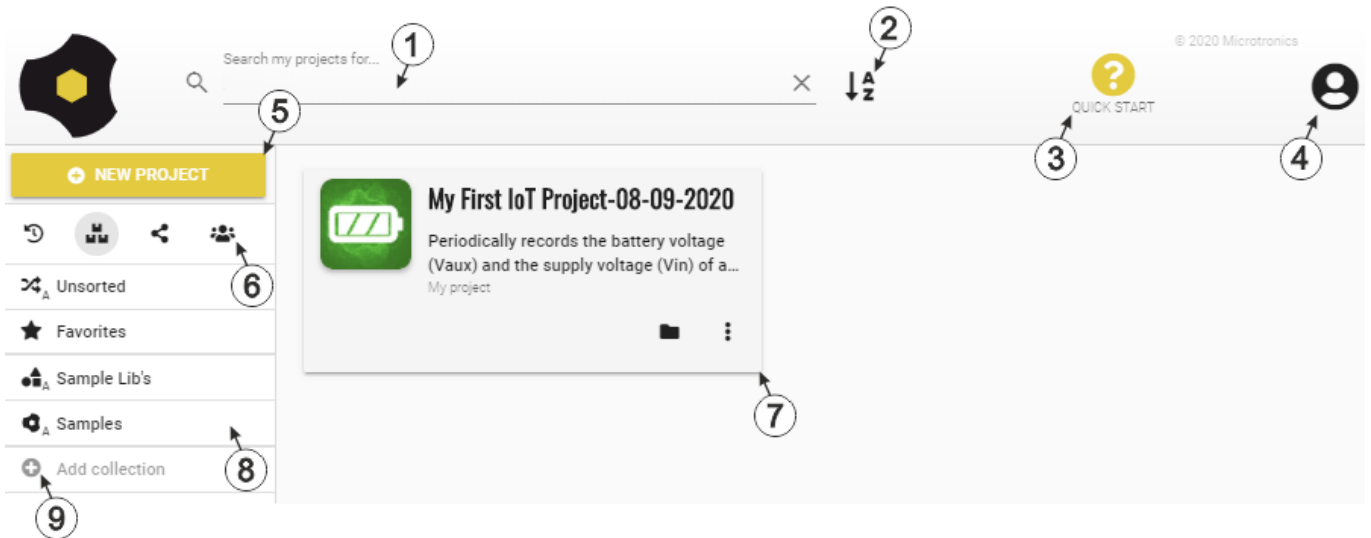
The sharing function implemented in the rapidM2M Studio enables developers from different disciplines (firmware programmers, cloud developers, web designers, etc. ) to create an IoT application together as well as to share projects and libraries with colleagues and the community. The integrated version management also ensures controlled distribution of updates of an IoT application across the entire chain from the rapidM2M Studio to the rapidM2M Store to the sites (that were created based on the IoT application) to the myDatalogC33x .

---

## 12.2 Prerequisites

Interfaces	1 x USB
Operating system	Windows 7 Windows 10 (recommended) MacOS 10.12 or higher Linux (Fedora 32, Ubuntu 20.04, Archlinux 2020.06.01)
Internet connection	Required
Required disk space	No installation required
Browser	Google Chrome only

## 12.3 Project dashboard



Project dashboard of the rapidM2M Studio

1	Search field to filter the list of projects										
2	Button for switching sorting of the project list according to alphabetical order or last use										
3	Opens the quick guide for the rapidM2M Studio										
4	Button for displaying the menu that contains all relevant settings for the currently active user										
5	Button for creating a new project										
6	Buttons for filtering the list of the projects according to:										
	<table border="1"> <tr> <td></td> <td>Recently used</td> </tr> <tr> <td></td> <td>All my projects</td> </tr> <tr> <td></td> <td>Projects shared by me</td> </tr> <tr> <td></td> <td>Projects shared with me</td> </tr> </table>		Recently used		All my projects		Projects shared by me		Projects shared with me		
	Recently used										
	All my projects										
	Projects shared by me										
	Projects shared with me										
7	Tile that contains all important information about an IoT project										
8	List of the "Collections"										
	<table border="1"> <tr> <td></td> <td>All projects that are not assigned to another "Collection"</td> </tr> <tr> <td></td> <td>Favourite projects</td> </tr> <tr> <td></td> <td>Sample libraries provided by Microtronics</td> </tr> <tr> <td></td> <td>Samples provided by Microtronics</td> </tr> <tr> <td></td> <td>"Collection" created by user</td> </tr> </table>		All projects that are not assigned to another "Collection"		Favourite projects		Sample libraries provided by Microtronics		Samples provided by Microtronics		"Collection" created by user
	All projects that are not assigned to another "Collection"										
	Favourite projects										
	Sample libraries provided by Microtronics										
	Samples provided by Microtronics										
	"Collection" created by user										
9	Button for creating a new "Collection"										

## 12.4 CODEbed

MY FIRST IOT PROJECT-08-09-2020  
APP | M22x | 31kB/3MB

```

1 // ---
2 // application entry point
3 // ---
4 main()
5
6

```

RESULTS    
CLIBIMPORT - OK - 3 imports  
DDE - OK  
DLO - OK - total 20712 bytes, code 13916, data 2072, stack/heap 4096, header 628  
HELP - OK - total 2 files

SUMMARY 2 containers, 8% memory occupied  

name	bytes	fields	no
config8	9	3	Re
histdata8	9	2	Ys

CODEbed of the rapidM2M Studio

1	Navigation panel
2	Back to the project dashboard
3	Editor panel
4	Compiler results incl. warnings and errors
5	Memory usage
6	Context-sensitive help
7	Installs the current binaries of the project on the device and backend (i.e. on the myDatnet server) and opens the TESTbed

## 12.5 TESTbed

The screenshot displays the TESTbed interface with the following elements and annotations:

- 1**: Project title and logo.
- 2**: "SELECT DUT & BUT" button.
- 3**: "DUT not ready!" warning message.
- 4**: "WATCHES" panel.
- 5**: "BACKEND" terminal showing configuration.
- 6**: "RESTART" button.
- 7**: "More options" menu icon.
- 8**: Console output showing DUT and BUT states.

TESTbed of the rapidM2M Studio

1	Debug console
2	First opens the window for selecting and connecting the "Device under test" and then the window for entering the access data for the "Backend under test"
3	Information on the "Device under test"
4	Watch panel
5	Information on the "Backend under test" (i.e. the myDatenet server)
6	Restarts the device logic installed on the device. The device logic is reloaded onto the device for this purpose. However, any changes made in the CODEbed are not taken into account here. This is only done again by clicking the button "Install & Run" in the CODEbed.
7	Button for displaying/fading out additional panels
8	Button for deleting the console output





---

# Chapter 13 Device Logic

## 13.1 General

The following chapter describes the functionality of the device logic. The programming language used is built on Pawn, a scripting language similar to C that runs on embedded systems.

Additional, more detailed information is provided on the developer's website:  
<http://www.compuphase.com/pawn/pawn.htm>.

There are several ways to create a device logic for the myDatalogC33x :

- Direct entry in the "Device Logic" input field in the "Control" configuration section
- Upload of a previously created binary file (\*.amx) to the myDatanet server
- Usage of the CODEbed (see "CODEbed" on page 102) of the web-based development environment rapidM2M Studio

### 13.1.1 Direct entry of a device logic

The device logic is entered via the "Control" configuration section (see "Control" on page 75) of the input screen for configuring the site. "Pawn" must be selected as the "Device Logic Type" so that the myDatalogC33x interprets the commands entered under "Device Logic" as a script.

### 13.1.2 Uploading a binary file

If the "Upload a compiled device logic" entry was selected via the "Device logic source" list selection in the "Control" configuration section (see "Control" on page 75) of the input screen for configuring the site, a binary file that was, for example, previously created via the web-based development environment rapidM2M Studio (see "rapidM2M Studio" on page 99) can be uploaded to the myDatanet server. This is then loaded into the myDatalogC33x during the next connection. When using this method, "Pawn" must also be selected as the "Device Logic Type" so that the myDatalogC33x interprets the commands as a script.

### 13.1.3 Using the CODEbed of the web-based development environment rapidM2M Studio

The CODEbed is one of the two main interfaces of the web-based development environment rapidM2M Studio. The CODEbed serves to create and compile source codes for all elements (device logic, backend logic, data descriptor and portal view) of an IoT application. The functional scope of the rapidM2M Studio also includes transfer of the compiled device logic into the myDatalogC33x via a USB connection and copying of the data descriptor to the development site with which the myDatalogC33x is linked.

---

## 13.2 Compiler options

### Compressing the pawn program code

```
// The parameter is used to specify which of the sections should be
// compressed
// 0: no compression (default)
// 1: DATA
// 2: DATA and CODE
// 3: DATA, CODE and TABLES

#pragma amxcompress <0-3>
```

## 13.3 Device API

### 13.3.1 Constants

#### Return codes for general purposes

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

## 13.3.2 Timer, date & time

### 13.3.2.1 Arrays with symbolic indices

#### TrM2M\_DateTime

*Detailed breakdown of the date and time*

```
// year          Year specified relates to the 21st century, i.e. 14 refers to
//              the year 2014
// month        Month   (1..12)
// day          Day     (1..31)
// hour         Hours   (0..23)
// minute       Minutes (0..59)
// second       Seconds (0..59)
// DoW          Weekday (0 = Monday ... 6 = Sunday)
// timestamp    Time stamp (seconds since 31.12.1999)
// timestamp256 Fraction of the next started sec. (resolutions 1/256 sec.)

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

### 13.3.2.2 Constants

#### Time basis flags

*Control flags for the rM2M\_SetDateTime() function*

```
RM2M_DATETIME_LOCALTIME = 0b00000001, // transferred time in local time
```

### 13.3.2.3 Functions

#### native rM2M\_GetTime(&hour=0, &minute=0, &second=0, timestamp=0);

*If no time stamp was transferred (timestamp=0), the current system time (in UTC) is converted to hours/minutes/seconds. Alternatively, the transferred time stamp is converted to hours/minutes/seconds.*

<b>Parameter</b>	<b>Explanation</b>
<i>hour</i>	<i>Variable to store the hours - OPTIONAL</i>
<i>minute</i>	<i>Variable to store the minutes - OPTIONAL</i>
<i>second</i>	<i>Variable to store the seconds - OPTIONAL</i>
<i>timestamp</i>	<i>Time stamp that should be converted</i> = 0: <i>The current system time (in UTC) is converted.</i> > 0: <i>The transferred time stamp is converted.</i> (The time stamp must be specified in seconds since 31.12.1999.)

	<b>Explanation</b>
<i>Return value</i>	<ul style="list-style-type: none"> <li><i>timestamp = 0: Seconds since 31.12.1999 (current system time in UTC)</i></li> <li><i>timestamp &gt; 0: The transferred time stamp is returned.</i></li> </ul>

**native rM2M\_GetDate(&year=0, &month=0, &day=0, timestamp=0);**

If no time stamp was transferred (timestamp=0), the date (year, month, day) is determined for the current system time (in UTC). Alternatively, the date (year, month, day) is determined for the transferred time stamp.

<b>Parameter</b>	<b>Explanation</b>
year	Variable to store the year - OPTIONAL  <b>Note:</b> The year specified relates to the 21st century, i.e. the value 14 refers to the year 2014.
month	Variable to store the month - OPTIONAL
day	Variable to store the day - OPTIONAL
timestamp	Time stamp for which the date should be determined  = 0: The date for the current system time (in UTC) is determined. > 0: The date for the transferred time stamp is determined. (The time stamp must be specified in seconds since 31.12.1999.)

	<b>Explanation</b>
Return value	<ul style="list-style-type: none"> <li>• timestamp = 0: Seconds since 31.12.1999 (current system time in UTC)</li> <li>• timestamp &gt; 0: The transferred time stamp is returned.</li> </ul>

**native rM2M\_GetDateTime(datetime[TrM2M\_DateTime]);**

Reads the current time (in UTC) and date from the system

<b>Parameter</b>	<b>Explanation</b>
datetime	Structure for storing a detailed breakdown of the date and time (see "TrM2M_DateTime" in chapter "Arrays with symbolic indices" on page 107)

	<b>Explanation</b>
Return value	<ul style="list-style-type: none"> <li>• OK, if successful</li> <li>• ERROR, if an invalid parameter was transferred</li> </ul>

**native rM2M\_SetDateTime(datetime[TrM2M\_DateTime], flags=0);**

Sets the system date and time to the values contained in the transferred structure

<b>Parameter</b>	<b>Explanation</b>
<i>datetime</i>	Structure that contains a detailed breakdown of the date and time (see "TrM2M_DateTime" in chapter "Arrays with symbolic indices" on page 107).  <i>.timestamp = 0:</i> The values contained in <i>.year</i> , <i>.month</i> , <i>.day</i> , <i>.hour</i> , <i>.minute</i> and <i>.second</i> are used to set the date/time.  <i>.timestamp != 0:</i> The time stamp contained in <i>.timestamp</i> is used to set the date/time.
<i>flags</i>	Configuration flags for setting the system time - OPTIONAL  <i>Bit0 (RM2M_DATETIME_LOCALTIME):</i> must be set if the transferred structure contains the time in local time

	<b>Explanation</b>
<i>Return value</i>	<ul style="list-style-type: none"> <li>• &gt; 0, difference in seconds between the current time and time to be set</li> <li>• 0, if the difference between the current time and time to be set is less than 5 sec.</li> <li>• ERROR, if invalid parameters were transferred</li> <li>• ERROR-1, if the time to be set is more than one day ahead of the current system time</li> </ul>

**native rM2M\_GetTimezoneOffset();**

Returns the difference (in seconds) between the system time (UTC) and local time configured for the site on the myDatanet server. This can be used to determine the local time in the script by adding the difference to the system time (UTC). The offset value is determined by the myDatanet server in accordance with the set time zone (including summer/winter time) and is synchronised during every connection to the device.

Example: Central European time (CET = UTC+1) is used for the site -> Offset = 3600 sec.

	<b>Explanation</b>
<i>Return value</i>	Offset value in seconds

**native rM2M\_DoW(timestamp);**

Calculates the weekday from a given timestamp

<b>Parameter</b>	<b>Explanation</b>
<i>timestamp</i>	Timestamp of the day in question

	<b>Explanation</b>
<i>Return value</i>	Weekday, 0=Monday ... 6=Sunday

---

**native rM2M\_TimerAdd(funcidx);**

*Generates a new 1s timer*

<b>Parameter</b>	<b>Explanation</b>
<i>funcidx</i>	<i>Index of the public function that should be called up following expiry of the timer</i>  <i>Type of function: public func();</i>

	<b>Explanation</b>
<i>Return value</i>	<ul style="list-style-type: none"><li>• <i>OK, if successful</i></li><li>• <i>ERROR, if one of the following errors occurs:</i><ul style="list-style-type: none"><li>• <i>No valid index was transferred</i></li><li>• <i>No further timers can be created (maximum number reached)</i></li><li>• <i>In the event of an internal error</i></li></ul></li><li>• <i>&lt; OK, if another error occurs (see "Return codes for general purposes" in chapter "Constants" on page 106).</i></li></ul>

**native rM2M\_TimerRemove(funcidx);**

*Removes a 1s timer*

<b>Parameter</b>	<b>Explanation</b>
<i>funcidx</i>	<i>Index of the public function of the timer that should be removed</i>  <i>Type of function: public func();</i>

	<b>Explanation</b>
<i>Return value</i>	<ul style="list-style-type: none"><li>• <i>OK, if successful</i></li><li>• <i>ERROR, if no valid index was transferred or in the event of an internal error</i></li><li>• <i>&lt; OK, if another error occurs (see "Return codes for general purposes" in chapter "Constants" on page 106).</i></li></ul>

**native rM2M\_TimerAddExt(funcidx, bool:cyclic, time);***Generates a new ms timer***Important note:** *The maximum number of simultaneously active ms timers is 8.*

<b>Parameter</b>	<b>Explanation</b>
<i>funcidx</i>	<i>Index of the public function that should be called up following expiry of the timer</i> <i>Type of function: public func();</i>
<i>cyclic</i>	<i>Setting for the behaviour following expiry of the timer interval:</i> <i>true: The timer must be restarted following expiry of the interval.</i> <i>false: The timer is stopped following expiry of the interval.</i>
<i>time</i>	<i>Timer interval in milliseconds (max. 60,000 ms)</i>  <b>Note:</b> <i>By setting the interval to 0ms, a timer can be generated for which the call back function is called up immediately after the current code block (e.g. main function) is executed. However, only single shot timers (i.e. the timer is stopped upon expiry of an interval) may be initialised with an interval of 0ms.</i>

	<b>Explanation</b>
<i>Return value</i>	<ul style="list-style-type: none"> <li>• <i>OK, if successful</i></li> <li>• <i>ERROR, if one of the following errors occurs</i> <ul style="list-style-type: none"> <li>• <i>No valid index was transferred.</i></li> <li>• <i>An interval of 0ms was specified and the timer should be restarted automatically upon expiry of the timeout (i.e. cyclical 0ms timer).</i></li> <li>• <i>Internal error</i></li> <li>• <i>No additional timers can be created (maximum number reached).</i></li> </ul> </li> <li>• <i>&lt; OK, if another error occurs (see "Return codes for general purposes" in chapter "Constants" on page 106).</i></li> </ul>

**native rM2M\_TimerRemoveExt(funcidx);***Removes a ms timer*

<b>Parameter</b>	<b>Explanation</b>
<i>funcidx</i>	<i>Index of the public function of the timer that should be removed</i> <i>Type of function: public func();</i>

	<b>Explanation</b>
<i>Return value</i>	<ul style="list-style-type: none"> <li>• <i>OK, if successful</i></li> <li>• <i>ERROR, if no valid index was transferred or in the event of an internal error</i></li> <li>• <i>&lt; OK, if another error occurs (see "Return codes for general purposes" in chapter "Constants" on page 106).</i></li> </ul>

---

## 13.3.3 Uplink

### 13.3.3.1 Arrays with symbolic indices

#### TrM2M\_GSMInfo

*Information regarding the GSM modem, SIM chip and the GSM network used during the last connection*

```
// cgmi      Manufacturer identification of the modem
// cgmm      Modem model information
// cgmr      Modem revision information
// imei      International mobile equipment identity of the modem
// imsi      International mobile subscriber identity of the SIM chip that was
//           used for the last connection
//           Empty string, if no connection has been established yet
// iccid     Integrated circuit card identifier of the SIM chip that was used
//           for the last connection
//           Empty string, if no connection has been established yet
// mcc      MCC (Mobile Country Code) of the network used for the last/current
//           connection
//           0, if no connection has been established yet
// mnc      MNC (Mobile Network Code) of the network used for the last/current
//           connection
//           0, if no connection has been established yet
// simstate  Current SIM state (see "SIM state" in chapter
//           "Constants" on page 113)
// act      Radio access technology used for the last/current connection (see
//           "Mobile radio Act" in chapter "Constants" on page 113)
// lac      LAC (location area code) of the network used for the last/current
//           connection
// cid      Cell identifier of the network used for the last/current
//           connection
//           2G Act:16-bit cell ID
//           3G Act:28-bit UTRAN cells ID (16-bit cells ID + 12-bit RNC-ID)
//           4G Act:28-bit E-UTRAN cells ID (8-bit sector ID + 20-bit
//           eNodeB-ID)

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

#### TrM2M\_TxItfStats

*Statistical information on the uplink communication interface*

```
// rtt      Time [ms] it takes for the device to receive an answer from the
//           server for a keep alive ping sent to the server (round trip
//           time)1)

#define TrM2M_TxItfStats [.rtt]
```

<sup>1)</sup> can only be determined if the "Bidirectional alive ping" is activated on the server. The "Bidirectional alive ping" enables the device and server to easily detect whether the connection is still established. The "Bidirectional alive ping" can be activated globally for the complete server, for a specific customer or for a single site (see "myDatanet Server Manual " 805002).



### 13.3.3.2 Constants

#### SIM state

```

//Connection can be initiated via devic logic
RM2M_SIM_STATE_NONE          = 0,    //Initial state
RM2M_SIM_STATE_PRODUCTION    = 1,    //Newly produced device is in stock
RM2M_SIM_STATE_HOT           = 2,    //Valid contract

//Connection cannot be initiated via device logic
RM2M_SIM_STATE_COLD          = 3,    /*End of the contract or fair use policy
                                       violated*/
RM2M_SIM_STATE_DISCARDED     = 4,    //Device has been decommissioned

```

#### Mobile radio AcT (access technology)

```

// Mobile radio AcT (access technology) as per 3GPP TS27.007
RM2M_TX_ACT_GSM              = 0,    // GSM
RM2M_TX_ACT_GSM_COMPACT,     = 1,    // GSM compact
RM2M_TX_ACT_UTRAN,           = 2,    // UTRAN
RM2M_TX_ACT_GSM_W_EGPRS,     = 3,    // GSM with EGPRS
RM2M_TX_ACT_UTRAN_W_HSDPA,   = 4,    // UTRAN with HSDPA
RM2M_TX_ACT_UTRAN_W_HSUPA,   = 5,    // UTRAN with HSUPA
RM2M_TX_ACT_UTRAN_W_HSDPA_HSUPA = 6, // UTRAN with HSDPA and HSUPA
RM2M_TX_ACT_E_UTRAN,         = 7,    // E-UTRAN

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

RM2M_TX_ACT_UNKNOWN          = 255 // Unknown

```

#### Connection flags

##### Control flags for the `rM2M_TxStart()` function

```

RM2M_TX_POSUPDATE           = 0b00000001, /* Update of the GSM position data
                                             when establishing a connection */
RM2M_TX_REFRESH_CONFIG      = 0b00000100, /* Additionally establishing a
                                             connection to the maintenance
                                             server */
RM2M_TX_SUPPRESS_POSUPDATE = 0b00001000, /* Suppress update of the GSM position
                                             data when establishing a
                                             connection 1) */
RM2M_TX_POSUPDATE_ONLY      = 0b00010000, /* When establishing a connection, only
                                             the GSM position data is updated.
                                             Measurement data, configurations,
                                             etc. are not synchronised. */

```

<sup>1)</sup> This suppresses the update of the GSM position data that is automatically executed by the firmware every 24h .

---

## Communication modes

*Communication modes for the rM2M\_TxSetMode() function*

```
RM2M_TXMODE_TRIG      = 0,           // Interval
RM2M_TXMODE_WAKEUP   = 1,           // Interval & wakeup
RM2M_TXMODE_ONLINE   = 2,           // Online
```

## Communication mode flags

*Configuration flags for the rM2M\_TxSetMode() function*

```
RM2M_TXMODE_SUPPRESS_SYNC = 0b00000001, /* no auto. sync. with the server when
the connection type is changed */
```

## Connection status

*Return values of the rM2M\_TxGetStatus() function*

```
RM2M_TX_FAILED       = 0b0000000001, // Connection establishment failed
RM2M_TX_ACTIVE       = 0b0000000010, // GPRS connection established
RM2M_TX_STARTED      = 0b0000000100, // Connection establishment started
RM2M_TX_RETRY        = 0b0000001000, // Delay until retry
RM2M_TX_WAKEUPABLE   = 0b0000010000, // Modem is logged into the GSM network

RM2M_TX_DISABLED     = 0b0001000000, // Modem was deactivated
RM2M_TX_WAKEUP       = 0b0100000000, /* Connection establishment triggered
by wakeup SMS */

RM2M_TX_POSUPDATE_ACTIVE = 0b1000000000, // Positioning is running
```

## Connection error codes

*Error codes that are returned by the rM2M\_TxGetStatus() function via the optional "errorcode" parameter if the last connection attempt failed.*

```
RM2M_TXERR_NONE = 0,           // no error

// general errors
RM2M_TXERR_CONNECTION_TIMEOUT, // connection timed out
RM2M_TXERR_NEWDATA_TIMEOUT,    // timeout during server sync in online mode
RM2M_TXERR_IRREGULAR_OFF,      // irregularly closed connection
RM2M_TXERR_SERVER_NOT_AVAILABLE, // server not available
RM2M_TXERR_SERVER_COMMUNICATION, // error during communication with server

// general modem errors
RM2M_TXERR_MODEM = 10,         // unspecified modem error
RM2M_TXERR_MODEM_TIMEOUT,      // timeout modem communication
RM2M_TXERR_MODEM_HW_NOT_FOUND, // modem not found
RM2M_TXERR_MODEM_HW_UNKNOWN,   // unknown modem
RM2M_TXERR_MODEM_INIT,         // error during init
RM2M_TXERR_MODEM_UNRESTART,    /* unsolicited restart (e.g. due to weak power
supply) */
RM2M_TXERR_MODEM_RESETLOOP,    // modem reset-loop detected
RM2M_TXERR_MODEM_UNDERVOLTAGE, // modem undervoltage (power failure) detected
RM2M_TXERR_MODEM_OVERHEAT,     // modem overheat detected
```

```

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

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

// TCP related modem errors
RM2M_TXERR_MODEM_TCP = 70,          /* TCP error (e.g. timeout, server not
                                     available) */

// general WIFI errors
RM2M_TXERR_WIFI = 200,              // unspecified WIFI error
RM2M_TXERR_WIFI_TIMEOUT,            // timeout WIFI communication
RM2M_TXERR_WIFI_HW_NOT_FOUND,       // WIFI device not found
RM2M_TXERR_WIFI_INIT,               // error during init
RM2M_TXERR_WIFI_IO,                 // error IO communication

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

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

// general Ethernet errors
RM2M_TXERR_ETH = 300,               // unspecified Ethernet error
RM2M_TXERR_ETH_TIMEOUT,              // timeout Ethernet communication
RM2M_TXERR_ETH_INIT,                 // error during init
RM2M_TXERR_ETH_IO,                   // error IO communication

```

---

```

RM2M_TXERR_ETH_INIT_MAC_PHY,      // error initialising MAC/PHY interface
RM2M_TXERR_ETH_ITF_UP,           /* TCP/IP stack: error bringing itf up
                                (including dhcp) */

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

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

```

### Available uplink interfaces

Selectable uplink interfaces for the `rM2M_TxSelectItf()` function

```

RM2M_TXITF_NONE      = 0,        /* no uplink, communication with the server
                                not possible */
RM2M_TXITF_MODEM     = 1,        // Mobile network modem
RM2M_TXITF_WIFI      = 2,        // WiFi module
RM2M_TXITF_LAN       = 3,        // LAN interface

```

### Signal strength measurement flags

Control flags for the `rM2M_GSMGetRSSI()` and `rM2M_GetRSSI()` functions.

```

RM2M_RSSI_EXTENDED_VALUE = 0b00000001, /* activates the extended value range
                                (-32768 .. 32767) for the return
                                values of the signal strength */

```

### Configuration flags for the `rM2M_CfgInit()` function

```

RM2M_CFG_VOLATILE = 0b00000001, // volatile storage (RAM)

```

### 13.3.3.3 Callback functions

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

Function to be provided by the device logic developer, that is called up, once a data record has been read (using the function "rM2M\_ReadData()") from the internal flash memory.

**Important note:** The parameter "timestamp256" has only been added in later firmware versions. The number of arguments transferred from the firmware to the callback function should thus be checked via the function "numargs()".

**Example:**

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

<b>Parameter</b>	<b>Explanation</b>
<i>data</i>	Array that contains the data of the read data record
<i>len</i>	Length of the data area of the read data record in bytes (max. 1024 Byte )
<i>timestamp</i>	Time stamp of the data record (in UTC)
<i>timestamp256</i>	Fraction of the next started sec. (Resolution 1/256 sec.)

#### **public func(cfg);**

Function to be provided by the script developer, that is called up if one of the configuration memory blocks has changed.

<b>Parameter</b>	<b>Explanation</b>
<i>cfg</i>	Number of the changed configuration memory block starting with 0 for the first memory block

### 13.3.3.4 Functions

#### native rM2M\_TxStart(flags=0);

triggers a connection to the server with subsequent synchronisation of all memory areas (measurement data, configuration, position data, device log, files,...) between the device and the server. Only those memory areas are transmitted whose content has been changed. If the device is in "online" mode and an active connection to the server is established then this function only triggers synchronisation. The established connection is not disconnected beforehand and then re-established.


**Important note:** In "online" mode new measurement data that are stored in the internal flash via the "rM2M\_RecData()" function are transferred to the server immediately. Calling the "rM2M\_TxStart()" function is thus not necessary to transfer the measurement data in this case. Calling the function and the related synchronisation of all memory areas after generating every single measurement data record would lead to a much higher volume of data. The same also applies to transfer of the configurations. However it is recommended to call the "rM2M\_TxStart()" function occasionally (e.g. every 2h) even in "online" mode since not all memory areas are automatically synchronised.

Parameter	Explanation
flags	<p>Configuration flags for the connection establishment</p> <p>Bit0 (RM2M_TX_POSUPDATE): If set, the GSM position data is also updated.</p> <p>Bit2 (RM2M_TX_REFRESH_CONFIG): If set, a connection to the maintenance server is also established</p> <p>Bit3 (RM2M_TX_SUPPRESS_POSUPDATE): If set, this suppresses the update of the GSM position data that is automatically executed by the firmware every 24h</p> <p>Bit4 (RM2M_TX_POSUPDATE_ONLY): If set, only the GSM position data is updated when a connection is established. Measurement data, configurations etc. are not synchronised.</p>

	Explanation
Return value	<ul style="list-style-type: none"> <li>• OK, if successful</li> <li>• ERROR_SIM_STATE, if a connection is not possible due to the current SIM state (see "SIM state" in chapter "Constants" on page 113)</li> <li>• ERROR_MODEM_DISABLED, if the connection cannot be established due to the supply voltage being too low</li> <li>• ERROR_TXITF, if the connection cannot be established due to the TX interface configuration (e.g. TX interface not open)</li> <li>• &lt; OK, if another error occurs (see "Return codes for general purposes" in chapter "Constants" on page 106).</li> </ul>




**native rM2M\_TxSetMode(mode, flags=0);**

Sets the connection type to be used. If the connection type is changed to "Online" or "Interval & wakeup", this is immediately followed by a connection being established incl. a synchronisation with the server, as long as this is not suppressed by the "RM2M\_TXMODE\_SUPPRESS\_SYNC" flags being set. The same also applies to changing the connection type from "Interval" to "Interval & wakeup".

<b>Parameter</b>	<b>Explanation</b>
<i>mode</i>	<p>Connection type to be used:</p> <p><i>RM2M_TXMODE_TRIG</i>: The connection is established when the "rM2M_TxStart()" function is called</p> <p><i>RM2M_TXMODE_WAKEUP</i>: The connection is established in the same way as in "Interval" mode when the "rM2M_TxStart()" function is called. Additionally, the device can be initiated via the server to immediately establish a connection (see "myDatanet Server Manual " 805002). For this purpose, the device immediately logs into the GSM network as soon as this mode has been set.</p>  <p><i>RM2M_TXMODE_ONLINE</i>: The device does not disconnect the connection and continuously transmits the measurement data. However, every 7 days, the connection is temporarily interrupted in order to verify the server assignment. The connection is established as soon as this mode has been set. Calling the "rM2M_TxStart()" function is not necessary.</p>
<i>flags</i>	<p>Configuration flags for the communication mode</p> <p><i>Bit0</i>: automatic sync. with the server when the connection type is changed  0 = Execute synchronisation  <i>RM2M_TXMODE_SUPPRESS_SYNC</i> = Suppress synchronisation</p>

	<b>Explanation</b>
<i>Return value</i>	<ul style="list-style-type: none"> <li>• OK, if successful</li> <li>• <i>ERROR_SIM_STATE</i>, if the mode is not possible due to the current SIM state (see "SIM state" in chapter "Constants" on page 113)</li> <li>• <i>ERROR_MODEM_DISABLED</i>, if the connection cannot be established due to the supply voltage being too low</li> <li>• <i>ERROR_TXITF</i>, if the connection cannot be established due to the TX interface configuration (e.g. TX interface not open)</li> <li>• &lt; OK, if another error occurs (see "Return codes for general purposes" in chapter "Constants" on page 106).</li> </ul>

**Note:** Additional explanation about the connection types

Connection type	Energy consumption	Data volumes	Response time
online			
Interval & wakeup			
Interval			

**native rM2M\_TxGetStatus(&errorcode=0);**

Returns the current connection status

Parameter	Explanation
errorcode	<p>Variable to store the error code that occurred during the last connection attempt</p> <p><i>RM2M_TXERR_NONE:</i> Last connection establishment successful</p> <p><i>&gt; RM2M_TXERR_NONE:</i> Last connection establishment failed. Detailed breakdown of the error codes is provided in "Connection error codes" in chapter "Uplink" on page 112.</p>

	Explanation
Return value	<p><i>Bit0 (RM2M_TX_FAILED):</i> set if the last GPRS connection establishment failed</p> <p><i>Bit1 (RM2M_TX_ACTIVE):</i> set when a GPRS connection is established</p> <p><i>Bit2 (RM2M_TX_STARTED):</i> set when a connection establishment has been started</p> <p><i>Bit3 (RM2M_TX_RETRY):</i> set during the delay until the next automatic retry in the event of connection problems</p> <p><i>Bit4 (RM2M_TX_WAKEUPABLE):</i> set when the modem is logged into the GSM network (wakeup possible)</p> <p><i>Bit6 (RM2M_TX_DISABLED):</i> set if the modem has been deactivated</p> <p><i>Bit8 (RM2M_TX_WAKEUP):</i> Set when a connection establishment was triggered upon receipt of a wakeup SMS</p> <p><i>Bit9 (RM2M_TX_POSUPDATE_ACTIVE):</i> set when positioning is running</p>



**native rM2M\_TxSelectItf(itf);**

Selects the communication interface to be used for the uplink

<b>Parameter</b>	<b>Explanation</b>
<i>itf</i>	Selection of the communication interface  <i>RM2M_TXITF_NONE</i> : No uplink, communication with the server not possible  <i>RM2M_TXITF_MODEM</i> : Mobile network modem  <i>RM2M_TXITF_WIFI</i> : WiFi module  <i>RM2M_TXITF_LAN</i> : LAN interface

	<b>Explanation</b>
<i>Return value</i>	<ul style="list-style-type: none"> <li>• OK, if successful</li> <li>• ERROR, if the selected communication interface is not supported by the device or another error occurs</li> </ul>

**native rM2M\_TxItfGetStats(stats[TrM2M\_TxItfStats], len=sizeof stats);**

Returns the statistical information on the uplink communication interface

<b>Parameter</b>	<b>Explanation</b>
<i>stats</i>	Structure for storing the statistical information (see "TrM2M_TxItfStats" in chapter "Arrays with symbolic indices" on page 112)
<i>len</i>	Size (in cells) of the structure to store the statistical information – OPTIONAL

	<b>Explanation</b>
<i>Return value</i>	<ul style="list-style-type: none"> <li>• OK, if successful</li> </ul>

**native rM2M\_SetTCPKeepAlive(time=0);**

Sets the interval at which the keep alive pings are sent during online mode

<b>Parameter</b>	<b>Explanation</b>
<i>time</i>	Sets the time between the keep alive pings  <i>0</i> : Default setting saved in the firmware is used (15 min. 3 sec.) <i>&lt; 241</i> : in 1 sec. increments <i>241 .. 255</i> : in 5 min. increments, whereby 3 sec. is subsequently added  e.g. 243: 3*5 min. + 3 sec. = 15 min. 3 sec.

	<b>Explanation</b>
<i>Return value</i>	<ul style="list-style-type: none"> <li>• OK, if successful</li> </ul>

---

**native rM2M\_GSMGetRSSI(flags=0);**

Returns the GSM/UMTS/LTE signal strength

**Important note:** Although this function will still be supported for the purpose of downward compatibility, it should no longer be used for new projects. The "rM2M\_GetRSSI()" function should be used as an alternative.

<b>Parameter</b>	<b>Explanation</b>
flags	Configuration flags for the signal strength measurement  Bit0 (RM2M_RSSI_EXTENDED_VALUE): If set, the extended value range for the return of the signal strength is used

	<b>Explanation</b>
Return value	Signal strength in [dBm]  RM2M_RSSI_EXTENDED_VALUE not set: <ul style="list-style-type: none"><li>• Maximum value range: -127 to 127</li><li>• Out of range at: -128</li></ul> RM2M_RSSI_EXTENDED_VALUE set: <ul style="list-style-type: none"><li>• Maximum value range: -32767 to 32767</li><li>• Out of range at: -32768</li></ul> GSM values range from -113 to -51 dBm. UMTS values range from -116 to -54 dBm. LTE values range from -141 to -44 dBm.

**native rM2M\_GetRSSI(flags=0);**

Returns the signal strength at the communication interface used for the uplink

	<b>Explanation</b>
Return value	Signal strength in [dBm]  RM2M_RSSI_EXTENDED_VALUE not set: <ul style="list-style-type: none"><li>• Maximum value range: -127 to 127</li><li>• Out of range at: -128</li></ul> RM2M_RSSI_EXTENDED_VALUE set: <ul style="list-style-type: none"><li>• Maximum value range: -32767 to 32767</li><li>• Out of range at: -32768</li></ul> GSM values range from -113 to -51 dBm. UMTS values range from -116 to -54 dBm. LTE values range from -141 to -44 dBm. When using the LAN interface the return value is 0 dBm.

**native rM2M\_GSMGetInfo(info[TrM2M\_GSMInfo], len=sizeof info);**

Returns information on the GSM modem, SIM chip and the GSM network used during the last connection

<b>Parameter</b>	<b>Explanation</b>
<i>info</i>	Structure for storing the information (see "TrM2M_GSMInfo" in chapter "Arrays with symbolic indices" on page 112)
<i>len</i>	Size (in cells) of the structure to store the information - OPTIONAL

	<b>Explanation</b>
<i>Return value</i>	<ul style="list-style-type: none"> <li>• Used size (in cells) of the structure for storing the information</li> <li>• ERROR if the address and/or length of the info structure are invalid (outside the script data memory)</li> <li>• &lt; OK, if another error occurs (see "Return codes for general purposes" in chapter "Constants" on page 106).</li> </ul>

**native rM2M\_LiveData(const data{}, len);**

Transmits a data record as live data to the server. Calling this function is only permissible if the device is in "online" mode and an active connection to the server is established. Use the "rM2M\_Pack", "rM2M\_SetPacked" or "rM2M\_SetPackedB" functions to generate the data area.

<b>Parameter</b>	<b>Explanation</b>
<i>data</i>	Array that contains the live data to be transferred  <p style="text-align: center;"><b>Important note:</b> The structure of the live data must be identical to that of the measurement data saved in the internal flash using the "rM2M_RecData()" function.</p>
<i>len</i>	Number of bytes to be transferred (max. 1024 Byte )

	<b>Explanation</b>
<i>Return value</i>	<ul style="list-style-type: none"> <li>• OK, if successful</li> <li>• ERROR, if an error occurs (e.g. the server does not support receipt of live data.)</li> </ul>

---

**native rM2M\_RecData(timestamp, const data{}, len);**

Saves a data record in the internal flash memory. Use the "rM2M\_Pack", "rM2M\_SetPacked" or "rM2M\_SetPackedB" functions to generate the data area.

<b>Parameter</b>	<b>Explanation</b>
<i>timestamp</i>	<i>Time stamp that should be used for the recording</i>  <i>= 0: The current system time is used as the time stamp.</i> <i>&gt; 0: The transferred time stamp is used.</i> <i>(The time stamp must be specified in seconds since 31.12.1999)</i>
<i>data</i>	<i>Array that contains the data to be saved</i>
<i>len</i>	<i>Number of bytes to be saved (max. 1024 Byte )</i>

	<b>Explanation</b>
<i>Return value</i>	<ul style="list-style-type: none"><li>• <i>OK, if successful</i></li><li>• <i>-2, if data storage is not currently possible as the internal memory is being reorganised. The data must be temporarily saved in the script and stored again at a later date.</i></li><li>• <i>ERROR, if one of the following errors occurs</i><ul style="list-style-type: none"><li>• <i>Memory area (data{}, len) is invalid.</i></li><li>• <i>More than 10 calls during one script run.</i></li><li>• <i>Number of bytes to be saved &gt; 1024 Byte</i></li><li>• <i>FLASH write process not successful</i></li><li>• <i>The transfer parameter "timestamp" is more than 5 minutes in the future</i></li></ul></li><li>• <i>&lt; OK, if another error occurs (see "Return codes for general purposes" in chapter "Constants" on page 106).</i></li></ul>

**native rM2M\_ReadData(recidx, funcidx);**

Reads out a data record saved in the internal flash and then calls up the function for which the index was transferred.

<b>Parameter</b>	<b>Explanation</b>
<i>recidx</i>	<i>Index of the data record to be read (-1 = last/current data record, -2 = penultimate data record, .... )</i>
<i>funcidx</i>	<i>Index of the public function that should be called once the data record has been read from the internal flash memory.</i>  <i>Type of function: public func(const data[], len, timestamp, timestamp256);</i>

	<b>Explanation</b>
<i>Return value</i>	<ul style="list-style-type: none"><li>• <i>OK, if the read process has been started</i></li><li>• <i>ERROR, if an error occurs</i></li></ul>

**native rM2M\_CfgInit(cfg, flags);**

Sets the configuration for a configuration memory block. Calling the function is only necessary if one of the configuration flags should be set.

<b>Parameter</b>	<b>Explanation</b>
<i>cfg</i>	Number of the configuration memory block starting with 0 for the first memory block. The device comprises 10 independent memory blocks.
<i>flags</i>	Configuration flags to be set/deleted  Bit0: Type of storage 0 (default) = stored in FLASH in non-volatile manner RM2M_CFG_VOLATILE = saved in RAM in volatile manner

	<b>Explanation</b>
<i>Return value</i>	<ul style="list-style-type: none"> <li>• OK, if successful</li> <li>• &lt; OK, if an error occurs (see "Return codes for general purposes" in chapter "Constants" on page 106)</li> </ul>

**Note:** Additional explanation on the type of storage:

If Bit 0 was not set (default), the non-volatile storage of the configuration memory block in the FLASH is initiated when the "rM2M\_CfgWrite" function is called up.

If Bit0 was set (Bit0 = RM2M\_CFG\_VOLATILE), the configuration memory block is saved in the RAM in a volatile manner when the "rM2M\_CfgWrite" function is called. This option is recommended if the data in the configuration memory block changes frequently as this will reduce the number of flash write cycles. The "rM2M\_CfgFlush" function must be called up so that the configuration memory block is saved in a non-volatile manner in the FLASH.

---

**native rM2M\_CfgWrite(cfg, pos, const data[], size);**

Saves the transferred data block at the specified position in a configuration memory block. Note that the configuration memory block is either saved in the RAM in a volatile manner (Bit0 = RM2M\_CFG\_VOLATILE) or in the FLASH in a non-volatile manner (Bit0 = 0, default) depending on the type of storage selected via the "rM2M\_CfgInit" function. The function is also passed which of the 10 available memory blocks in the internal flash memory should be used. Use the "rM2M\_Pack", "rM2M\_SetPacked" or "rM2M\_SetPackedB" functions to generate the data block that should be saved. The time stamp is updated, so that the configuration memory block is automatically synchronised with the myDatanet server during the next connection.

<b>Parameter</b>	<b>Explanation</b>
<i>cfg</i>	Number of the configuration memory block starting with 0 for the first memory block. The device comprises 10 independent memory blocks.
<i>pos</i>	Byte offset within the configuration memory block to determine the position where the data should be written.
<i>data</i>	Array that contains the data that should be written in the configuration memory block
<i>size</i>	Number of bytes that should be written in the configuration memory block

	<b>Explanation</b>
<i>Return value</i>	<ul style="list-style-type: none"><li>• &gt; 0: Current size of the configuration memory block if successful</li><li>• ERROR_MEM, if enough temporary memory (RAM) is not currently available. (can occur if "RAM in a volatile manner" is selected as the type of storage for several configuration memory blocks)</li><li>• &lt; OK, if another error occurs (see "Return codes for general purposes" in chapter "Constants" on page 106).</li></ul>

**native rM2M\_CfgFlush(cfg);**

Saves the configuration memory block for which the number was transferred in the FLASH in a non-volatile manner. Calling the function is only necessary if "volatile in RAM (Bit0 = RM2M\_CFG\_VOLATILE)" was selected as the type of storage for the relevant configuration memory block via the "rM2M\_CfgInit" function.

<b>Parameter</b>	<b>Explanation</b>
<i>cfg</i>	Number of the configuration memory block starting with 0 for the first memory block. The device comprises 10 independent memory blocks.

	<b>Explanation</b>
<i>Return value</i>	<ul style="list-style-type: none"><li>• OK, if successful</li><li>• &lt; OK, if an error occurs (see "Return codes for general purposes" in chapter "Constants" on page 106)</li></ul>

**native rM2M\_CfgRead(cfg, pos, data{}, size);**

Reads a data block from the specified position in a configuration memory block. The function is also informed which of the 10 available memory blocks in the internal flash memory should be read. Use the "rM2M\_Pack", "rM2M\_GetPacked" or "rM2M\_GetPackedB" functions to unpack the read data.

<b>Parameter</b>	<b>Explanation</b>
<i>cfg</i>	Number of the configuration memory block starting with 0 for the first memory block. The device comprises 10 independent memory blocks.
<i>pos</i>	Byte offset within the configuration memory block to determine the position from which the data should be read
<i>data</i>	Array to store the data to be read
<i>size</i>	Number of bytes that should be read from the configuration memory block

	<b>Explanation</b>
<i>Return value</i>	<ul style="list-style-type: none"> <li>• &gt;0: Number of bytes actually read. This may be less or equal to the requested number of bytes.</li> <li>• <i>ERROR_MEM</i>, if enough temporary memory (RAM) is not currently available. (can occur if "RAM in a volatile manner" is selected as the type of storage for several configuration memory blocks)</li> <li>• &lt; OK, if another error occurs (see "Return codes for general purposes" in chapter "Constants" on page 106).</li> </ul>

**native rM2M\_CfgDelete(cfg);**

Deletes all of the data of the transferred configuration memory block

<b>Parameter</b>	<b>Explanation</b>
<i>cfg</i>	Number of the configuration memory block starting with 0 for the first memory block. The device comprises 10 independent memory blocks.

	<b>Explanation</b>
<i>Return value</i>	<ul style="list-style-type: none"> <li>• OK, if successful</li> <li>• <i>ERROR_MEM</i>, if enough temporary memory (RAM) is not currently available. (can occur if "RAM in a volatile manner" is selected as the type of storage for several configuration memory blocks)</li> <li>• &lt; OK, if another error occurs (see "Return codes for general purposes" in chapter "Constants" on page 106).</li> </ul>

---

## native rM2M\_CfgOnChg(funcidx);

*Specifies the function that should be called if one of the configuration memory blocks has changed*

<b>Parameter</b>	<b>Explanation</b>
<i>funcidx</i>	<i>Index of the public function that should be called up if the configuration has changed</i>  <i>Type of function: public func(cfg);</i>

	<b>Explanation</b>
<i>Return value</i>	<ul style="list-style-type: none"><li>• <i>OK, if successful</i></li><li>• <i>ERROR, if no valid index of a public function was transferred</i></li><li>• <i>&lt; OK, if another error occurs (see "Return codes for general purposes" in chapter "Constants" on page 106).</i></li></ul>

## 13.3.4 Encoding

### 13.3.4.1 Constants

#### Configuration flags for the rM2M\_Pack() function

```
RM2M_PACK_GET           = 0b00000001, // Value should be read (get packed)  
RM2M_PACK_BE           = 0b00000010, // Use "Big endian" format  
RM2M_PACK_U8           = 0b00010000, // 8-bit unsigned  
RM2M_PACK_S8           = 0b10010000, // 8-bit signed  
RM2M_PACK_U16          = 0b00100000, // 16-bit unsigned  
RM2M_PACK_S16          = 0b10100000, // 16-bit signed  
RM2M_PACK_U32          = 0b01000000, // 32-bit unsigned  
RM2M_PACK_S32          = 0b11000000, // 32-bit signed  
RM2M_PACK_F32          = 0b01000000, // 32-bit float
```



### 13.3.4.2 Functions

**native rM2M\_SetPacked(data{}, pos, &{Float,Fixed,\_}:value, size=4, bool:bigendian=false);**

*Writes the transferred value to a specified position in an array*

**Important note:** Although this function will still be supported for the purpose of downward compatibility, it should no longer be used for new projects as the signed data types might lead to problems. The „rM2M\_Pack()“ function should be used as an alternative.

<b>Parameter</b>	<b>Explanation</b>
<i>data</i>	<i>Array that should be used as a data area for a data record or a configuration</i>
<i>pos</i>	<i>Byte offset within the array to determine the position where the value should be written</i>
<i>value</i>	<i>Value that should be written in the array</i>
<i>size</i>	<i>Number of bytes that should be used for the value to be written</i>
<i>bigendian</i>	<i>Settings for the byte sequence that should be used when writing the value: true: "Big endian" is used false: "Little endian" is used</i>

	<b>Explanation</b>
<i>Return value</i>	<ul style="list-style-type: none"> <li>• OK, if successful</li> <li>• &lt; OK, if an error occurs (see "Return codes for general purposes" in chapter "Constants" on page 106)</li> </ul>

**Note:** Additional explanation on the byte sequence:

*In the following example, the whole number 439.041.101 is saved as a 32-bit integer value from memory address 10000.*

<b>Addresses</b>	<b>Big endian</b>			<b>Little endian</b>		
	<b>Hex</b>	<b>Dez</b>	<b>Binary</b>	<b>Hex</b>	<b>Dez</b>	<b>Binary</b>
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);**

*Writes the transferred data block to the specified position in an array*

<b>Parameter</b>	<b>Explanation</b>
<i>data</i>	<i>Array that should be used as a data area for a data record or a configuration</i>
<i>pos</i>	<i>Byte offset within the array to determine the position where the data block should be written</i>
<i>block</i>	<i>Data block that should be written in the array</i>
<i>size</i>	<i>Number of bytes to be written from the data block to the array</i>

	<b>Explanation</b>
<i>Return value</i>	<ul style="list-style-type: none"><li>• <i>OK, if successful</i></li><li>• <i>&lt; OK, if an error occurs (see "Return codes for general purposes" in chapter "Constants" on page 106)</i></li></ul>

**native rM2M\_GetPacked(const data{}, pos, &{Float,Fixed,}\_:value, size=4, bool:bigendian=false);**  
*Supplies the value that is located at the specified position in an array*

**Important note:** Although this function will still be supported for the purpose of downward compatibility, it should no longer be used for new projects as the signed data types might lead to problems. The „rM2M\_Pack()" function should be used as an alternative.

<b>Parameter</b>	<b>Explanation</b>
<i>data</i>	<i>Array that should be used as a data area for a data record or a configuration</i>
<i>pos</i>	<i>Byte offset within the array to determine the position from which the data should be read</i>
<i>value</i>	<i>Variable to store the data to be read</i>
<i>size</i>	<i>Number of bytes that should be read</i>
<i>bigendian</i>	<i>Specifies how the packed data must be interpreted:  true: The data is saved in "Big endian" format in the array. false: The data is saved in "Little endian" format in the array.</i>

	<b>Explanation</b>
<i>Return value</i>	<ul style="list-style-type: none"> <li>• OK, if successful</li> <li>• &lt; OK, if an error occurs (see "Return codes for general purposes" in chapter "Constants" on page 106)</li> </ul>

**Note:** Additional explanation on the byte sequence:

*In the following example, the whole number 439.041.101 is saved as a 32-bit integer value from memory address 10000.*

<b>Addresses</b>	<b>Big endian</b>			<b>Little endian</b>		
	<b>Hex</b>	<b>Dez</b>	<b>Binary</b>	<b>Hex</b>	<b>Dez</b>	<b>Binary</b>
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);**

*Reads a data block that is located at the specified position in an array*

<b>Parameter</b>	<b>Explanation</b>
<i>data</i>	<i>Array that should be used as a data area for a data record or a configuration</i>
<i>pos</i>	<i>Byte offset within the array to determine the position from which the data should be read</i>
<i>block</i>	<i>Array to store the data to be read</i>
<i>size</i>	<i>Number of bytes that should be read</i>

	<b>Explanation</b>
<i>Return value</i>	<ul style="list-style-type: none"><li>• <i>OK, if successful</i></li><li>• <i>&lt; OK, if an error occurs (see "Return codes for general purposes" in chapter "Constants" on page 106)</i></li></ul>

**native rM2M\_Pack(const data{}, pos, &{Float,Fixed,}\_:value, type);**

Function to access packed data. If the Bit0 (RM2M\_PACK\_GET) of the "type" parameter was set, the function returns the value that is located at the specified position in the array. Otherwise the function writes the transferred value to the specified position in the array.

<b>Parameter</b>	<b>Explanation</b>
<i>data</i>	Array with the packed content  Set packed: Array to which the value should be written Get packed: Array from which the value should be read
<i>pos</i>	Byte offset within the array  Set packed: Position to which the value should be written Get packed: Position from which the value should be read
<i>value</i>	Set packed: Value that should be written in the array Get packed: Variable to store the data to be read
<i>type</i>	Configuration flags for the function  Bit0: Select "Set packed" / "Get packed" 0 = value should be written 1 = value should be read  Bit1: Byte order 0 = "Little endian" format 1 = "Big endian" format  Bit2...3 reserved for extensions  Bit4...7: Data type 1 = 8-bit unsigned 2 = 16-bit unsigned 4 = 32-bit unsigned / 32-bit float 9 = 8-bit signed 10 = 16-bit signed 12 = 32-bit signed  <b>Note:</b> You can also use the predefined constants for this parameter (see "Configuration flags for the rM2M_Pack() function" in chapter "Constants" on page 128). The constants can also be combined using the "or" link.

	<b>Explanation</b>
<i>Return value</i>	<ul style="list-style-type: none"> <li>• OK, if successful</li> <li>• &lt; OK, if an error occurs (see "Return codes for general purposes" in chapter "Constants" on page 106)</li> </ul>

---

## 13.3.5 Registry

### 13.3.5.1 Constants

#### Indices of the registration memory blocks

that can be accessed via the "rM2M\_RegGetString()", "rM2M\_RegGetValue()", "rM2M\_RegSetString()", "rM2M\_RegSetValue()", "rM2M\_RegDelValue()" and "rM2M\_RegDelKey()" functions. Detailed information on the registration memory blocks is provided in chapter "Registration memory blocks" on page 37.

```
//System-specific data
RM2M_REG_SYS_OTP    = 0, /* Written once as part of the production
                           process (readonly by device logic). */
RM2M_REG_SYS_FLASH = 1, /* Can be changed during operation (readonly by
                           device logic) */

//Application-specific data
RM2M_REG_APP_OTP    = 2, /* Recommendation: Write only once as part of
                           the production process (readable and writeable
                           by device logic) */
RM2M_REG_APP_FLASH = 3, /* Can be changed during operation (readable and
                           writeable by device logic) */

//Application-specific, volatile data
RM2M_REG_APP_STATE = 4, /* Can be changed during operation (readable and
                           writeable by device logic). Requires
                           "rM2M_RegInit()" */

//Number of registration memory blocks
RM2M_REG_NUM_REGS  = 5,
```

#### Error codes for the registration memory block access operations

```
RM2M_REG_ERROR_TOKENMEM = -101, // Not enough tokens were provided
RM2M_REG_ERROR_INVALID = -102, // Invalid character inside JSON string
RM2M_REG_ERROR_PART     = -103, /* The string is not a full JSON packet, more
                                   bytes expected */

RM2M_REG_ERROR_NOMEM    = -200, // memory allocation failed
RM2M_REG_ERROR_NUMTOKENS = -201, /* not enough token available for this
                                   object/array size */

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

#### Configuration flags for the rM2M\_RegInit() function

```
RM2M_REG_VOLATILE = 0b00000001, // volatile storage (RAM)
```

### 13.3.5.2 Callback functions

#### public func(reg);

Function to be provided by the device logic developer, that is called up if the registration has changed

<b>Parameter</b>	<b>Explanation</b>
<i>reg</i>	Index of the registration memory block (see "Indices of the registration memory blocks" in chapter "Constants" on page 134) that has been changed

### 13.3.5.3 Functions

#### native rM2M\_RegInit(reg, flags, data{}, len=sizeof data);

initialises one of the optional registration memory blocks stored in the RAM. Calling up the function is only necessary for the registration memory blocks listed in the explanation of the "reg" parameter. Detailed information on the registration memory blocks is provided in chapter "Registration memory blocks" on page 37.

<b>Parameter</b>	<b>Explanation</b>
<i>reg</i>	Registration memory block index  The following registration memory blocks require an initialisation: <ul style="list-style-type: none"> <li>• <i>RM2M_REG_APP_STATE</i>: Application-specific, volatile data (e.g. current device status)</li> </ul>
<i>flags</i>	Configuration flags to be set/deleted  Bit0: Type of storage 0 = invalid, currently not supported <i>RM2M_REG_VOLATILE</i> = saved in RAM in volatile manner
<i>data</i>	Array to store the registration memory block
<i>len</i>	Size (in cells) of the transferred array to store the registration memory block (max. 1kB) - OPTIONAL

	<b>Explanation</b>
<i>Return value</i>	<ul style="list-style-type: none"> <li>• OK, if successful</li> <li>• ERROR, if an unspecified errors occurs</li> <li>• &lt; OK, if another error occurs (see "Error codes for the registration memory block access operations" in chapter "Constants" on page 134)</li> </ul>

**native rM2M\_RegGetString(reg, const name[], string[], len=sizeof string);**

*Reads a character string from a registration memory block. Detailed information on the registration memory blocks is provided in chapter "Registration memory blocks" on page 37.*

<b>Parameter</b>	<b>Explanation</b>
<i>reg</i>	<i>Index of the registration memory block (see "Indices of the registration memory blocks" in chapter "Constants" on page 134)</i>  <b>Note:</b> <i>RM2M_REG_APP_STATE</i> requires "rM2M_RegInit ()" before.
<i>name</i>	<i>Name of the entry</i>
<i>string</i>	<i>Array to store the string to be read</i>  <i>(Extended JSON string format, see "rM2M_RegSetString ()" for details)</i>
<i>len</i>	<i>Size (in cells) of the transferred array to store the string to be read - OPTIONAL</i>

	<b>Explanation</b>
<i>Return value</i>	<ul style="list-style-type: none"> <li>• <i>OK, if successful</i></li> <li>• <i>ERROR, if an unspecified errors occurs</i></li> <li>• <i>RM2M_REG_ERROR_NOTFOUND, if the specified entry does not exist</i></li> <li>• <i>RM2M_REG_ERROR_ISNULL if the value of the specified entry is set to "null"</i></li> <li>• <i>&lt; OK, if another error occurs (see "Error codes for the registration memory block access operations" in chapter "Constants" on page 134)</i></li> </ul>

**native rM2M\_RegGetValue(reg, const name[], &{Float,Fixed,\_}:value, tag=tagof value);**

*reads a value from a registration memory block. Detailed information on the registration memory blocks is provided in chapter "Registration memory blocks" on page 37.*

<b>Parameter</b>	<b>Explanation</b>
<i>reg</i>	<i>Index of the registration memory block (see "Indices of the registration memory blocks" in chapter "Constants" on page 134)</i>  <b>Note:</b> <i>RM2M_REG_APP_STATE</i> requires "rM2M_RegInit ()" before.
<i>name</i>	<i>Name of the entry</i>
<i>value</i>	<i>Variable to store the value to be read</i>
<i>tag</i>	<i>The integer and floating-point conversion are differentiated by the "tag" of the variables. - OPTIONAL</i>

	<b>Explanation</b>
<i>Return value</i>	<ul style="list-style-type: none"> <li>• <i>OK, if successful</i></li> <li>• <i>ERROR, if an unspecified errors occurs</i></li> <li>• <i>RM2M_REG_ERROR_NOTFOUND, if the specified entry does not exist</i></li> <li>• <i>RM2M_REG_ERROR_ISNULL if the value of the specified entry is set to "null"</i></li> <li>• <i>&lt; OK, if another error occurs (see "Error codes for the registration memory block access operations" in chapter "Constants" on page 134)</i></li> </ul>



**native rM2M\_RegSetString(reg, const name[], const string[]);**

Writes a character string into a registration memory block. Detailed information on the registration memory blocks is provided in chapter "Registration memory blocks" on page 37.

**Important note:** This function accepts even characters which are forbidden according to JSON standard, such as "\t", "\n", ... Due to this, Javascript's JSON.parse() will fail with error messages. Use JSON5 instead to decode such extended strings.

Parameter	Explanation
reg	Index of the registration memory block (see "Indices of the registration memory blocks" in chapter "Constants" on page 134)  <b>Note:</b> RM2M_REG_APP_STATE requires "rM2M_RegInit ()" before.
name	Name of the entry  If an entry with this name already exists, the existing character string is replaced by the transferred character string. Otherwise a new entry is created.
string	Array that contains the string to be written

	Explanation
Return value	<ul style="list-style-type: none"> <li>• OK, if successful</li> <li>• ERROR, if an unspecified error occurs</li> <li>• &lt; OK, if another error occurs (see "Error codes for the registration memory block access operations" in chapter "Constants" on page 134)</li> </ul>

**native rM2M\_RegSetValue(reg, const name[], {Float,Fixed,}:value, tag=tagof value);**

Writes a value into a registration memory block. Detailed information on the registration memory blocks is provided in chapter "Registration memory blocks" on page 37.

Parameter	Explanation
reg	Index of the registration memory block (see "Indices of the registration memory blocks" in chapter "Constants" on page 134)  <b>Note:</b> RM2M_REG_APP_STATE requires "rM2M_RegInit ()" before.
name	Name of the entry  If an entry with this name already exists, the existing value is replaced by the transferred value. Otherwise a new entry is created.
value	Value to be written
tag	The integer and floating-point conversion are differentiated by the "tag" of the value. - OPTIONAL

	Explanation
Return value	<ul style="list-style-type: none"> <li>• OK, if successful</li> <li>• ERROR, if an unspecified error occurs</li> <li>• &lt; OK, if another error occurs (see "Error codes for the registration memory block access operations" in chapter "Constants" on page 134)</li> </ul>

---

**native rM2M\_RegDelValue(reg, const name[]);**

Searches for an entry based on its name and sets the value of this entry (regardless of whether it is a string or value) to "null". Detailed information on the registration memory blocks is provided in chapter "Registration memory blocks" on page 37.

<b>Parameter</b>	<b>Explanation</b>
<i>reg</i>	<i>Index of the registration memory block (see "Indices of the registration memory blocks" in chapter "Constants" on page 134)</i>
<i>name</i>	<i>Name of the entry for which the value should be set to "null"</i>

	<b>Explanation</b>
<i>Return value</i>	<ul style="list-style-type: none"><li>• <i>OK, if successful</i></li><li>• <i>ERROR, if an unspecified errors occurs</i></li><li>• <i>&lt; OK, if another error occurs (see "Error codes for the registration memory block access operations" in chapter "Constants" on page 134)</i></li></ul>

**native rM2M\_RegDelKey(reg, const name[]);**

Searches for an entry based on its name and deletes the entry from the registration memory block. Detailed information on the registration memory blocks is provided in chapter "Registration memory blocks" on page 37.

<b>Parameter</b>	<b>Explanation</b>
<i>reg</i>	<i>Index of the registration memory block (see "Indices of the registration memory blocks" in chapter "Constants" on page 134)</i>
<i>name</i>	<i>Name of the entry that should be deleted from the registration memory block</i>

	<b>Explanation</b>
<i>Return value</i>	<ul style="list-style-type: none"><li>• <i>OK, if successful</i></li><li>• <i>ERROR, if an unspecified errors occurs</i></li><li>• <i>&lt; OK, if another error occurs (see "Error codes for the registration memory block access operations" in chapter "Constants" on page 134)</i></li></ul>

**native rM2M\_RegOnChg(funcidx);**

Specifies the function that should be called up if one of the registration memory blocks has changed (i.e. has been updated by the server). The callback is not triggered upon local (device-side) changes of a registration memory. Detailed information on the registration memory blocks is provided in chapter "Registration memory blocks" on page 37.

<b>Parameter</b>	<b>Explanation</b>
<i>funcidx</i>	Index of the public function that should be called up if the registration has changed  Type of function: <i>public func(reg);</i>

	<b>Explanation</b>
<b>Return value</b>	<ul style="list-style-type: none"> <li>• OK, if successful</li> <li>• &lt; OK, if an error occurs (see "Return codes for general purposes" in chapter "Constants" on page 106)</li> </ul>

**13.3.6 Position****13.3.6.1 Arrays with symbolic indices****TrM2M\_GSMPos**

Information about a GSM/UMTS/LTE cell in the receiving range

```
// mcc      MCC (Mobile Country Code) of the GSM cell
// mnc      MNC (Mobile Network Code) of the GSM cell
// lac      LAC (Location Area Code) of the GSM cell
// cellid   Cell ID of the GSM cell
// rssi     Detected GSM level [dBm] for the GSM cell
// ta       TA (Timing Advance) of the GSM cell (currently always 0)

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

**TrM2M\_PosUpdateGSM**

Information about a GSM cell in the receiving range

```
// type     specifies the type of the entry (RM2M_POSUPDATE_TYPE_GSM)
// stamp    Time when data was recorded
// mcc      MCC (Mobile Country Code) of the GSM cell
// mnc      MNC (Mobile Network Code) of the GSM cell
// lac      LAC (Location Area Code) of the GSM cell
// cid      Cell ID of the GSM cell
// rssi     Detected GSM level [dBm] for the GSM cell
// ta       TA (Timing Advance) of the GSM cell (currently always 0)

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

---

## TrM2M\_PosUpdateUMTS

*Information about a UMTS cell in the receiving range*

```
// type          specifies the type of the entry (RM2M_POSUPDATE_TYPE_UMTS)
// stamp         Time when data was recorded
// mcc           MCC (Mobile Country Code) of the GSM cell
// mnc           MNC (Mobile Network Code) of the GSM cell
// lac           LAC (Location Area Code) of the GSM cell
// cid           Cell ID of the GSM cell
// rscp          Received Signal Code Power [dBm]
// pscr          Primary Scrambling Code

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

## TrM2M\_PosUpdateLTE

*Information about an LTE cell in the receiving range*

```
// type          specifies the type of the entry (RM2M_POSUPDATE_TYPE_LTE)
// stamp         Time when data was recorded
// mcc           MCC (Mobile Country Code) of the GSM cell
// mnc           MNC (Mobile Network Code) of the GSM cell
// lac           LAC (Location Area Code) of the GSM cell
// cid           Cell ID of the GSM cell
// rsrp          Reference Signal Received Power [dBm]

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

## TrM2M\_PosUpdateWiFi

*Information about a WiFi network in the receiving range*

```
// type          specifies the type of the entry (RM2M_POSUPDATE_TYPE_WIFI)
// stamp         Time when data was recorded
// ch            WiFi RF channel (1-14 in 2.4 GHz frequency range), data type: u8
// rssi          WiFi RSSI Level [dBm], data type: s8
// bssid         Basic Service Set Identification (e.g. MAC address of the
//              access point)
// ssid          Service Set Identifier of the WiFi network

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

**TNMEA\_GGA**

*Information (position, height above sea level and accuracy) extracted from a GGA data record*

```
// Lat      geographical latitude in degrees (resolution: 0.000001°)
//          -90,000,000 = South pole 90° S,
//          0 = Equator,
//          +90,000,000 = North pole 90° N
//
// Long     geographical longitude in degrees (resolution: 0.000001°)
//          -180,000,000 =180° West, 0 =Zero meridian, +180,000,000 =180° East
//
// Alt      Height above sea level in meters
// Qual     NMEA Quality indicator(see "Constants" on page 141)
// SatUsed  Number of satellites used for the positioning
// HDOP     relative accuracy of the horizontal position [0,01]

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

**13.3.6.2 Constants****List of the supported types of cell/network information entries**

*Possible types of cell/network information entries that can be read from the system via the function "rM2M\_EnumPosUpdate()"*

```
RM2M_POSUPDATE_TYPE_ERR = 0, //invalid entry
RM2M_POSUPDATE_TYPE_GSM = 1, //Information about a GSM cell
RM2M_POSUPDATE_TYPE_UMTS = 2, //Information about a UMTS cell
RM2M_POSUPDATE_TYPE_LTE = 3, //Information about an LTE cell
RM2M_POSUPDATE_TYPE_WIFI = 4, //Information about a WiFi network
```

**NMEA error codes**

*Error codes of the function rM2M\_SetPosNMEA()*

```
RM2M_NMEA_ERR_DATATYPE = -2, // Data type (e.g. $GGA) not supported.
RM2M_NMEA_ERR_SENTENCE = -3, // Sentence invalid (e.g. checksum error)
RM2M_NMEA_ERR_LATITUDE = -4, // Geographical latitude invalid
RM2M_NMEA_ERR_LONGITUDE = -5, // Geographical longitude invalid
RM2M_NMEA_ERR_ALTITUDE = -6, // Altitude above sea level invalid
RM2M_NMEA_ERR_SAT_USED = -7, // Number of satellites used invalid.
RM2M_NMEA_ERR_QUAL = -8, // GPS quality indication not supported.
```

---

## NMEA quality indicator

```
RM2M_NMEA_FIX_NOK      = 0,    // invalid/no fix
RM2M_NMEA_FIX_GPS      = 1,    // Non-differential GPS fix
RM2M_NMEA_FIX_DGPS     = 2,    // Differential GPS fix
RM2M_NMEA_FIX_PPS     = 3,    // Precise positioning service (PPS)
RM2M_NMEA_FIX_RTK     = 4,    // Real time kinematic (RTK)
RM2M_NMEA_FIX_FLOATRTK = 5,    // Float real time kinematic
RM2M_NMEA_FIX_EST      = 6,    // Estimated fix (dead reckoning, coupled
                               // navigation)
RM2M_NMEA_FIX_MAN      = 7,    // Manual input mode
RM2M_NMEA_FIX_SIM      = 8,    // Simulation mode
```

## List of supported GNSS device IDs

*Designed to identify the source of the NMEA data record (in accordance with the "Talker ID" used with the NMEA 0183 standard)*

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

## List of supported NMEA data records

```
RM2M_NMEA_RECORD_GGA = 0x474741, // GGA (global positioning system fix data)
```

### 13.3.6.3 Functions

#### native `RM2M_SetPos(Lat, Long, Elev, Qual, SatUsed)`;

Saves the GPS position information in the device. A historical record is not maintained. This means that the current position information always overwrites the last known position. The information is transmitted to the myDatatnet server and can, for example, be read out via the API (see "API" on page 219).

<b>Parameter</b>	<b>Explanation</b>
<i>Lat</i>	geographical latitude in degrees (resolution: 0.000001°)  -90 000 000 = South pole 90° south 0 = Equator +90 000 000 = North pole 90° north
<i>Long</i>	geographical longitude in degrees (resolution: 0.000001°)  -180 000 000 = 180° west 0 = Zero meridian (Greenwich) +180 000 000 = 180° east
<i>Elev</i>	Height above sea level in meters (Valid range: -999...+9999)
<i>Qual</i>	Quality indicator (GPS quality indicator)  <i>RM2M_NMEA_FIX_NOK:</i> <i>invalid/no fix</i> <i>RM2M_NMEA_FIX_GPS:</i> <i>non-differential GPS fix</i> <i>RM2M_NMEA_FIX_DGPS:</i> <i>differential GPS fix</i> <i>RM2M_NMEA_FIX_PPS:</i> <i>Precise positioning service (PPS)</i> <i>RM2M_NMEA_FIX_RTK:</i> <i>Real time kinematic (RTK)</i> <i>RM2M_NMEA_FIX_FLOATRTK:</i> <i>Float real time kinematic</i> <i>RM2M_NMEA_FIX_EST:</i> <i>Estimated fix (dead reckoning, coupled navigation)</i>  <i>RM2M_NMEA_FIX_MAN:</i> <i>Manual input mode</i> <i>RM2M_NMEA_FIX_SIM:</i> <i>Simulation mode</i>
<i>SatUsed</i>	Number of satellites used for the positioning (valid range: 0 ... 99)

	<b>Explanation</b>
<i>Return value</i>	<ul style="list-style-type: none"> <li>• OK, if successful</li> <li>• ERROR</li> </ul> <p><b>Note:</b> The parameters are checked against the specified range limits. The function returns "ERROR" if the limits are not adhered to.</p>

**native rM2M\_DecodeNMEA(const sentence[], data[], len=sizeof data);**

*Decodes a transferred NMEA data record*

<b>Parameter</b>	<b>Explanation</b>
<i>sentence</i>	NMEA data record from a GPS receiver starting with the '\$' character.  <b>Important note:</b> The strings must be terminated ('\0') immediately after the checksum.
<i>data</i>	Buffer (cell array) to store the decoded data  [0]: Contains the GNSS device ID (see "List of supported GNSS device IDs" in chapter "Constants" on page 141)  [1]: Contains the type of decoded NMEA data record (see "List of supported NMEA data records" in chapter "Constants" on page 141)  [2] ... [n]: Dependent on type of decoded NMEA data record  For a type "RM2M_NMEA_RECORD_GGA" data record, the remaining structure is the same as the "TNMEA_GGA" structure.  [2]: .Lat [3]: .Long [4]: .Alt [5]: .Qual [6]: .SatUsed [7]: .HDOP
<i>len</i>	Size (in cells) of the buffer to record the decoded data – OPTIONAL

	<b>Explanation</b>
<i>Return value</i>	<ul style="list-style-type: none"> <li>• positive value, if successful (number of filled array elements, i.e. cells)</li> <li>• negative value, if an error has occurred (see "NMEA error codes" in chapter "Constants" on page 141)</li> </ul>



**native rM2M\_SetPosNMEA(const Sentence{});**

Takes the GPS position information from the transferred NMEA data record and saves it in the device. A historical record is not maintained. This means that the current position information always overwrites the last known position. The information is transmitted to the myDatanet server and can, for example, be read out via the API (see "API" on page 219).

<b>Parameter</b>	<b>Explanation</b>
Sentence	<p>NMEA data record from a GPS receiver starting with the '\$' character. The following data records are currently supported:</p> <ul style="list-style-type: none"> <li>• \$GPGGA - location specification (fix information)</li> </ul> <p><b>Important note:</b> The strings must be terminated ('\0') immediately after the checksum.</p>

	<b>Explanation</b>
Return value	<ul style="list-style-type: none"> <li>• OK, if successful</li> <li>• &lt; OK, if an error occurs (see "NMEA error codes" in chapter "Constants" on page 141)</li> </ul>

**native rM2M\_GetPos(&Lat, &Long, &Elev, &Qual=0, &SatUsed=0);**

*Reads out the GPS position information saved to the device*

<b>Parameter</b>	<b>Explanation</b>
<i>Lat</i>	Variable to store the geographical latitude in degrees (resolution: 0.000001°)  -90 000 000 = South pole 90° south 0 = Equator +90 000 000 = North pole 90° north
<i>Long</i>	Variable to store the geographical longitude in degrees (resolution: 0.000001°)  -180 000 000 = 180° west 0 = Zero meridian (Greenwich) +180 000 000 = 180° east
<i>Elev</i>	Variable to store the height above sea level in metres (valid range: -999...+9999)
<i>Qual</i>	Variable to store the quality indicator (GPS quality indicator) – OPTIONAL  RM2M_NMEA_FIX_NOK: invalid/no fix RM2M_NMEA_FIX_GPS: non-differential GPS fix RM2M_NMEA_FIX_DGPS: differential GPS fix RM2M_NMEA_FIX_PPS: Precise positioning service (PPS) RM2M_NMEA_FIX_RTK: Real time kinematic (RTK) RM2M_NMEA_FIX_FLOATRTK: Float real time kinematic RM2M_NMEA_FIX_EST: Estimated fix (dead reckoning, coupled navigation)  RM2M_NMEA_FIX_MAN: Manual input mode RM2M_NMEA_FIX_SIM: Simulation mode
<i>SatUsed</i>	Variable to store the number of satellites used for positioning – OPTIONAL

	<b>Explanation</b>
<i>Return value</i>	<ul style="list-style-type: none"> <li>• OK if valid GPS position information is stored in the device</li> <li>• &lt; OK, if an error occurs (see "Return codes for general purposes" in chapter "Constants" on page 106)</li> </ul>

**native rM2M\_PosUpdate(...)**

Makes it possible to transfer information about the GSM/UMTS/LTE cells and WiFi networks to the device. The position data is thus generated based on this information provided by the user. This function uses a variable list of parameters. The parameters to be passed depend on the purpose. The following procedure is recommended:

## 1. Preparing a new position data record

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

The internally saved list of cell/network information is discarded and a new one is prepared. The time stamp of this new list is set to the current time.

## 2. Transferring the cell/network information entries to the system

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

The transferred cell/network information entry is entered in the list that was created during step 1. The ".type" and ".stamp" entries in the "TrM2M\_PosUpdatexxx" structure are ignored. A maximum of 20 cell/network information entries can be entered in the list.

**Note:** A suitable type of cell/network information for the "TrM2M\_PosUpdatexxx" structure transferred in the "sData" data buffer must be specified for the "type".

## 3. The position data record is completed by transferring a "Dummy" cell/network information entry with a length of 0.

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

The list of cell/network information entries is sorted (in descending order based on the reception level) and the time stamp is updated. The time stamp is used for the synchronisation of the position data with the server. If the device is currently in online mode, the position data is immediately transferred to the server. If the cell/network information entries are read out again via the "rM2M\_EnumPosUpdate()" function, the time stamp generated during step 3 is entered in the "TrM2M\_PosUpdatexxx" structure.

Parameter	Explanation
type	Type of cell/network information transferred to the system (see "RM2M_POSUPDATE_TYPE_xxx" in chapter "Constants" on page 141)
sData	Cell/network information entry that should be transferred to the system. The data structure is dependent on the selected type ("type" parameter).  e.g. "TrM2M_PosUpdateWiFi" (see TrM2M_PosUpdateWiFi in chapter "Arrays with symbolic indices" on page 139) for type "RM2M_POSUPDATE_TYPE_WIFI".
len	<ul style="list-style-type: none"> <li>• Size (in cells) of the cell/network information entry that was copied into the "sData" data buffer</li> <li>• Zero to indicate that no further cell/network information entries, that should be taken into consideration for the current position data record, will be transferred to the system. --&gt;</li> </ul>

---

	<b>Explanation</b>
<i>Return value</i>	<ul style="list-style-type: none"><li>• <i>OK, if successful</i></li><li>• <i>ERROR_NOT_SUPPORTED</i></li><li>• <i>ERROR-1, if temporarily not possible (e.g. internal position update active),</i></li><li>• <i>&lt; OK, if another error occurs (see "Return codes for general purposes" in chapter "Constants" on page 106).</i></li></ul>

**native rM2M\_EnumPosUpdate(...);**

lists the information saved in the device about the GSM/UMTS/LTE cells and WiFi networks in the receiving range. With this function a variable list of parameters is used. The parameters to be passed depend on the purpose. The following procedure is recommended:

1. Reading of the number of available cell/network information entries

```
new nEnum;

rM2M_EnumPosUpdate (nEnum) ;
```

2. Determination of the particular type of the cell/network information entries

```
new type;
new idxEnum = 0;

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

3. Reading of cell/network information entries based on the types determined previously (in the following example only those that contain information about a GSM cell).

```
new sGSMPos[TrM2M_PosUpdateGSM] ;

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

<b>Parameter</b>	<b>Explanation</b>
<i>nEnum</i>	Variable to store the number of available cell/network information entries
<i>idxEnum</i>	Index of the cell/network information entry whose type should be determined or that should be read by the system.  Either the "type" parameter or the two "buf" and "len" parameters are required in addition depending on the desired action.
<i>type</i>	Variable to store the type of a cell/network information entry (see "RM2M_POSUPDATE_TYPE_xxx" in Chapter "Constants" on page 141)
<i>buf</i>	Buffer to store a cell/network information entry  The structure of the buffer depends on the cell/network information entry to be read (see "TrM2M_PosUpdatexxx" in Chapter "Arrays with symbolic indices" on page 139)
<i>len</i>	Size (in cells) of the structure to store a cell/network information entry

	<b>Explanation</b>
<b>Return value</b>	<ul style="list-style-type: none"> <li>• OK, if successful</li> <li>• ERROR, if an error occurs</li> </ul>

**native rM2M\_GetGSMPos(posidx, pos[TrM2M\_GSMPos]=0);**

Returns the number of GSM/UMTS/LTE cells for which valid information is saved to the device (*posidx < 0*) or reads out the information saved to the device about a GSM/UMTS/LTE cell in the receiving range (*posidx >= 0*)

**Note:** Use the "rM2M\_EnumPosUpdate()" function in order to get information on WiFi networks in the receiving range or more specific information on UMTS and/or LTE cells.

<b>Parameter</b>	<b>Explanation</b>
<i>posidx</i>	<p>Selection of the information returned by the function</p> <p><i>posidx &lt; 0:</i> Read the number of GSM/UMTS/LTE cells for which valid information is saved to the device</p> <p><i>posidx &gt;=0:</i> Number of the GSM/UMTS/LTE cell information block that should be read</p>
<i>pos</i>	<p><i>posidx &lt; 0:</i> Not required</p> <p><i>posidx &gt;=0:</i> Structure for storing the information about a GSM/UMTS/LTE cell in the receiving range (see "TrM2M_GSMPos" in chapter "Arrays with symbolic indices" on page 139)</p>

	<b>Explanation</b>
<i>Return value</i>	<p><i>posidx &lt; 0:</i> Number of GSM/UMTS/LTE cells for which valid information is saved to the device (max. 10)</p> <p><i>posidx &gt;=0:</i></p> <ul style="list-style-type: none"> <li>• OK, if the desired cell information block contains valid data of a GSM cell</li> <li>• OK+1, if the desired cell information block contains valid data of a UMTS cell</li> <li>• OK+2, if the desired cell information block contains valid data of a LTE cell</li> <li>• ERROR</li> </ul>

### 13.3.7 Math

#### Helpful constants

Definition	Value	Description
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	$\pi$
M_PI_2	1.57079632679489661923	$\pi/2$
M_PI_4	0.78539816339744830962	$\pi/4$
M_1_PI	0.31830988618379067154	$1/\pi$
M_2_PI	0.63661977236758134308	$2/\pi$
M_2_SQRTPI	1.12837916709551257390	$2/\sqrt{\pi}$
M_SQRT2	1.41421356237309504880	$\sqrt{2}$
M_SQRT1_2	0.70710678118654752440	$1/\sqrt{2}$

#### native fround(Float:x);

*Commercially rounds the transferred float*

Parameter	Explanation
x	Float that should be rounded

	Explanation
Return value	Commercially rounded integral value

#### native min(value1, value2);

*Supplies the smaller of the two transferred values*

Parameter	Explanation
value1	Two values of which the smaller one is to be determined
value2	

	Explanation
Return value	The smaller of the two transferred values

---

**native max(value1, value2);**

*Supplies the larger of the two transferred values*

<b>Parameter</b>	<b>Explanation</b>
<i>value1</i>	<i>Two values of which the larger one is to be determined</i>
<i>value1</i>	

	<b>Explanation</b>
<i>Return value</i>	<i>The larger of the two transferred values</i>

**native clamp(value, min=cellmin, max=cellmax);**

*Checks whether the transferred value is between "min" and "max"*

<b>Parameter</b>	<b>Explanation</b>
<i>value</i>	<i>Value that is to be checked</i>
<i>min</i>	<i>Lower limit</i>
<i>max</i>	<i>Upper limit</i>

	<b>Explanation</b>
<i>Return value</i>	<ul style="list-style-type: none"><li>• <i>"value" if the value is between "min" and "max"</i></li><li>• <i>"min" is the value is less than "min"</i></li><li>• <i>"max", if the value is greater than "max"</i></li></ul>

**native swapchars(c);**

*Swaps the order of the bytes*

<b>Parameter</b>	<b>Explanation</b>
<i>c</i>	<i>Value for which the bytes should be swapped over</i>

	<b>Explanation</b>
<i>Return value</i>	<i>Value for which the bytes in parameter "c" are swapped over (the lowest byte becomes the highest byte)</i>

The mode of operation of the following functions corresponds to that of the standard ANSI-C implementation:

**native Float:sin(Float:x);**

*Sine of x*

**native Float:cos(Float:x);**

*Cosine of x*

**native Float:tan(Float:x);**

*Tangent of x*



- 
- native Float:asin(Float:x);**  
*Arcsine(x) in the range  $[-\pi/2, \pi/2]$ , x element of  $[-1, 1]$*
- native Float:acos(Float:x);**  
*Arccosine(x) in the range  $[0, \pi]$ , x element of  $[-1, 1]$*
- native Float:atan(Float:x);**  
*Arctangent(x) in the range  $[-\pi/2, \pi/2]$*
- native Float:atan2(Float:y, Float:x);**  
*Arctangent(y/x) in the range  $[-\pi, \pi]$*
- native Float:sinh(Float:x);**  
*Hyperbolic sine of x*
- native Float:cosh(Float:x);**  
*Hyperbolic cosine of x*
- native Float:tanh(Float:x);**  
*Hyperbolic tangent of x*
- native Float:exp(Float:x);**  
*Exponential function  $e^x$*
- native Float:log(Float:x);**  
*Natural logarithm  $\ln(x)$ ,  $x > 0$*
- native Float:log10(Float:x);**  
*Logarithm as the basis 10  $\log_{10}(x)$ ,  $x > 0$*
- native Float:pow(Float:x, Float:y);**  
 *$x^y$ . An argument error has occurred if  $x = 0$  and  $y \leq 0$ , or if  $x < 0$  and y is not a whole number.*
- native Float:sqrt(Float:x);**  
*Square root x,  $x \geq 0$*
- native Float:ceil(Float:x);**  
*Smallest whole number that is not smaller than x*
- native Float:floor(Float:x);**  
*Largest whole number that is not larger than x*
- native Float:fabs(Float:x);**  
*Absolute value  $|x|$*
- native Float:ldexp(Float:x, n);**  
 *$x \cdot 2^n$*
- native Float:frexp(Float:x, &n);**  
*Breaks down x into a normalised mantissa in the range  $[1/2, 1]$  that is supplied as the result, and a potency of 2 that is filed in n. If x is zero, both parts of the result are zero.*
- native Float:modf(Float:x, &Float:ip);**  
*Breaks down x into an integral and residual part that both have the same prefix as x. The integral part is filed in ip, while the residual part is the result.*
- native Float:fmod(Float:x, Float:y);**  
*Residual floating point of x/y with the same prefix as x. The result is dependent on the implementation, if y is zero.*
- native isnan(Float:x);**  
*Returns a value that is not equal to zero, if x is not a number*
-

---

### 13.3.8 64 bit signed integer

**native s64\_set(val\_s64{8}, val);**

*transmits the content of a standard variable of the script language (32 bit signed integer) into an array which is designed for storing a 64 bit signed integer value.*

<b>Parameter</b>	<b>Explanation</b>
<i>val_s64</i>	<i>Array for storing a 64 bit signed integer value</i>
<i>val</i>	<i>Variable of which the content should be transmitted into a 64 bit signed integer value</i>

	<b>Explanation</b>
<i>Return value</i>	<ul style="list-style-type: none"><li>• <i>OK, if successful</i></li><li>• <i>ERROR, if an invalid parameter has been transmitted</i></li></ul>

**native s64\_get(val\_s64{8}, &val);**

*transmits a 64 bit signed integer value into a standard variable of the script language*

**Note:** *The standard variables of the script language are designed for storing only 32 bit signed integer values. If the value saved in the array exceeds the 32 bit signed integer values margin, the transmitted value in "val" is invalid as the top 4 bytes are cut off. It is recommended to check if the 32 bit signed integer values margin has been exceeded before reading out the data using the "s64\_cmp()" function.*

<b>Parameter</b>	<b>Explanation</b>
<i>val_s64</i>	<i>Array containing the 64 bit signed integer value</i>
<i>val</i>	<i>Variable for storing the value to be read</i>

	<b>Explanation</b>
<i>Return value</i>	<ul style="list-style-type: none"><li>• <i>OK, if successful</i></li><li>• <i>ERROR, if an invalid parameter has been transmitted</i></li></ul>

**native s64\_add(term1{8}, term2{8}, sum{8});**

*sums up two 64 bit signed integer values (sum = term1 + term2)*

<b>Parameter</b>	<b>Explanation</b>
<i>term1</i>	<i>Arrays containing the two values to be summed up</i>
<i>term2</i>	
<i>sum</i>	<i>Array for storing the result of the addition</i>

	<b>Explanation</b>
<i>Return value</i>	<ul style="list-style-type: none"><li>• <i>OK, if successful</i></li><li>• <i>ERROR, if an invalid parameter has been transmitted</i></li></ul>

**native s64\_sub(minuend{8}, subtrahend{8}, difference{8});**

*subtracts a 64 bit signed integer value from another one (difference = minuend - subtrahend)*

<b>Parameter</b>	<b>Explanation</b>
<i>minuend</i>	<i>Array containing the value from which should be subtracted</i>
<i>subtrahend</i>	<i>Array containing the value which should be subtracted</i>
<i>difference</i>	<i>Array for storing the result of the subtraction</i>

	<b>Explanation</b>
<i>Return value</i>	<ul style="list-style-type: none"> <li>• <i>OK, if successful</i></li> <li>• <i>ERROR, if an invalid parameter has been transmitted</i></li> </ul>

**native s64\_mul(multiplier{8}, multiplicand{8}, product{8});**

*multiplies two 64 bit signed integer values (product = multiplier \* multiplicand )*

<b>Parameter</b>	<b>Explanation</b>
<i>multiplier</i>	<i>Arrays containing the two values to be multiplied</i>
<i>multiplicand</i>	
<i>product</i>	<i>Array for storing the result of the multiplication</i>

	<b>Explanation</b>
<i>Return value</i>	<ul style="list-style-type: none"> <li>• <i>OK, if successful</i></li> <li>• <i>ERROR, if an invalid parameter has been transmitted</i></li> </ul>

**native s64\_div(dividend{8}, divisor{8}, quotient{8});**

*divides a 64 bit signed integer value by another one (quotient = dividend / divisor)*

<b>Parameter</b>	<b>Explanation</b>
<i>dividend</i>	<i>Array containing the value to be divided</i>
<i>divisor</i>	<i>Array containing the value to divide</i>
<i>quotient</i>	<i>Array for storing the result of the division</i>

	<b>Explanation</b>
<i>Return value</i>	<ul style="list-style-type: none"> <li>• <i>OK, if successful</i></li> <li>• <i>ERROR, if an invalid parameter has been transmitted</i></li> </ul>

---

**native s64\_mod(dividend{8}, divisor{8}, remainder{8});**

determines the remainder of the division of 64 bit signed integer values ( $remainder = dividend \% divisor$ )

<b>Parameter</b>	<b>Explanation</b>
<i>dividend</i>	Array containing the value to be divided
<i>divisor</i>	Array containing the value to divide
<i>remainder</i>	Array for storing the remainder remaining in the integer division

	<b>Explanation</b>
<i>Return value</i>	<ul style="list-style-type: none"><li>• OK, if successful</li><li>• ERROR, if an invalid parameter has been transmitted</li></ul>

**native s64\_lshift(val{8}, bits);**

shifts the 64 bit signed integer value contained in the array "bits" bits to the left ( $val \ll= bits$ ). The freed up bits are filled up with 0. This is an arithmetic shift, i.e. the leading sign is retained.

<b>Parameter</b>	<b>Explanation</b>
<i>val</i>	Array containing the value
<i>bits</i>	Number of bits that the value should be shifted to the left

	<b>Explanation</b>
<i>Return value</i>	<ul style="list-style-type: none"><li>• OK, if successful</li><li>• ERROR, if an invalid parameter has been transmitted</li></ul>

**native s64\_rshift(val{8}, bits);**

shifts the 64 bit signed integer value contained in the array "bits" bits to the right ( $val \gg= bits$ ). The freed up bits are filled up with 0. This is an arithmetic shift, i.e. the leading sign is retained.

<b>Parameter</b>	<b>Explanation</b>
<i>val</i>	Array containing the value
<i>bits</i>	Number of bits that the value should be shifted to the right

	<b>Explanation</b>
<i>Return value</i>	<ul style="list-style-type: none"><li>• OK, if successful</li><li>• ERROR, if an invalid parameter has been transmitted</li></ul>

**native s64\_cmp(val1{8}, val2{8});**

*compares the 64 bit signed integer values val1 and val2*

<b>Parameter</b>	<b>Explanation</b>
<i>val1</i>	<i>Arrays containing the two values to be compared</i>
<i>val2</i>	

	<b>Explanation</b>
<i>Return value</i>	<ul style="list-style-type: none"> <li>• 1: <i>val1 &gt; val2</i></li> <li>• 0: <i>the two values are equal</i></li> <li>• -1: <i>val1 &lt; val2</i></li> </ul>

### 13.3.9 Char & String

The mode of operation of the following functions essentially corresponds to that of the standard ANSI-C implementation:

**native strlen(const string[]);**

*Returns the length of the string (without '\0')*

<b>Parameter</b>	<b>Explanation</b>
<i>string</i>	<i>Character string for which the length has to be determined</i>

	<b>Explanation</b>
<i>Return value</i>	<i>Number of characters without the final '\0'</i>

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

*Saves the transferred format string in the array dest. The mode of operation of the functions corresponds to that of the "sprintf" function of the standard ANSI-C implementation.*

**Note:**

- If resulting string is longer than <dest>'s size, the very last character is set to terminating zero.
- <dest> size is always rounded up to full multiple of 4.

<b>Parameter</b>	<b>Explanation</b>
<i>dest</i>	<i>Array to store the formatted result</i>
<i>maxlength</i>	<i>Maximum number of characters that the array dest can store</i>
<i>format</i>	<p><i>The format character string to be used (C-style formatting codes)</i></p> <p><i>%b : Number in binary radix</i></p> <p><i>%c : Character</i></p> <p><i>%d : Number in decimal radix</i></p> <p><i>%f : Floating point number</i></p> <p><i>%s : String</i></p> <p><i>%x : Number in hexadecimal radix</i></p> <p><i>... : s32   f32   astr - Additional arguments.</i></p> <p><i>Depending on the format string, the function may expect a sequence of additional arguments, each containing a value to be used to replace a format specifier in the format string.</i></p>

	<b>Explanation</b>
<i>Return value</i>	<ul style="list-style-type: none"> <li>• <i>-1 in the event of a fault</i></li> <li>• <i>Number of characters that would have been written if the array dest had been long enough (without '\0').</i></li> </ul> <p><i>The array dest is always assigned a final zero. The length of the array dest cannot be exceeded.</i></p>

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

*Copies the source character string to the array dest (including '\0').*

<b>Parameter</b>	<b>Explanation</b>
<i>dest</i>	<i>Array to store the character string that should be copied</i>
<i>source</i>	<i>Character string that should be copied</i>
<i>maxlength</i>	<i>Size (in cells) of the array to store the character string to be copied - OPTIONAL</i>

	<b>Explanation</b>
<i>Return value</i>	<i>Number of copied characters</i>

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

Adds the source character string to the dest character string (including '\0')

**Important note:** Both strings must be zero-terminated.

<b>Parameter</b>	<b>Explanation</b>
dest	Array to store the result. This array already contains one character string to which the source character string should be added.
source	Character string that should be added to the character string included in the array dest
maxlength	Size (in Cells) of the array to store the result - OPTIONAL

	<b>Explanation</b>
Return value	Number of added characters

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

Compares character string1 and string2

**Important note:** Both strings must be zero-terminated.

<b>Parameter</b>	<b>Explanation</b>
string1	The two character strings that are to be compared
string2	
length	The maximum number of characters that should be taken into consideration during the comparison - OPTIONAL

	<b>Explanation</b>
Return value	<ul style="list-style-type: none"> <li>• 1: string1 &gt; string 2</li> <li>• 0: both of the character strings are the same (at least the length that is taken into account)</li> <li>• -1: string1 &lt; string 2</li> </ul>

**native strchr(const string[], char);**

Searches for a character (first occurrence) in a character string

<b>Parameter</b>	<b>Explanation</b>
string	Character string that should be searched
char	Character that the search is looking for

	<b>Explanation</b>
Return value	<ul style="list-style-type: none"> <li>• -1, if the character that the search is looking for is not included in the character string</li> <li>• Array index of the character that the search is looking for (first character occurring in the character string)</li> </ul>

---

**native strrchr(const string[], char);**

Searches for a character (last occurrence) in a character string

<b>Parameter</b>	<b>Explanation</b>
<i>string</i>	Character string that should be searched
<i>char</i>	Character that the search is looking for

	<b>Explanation</b>
<i>Return value</i>	<ul style="list-style-type: none"><li>• -1, if the character that the search is looking for is not included in the character string</li><li>• Array index for the character that the search is looking for (last character occurring in the character string)</li></ul>

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

Searches for the position of the first character in *string1* that is **not** included in the character string of permitted characters (*string2*)

<b>Parameter</b>	<b>Explanation</b>
<i>string1</i>	Character string that should be searched
<i>string2</i>	Character string of permitted characters

	<b>Explanation</b>
<i>Return value</i>	<ul style="list-style-type: none"><li>• Length of <i>string1</i> if no forbidden characters are found</li><li>• Position of the first character in the character string that should be searched that is not included in the character string of permitted characters</li></ul>

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

Searches for the position of the first character in *string1* that is also included in the character string of permitted characters (*string2*)

**Note:** See similar function *strpbrk* () which has a slightly different result.

<b>Parameter</b>	<b>Explanation</b>
<i>string1</i>	Character string that should be searched
<i>string2</i>	Character string of permitted characters

	<b>Explanation</b>
<i>Return value</i>	<ul style="list-style-type: none"><li>• Length of <i>string1</i> if no permitted character has been found</li><li>• Position of the first character in the character string that should be searched that is also included in the character string of permitted characters</li></ul>



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

Searches the array index of the first character that is also included in the character string of permitted characters

**Note:** See similar function `strcspn()` which has a slightly different result.

<b>Parameter</b>	<b>Explanation</b>
<i>string1</i>	Character string that should be searched
<i>string2</i>	Character string of permitted characters

	<b>Explanation</b>
<i>Return value</i>	<ul style="list-style-type: none"> <li>• -1: If no permitted character has been found</li> <li>• <math>\geq 0</math>: Array index of the first character in the character string that should be searched that is also included in the character string of permitted characters</li> </ul>

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

Searches character *string2* in character *string1*

<b>Parameter</b>	<b>Explanation</b>
<i>string1</i>	Character string to search in
<i>string2</i>	Character string to search for

	<b>Explanation</b>
<i>Return value</i>	<ul style="list-style-type: none"> <li>• -1: if character <i>string2</i> that is being searched for is not included in <i>string1</i></li> <li>• <math>\geq 0</math>: Array index where character <i>string2</i> that is being searched for starts in <i>string1</i></li> </ul>

---

**native strtol(const string[], base);**

*Converts a character string into a value*

**Note:**

- *Function differs slightly from it's C variant.*
- *Parsing consumes as many characters as possible, up to the first char not matching with given base.*

<b>Parameter</b>	<b>Explanation</b>
<i>string</i>	<i>Character string to be converted</i>  <b>Important note:</b> <i>Strings &gt; 128 bytes are not supported!</i>
<i>base</i>	<i>Specifies the basis that must be used for the conversion</i>  <i>2-36: The specified basis is used</i> <i>0: 8, 10 or 16 is used as the basis, depending on the character string to be converted</i> <i>Basis 8: with a leading 0</i> <i>Basis 16: with 0x or 0X</i> <i>Base 10: default</i>

	<b>Explanation</b>
<i>Return value</i>	<i>Value that corresponds to the character string</i>

**native Float: atof(const string[]);**

*Converts a character string into a float*

**Note:**

- *Decimal separator is always ".", no thousands separators supported.*
- *Parsing consumes as many characters as possible, up to the first none-float char.*

<b>Parameter</b>	<b>Explanation</b>
<i>string</i>	<i>Character string to be converted</i>  <b>Important note:</b> <i>Strings &gt; 128 bytes are not supported!</i>

	<b>Explanation</b>
<i>Return value</i>	<i>Float for which the numerical value corresponds to the character string</i>

**native memcpy\_native(dst{}, const dstofs, const src{}, const srcofs, const bytes, const dst\_cells=sizeof dst, const src\_cells=sizeof src);**

*Copies bytes from one buffer to another one*

<b>Parameter</b>	<b>Explanation</b>
<i>dst</i>	<i>Target buffer to which the data should be copied</i>
<i>dstofs</i>	<i>Position (byte offset) in the target buffer to which the data should be copied</i>
<i>src</i>	<i>Source buffer from which the data should be copied</i>
<i>srcofs</i>	<i>Position (byte offset) within the source buffer from which the data should be copied</i>
<i>bytes</i>	<i>Number of bytes that should be copied</i>
<i>dst_cells</i>	<i>Size (in cells) of the target buffer - OPTIONAL</i>
<i>src_cells</i>	<i>Size (in cells) of the source buffer - OPTIONAL</i>

	<b>Explanation</b>
<i>Return value</i>	<ul style="list-style-type: none"> <li>• <i>OK, if successful</i></li> <li>• <i>ERROR, if one of the following errors occurs</i> <ul style="list-style-type: none"> <li>• <i>If one of the two byte offsets or the number of bytes to be copied is &lt; 0</i></li> <li>• <i>If the byte offsets refer to a byte outside the relevant buffer</i></li> <li>• <i>If the reading was to exceed the source buffer (i.e. "Source byte offset" + "Number of bytes" would refer to a byte outside the source buffer)</i></li> <li>• <i>If the reading was to exceed the target buffer (i.e. "Target byte offset" + "Number of bytes" would refer to a byte outside the target buffer)</i></li> <li>• <i>If invalid buffers were transferred</i></li> </ul> </li> </ul>

**native memset\_native(dst{}, const dstofs, const srcval, const bytes, dstcells=sizeof dst);**

*Writes the desired value into the individual bytes of the transferred buffer*

<b>Parameter</b>	<b>Explanation</b>
<i>dst</i>	<i>Buffer in which the bytes should be set to the desired value</i>
<i>dstofs</i>	<i>Position (byte offset) within the transferred buffer from which the bytes should be set to the desired value</i>
<i>srcval</i>	<i>Value to which the individual bytes should be set</i>
<i>bytes</i>	<i>Number of bytes that should be set to the desired value</i>
<i>dstcells</i>	<i>Size (in cells) of the transferred buffer - OPTIONAL</i>

	<b>Explanation</b>
<i>Return value</i>	<ul style="list-style-type: none"> <li>• <i>OK, if successful</i></li> <li>• <i>ERROR, if one of the following errors occurs</i> <ul style="list-style-type: none"> <li>• <i>The byte offset is &lt; 0</i></li> <li>• <i>The byte offset refers to a byte outside the buffer</i></li> <li>• <i>If an invalid buffer was transferred</i></li> </ul> </li> </ul>

---

**native memcmp\_native(const src1{}, const src1ofs, const src2{}, const src2ofs, bytes, src1cells=sizeof src1, src2cells=sizeof src2);**

*Compares two buffers, byte for byte*

<b>Parameter</b>	<b>Explanation</b>
<i>src1</i>	<i>Buffer #1</i>
<i>src1ofs</i>	<i>Position (byte offset) within buffer #1 from which the bytes should be compared</i>
<i>src2</i>	<i>Buffer #2</i>
<i>src2ofs</i>	<i>Position (byte offset) within buffer #2 from which the bytes should be compared</i>
<i>bytes</i>	<i>Number of bytes that should be compared</i>
<i>src1cells</i>	<i>Size (in cells) of buffer #1 - OPTIONAL</i>
<i>src2cells</i>	<i>Size (in cells) of buffer #2 - OPTIONAL</i>

	<b>Explanation</b>
<i>Return value</i>	<ul style="list-style-type: none"> <li>• <i>&gt; 0: Buffer #1 &gt; Buffer #2</i></li> <li>• <i>0: the content of both of the buffers is the same (at least the bytes that are taken into account)</i></li> <li>• <i>&lt;0: Buffer #1 &lt; Buffer #2</i></li> </ul>

**native tolower(c);**

*Converts a character into lower case*

<b>Parameter</b>	<b>Explanation</b>
<i>c</i>	<i>Character that should be converted to lower case</i>

	<b>Explanation</b>
<i>Return value</i>	<i>The lower case variant of the transferred character, if available, or the unchanged character code of "c" if the letter "c" does not have a lower case equivalent.</i>

**native toupper(c);**

*Converts a character into upper case*

<b>Parameter</b>	<b>Explanation</b>
<i>c</i>	<i>Character that should be converted to upper case</i>

	<b>Explanation</b>
<i>Return value</i>	<i>The upper case variant of the transferred character, if available, or the unchanged character code of "c" if the letter "c" does not have a upper case equivalent.</i>

## 13.3.10 CRC & hash

### 13.3.10.1 Arrays with symbolic indices

#### TMD5\_Ctx

*Context structure for the MD5 calculation*

```
// init    After being set to "0", the context structure can be used to
//         calculate a new hash. If a calculation should be implemented by
//         calling up the "MD5" function repeatedly, there must not be
//         write access to this element.
// tmp     no write access permitted, for internal use
```

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

### 13.3.10.2 Functions

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

*Returns the calculated modbus CRC16 of the transferred data*

<b>Parameter</b>	<b>Explanation</b>
<i>data</i>	<i>Array that contains the data for which the CRC16 should be calculated</i>
<i>len</i>	<i>Number of bytes that must be taken into consideration during the calculation</i>
<i>initial</i>	<i>Initial value for calculating the CRC16 - OPTIONAL</i>

	<b>Explanation</b>
<i>Return value</i>	<i>Calculated CRC16</i>

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

*Returns the calculated Ethernet CRC32 of the transferred data*

<b>Parameter</b>	<b>Explanation</b>
<i>data</i>	<i>Array that contains the data for which the CRC32 should be calculated</i>
<i>len</i>	<i>Number of bytes that must be taken into consideration during the calculation</i>
<i>initial</i>	<i>Initial value for calculating the CRC32 - OPTIONAL</i>

	<b>Explanation</b>
<i>Return value</i>	<i>Calculated CRC32</i>

---

**native MD5(data{}, len, hash{16}, ctx[TMD5\_Ctx] = [0]);**

*Calculates the MD5 hash for the transferred data. If the hash for a data block should be calculated by calling up this function several times (e.g. when receiving data in blocks), then the same context structure must be transferred every time the function is called up. The context structure must not be changed between function call-ups. If the hash can be calculated by calling up the function once (e.g. complete data block is already available), then it is not necessary to transfer its own context structure.*

<b>Parameter</b>	<b>Explanation</b>
<i>data</i>	<i>Array that contains the data for which the MD5 hash should be calculated</i>
<i>len</i>	<i>Number of bytes that must be taken into consideration during the calculation</i>
<i>hash</i>	<i>Array to store the calculated 128-bit hash value</i>
<i>ctx</i>	<i>Context structure for the MD5 calculation – OPTIONAL (required only if calculation uses multiple calls to MD5() )</i>

	<b>Explanation</b>
<i>Return value</i>	<i>---</i>

## 13.3.11 Various

### 13.3.11.1 Arrays with symbolic indices

#### **TablePoint**

*Two-column reference point table, integer data type*

```
// key      Column that is searched
// value    Column with the result values that need to be returned

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

#### **TablePointF**

*Two-column reference point table, float data type*

```
// key      Column that is searched
// value    Column with the result values that need to be returned

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

**TrM2M\_Id***Information for identifying the module/device*

```
// string      rapidM2M module identification (e.g. "rapidM2M M23 HW3.0")
// module      rapidM2M module type (e.g. "M23")
// hwmajor     Hardware: Major version number
// hwminor     Hardware: Minor version number
// sn          Device serial number (binary) in BIG endian format
//             E.g.: "010146AF251CED1C" --> "01" in sn{0}, "1C" in sn{7}
// fwmajor     Firmware: Major version number
// fwminor     Firmware: Minor version number
// ctx         Site title (context)
//             Empty string if no context available

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

**TRTM\_Data***Information regarding the runtime measurement*

```
// runtime      Determined runtime in [ms]
// instructions Number of executed instructions
// tmp          For internal use, no write access permitted

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

**13.3.11.2 Constants****Error codes for the "CalcTable" and "CalcTableF" functions**

```
const
{
    TAB_ERR_FLOOR = -1, // searched value lower than the first table entry
    TAB_ERR_CEIL = -2, // searched value higher than the last table entry
};
```

**13.3.11.3 Functions****native getapilevel();***Issues the implemented API level of the script engine*

	<b>Explanation</b>
<b>Return value</b>	<i>Implemented API level of the script engine</i> <ul style="list-style-type: none"> <li>• 1: Initial functionality</li> <li>• 2: Added function "exists ()" to check availability of runtime function</li> <li>• 3: Added function "loadmodule ()" to load a device logic module</li> </ul>

---

**native exists(const name[]);**

checks whether the required rapidM2M API function is supported by the device firmware

**Important note:** Use `getapilevel ()` upfront to check if `exists()` is available with your firmware version.

<b>Parameter</b>	<b>Explanation</b>
<i>name</i>	Name of the required rapidM2M API function

	<b>Explanation</b>
<i>Return value</i>	<ul style="list-style-type: none"><li>• <i>true</i>, if the function is available</li><li>• <i>false</i>, if the device firmware does not support the function</li></ul>

**native loadmodule(mod{});**

Loads a script module at the runtime. This enables the script engine to be extended by its own native functions. The implementation of operations as a native function means that processing speeds can be increased significantly in comparison to the implementation in script. A script module can contain several native functions. After calling up this function, the native functions contained in the script module can be used in the same way as the standard functions available in the script engine.

<b>Parameter</b>	<b>Explanation</b>
<i>mod</i>	Byte array that contains the script module to be loaded.

	<b>Explanation</b>
<i>Return value</i>	<ul style="list-style-type: none"><li>• <i>OK</i>, if successful</li><li>• <i>ERROR</i>, if an error occurs</li></ul>

**native rtm\_start(measurement[TRTM\_Data]);**

Starts a runtime measurement

**Important note:** Execution of concurrent measurements is not allowed.

<b>Parameter</b>	<b>Explanation</b>
<i>measurement</i>	Structure for storing the information regarding a runtime measurement  <b>Important note:</b> This structure must be persistent from calling up " <code>rtm_start()</code> " to calling up " <code>rtm_stop()</code> ".

	<b>Explanation</b>
<i>Return value</i>	<ul style="list-style-type: none"><li>• <i>OK</i>, if successful</li><li>• <i>ERROR</i>, if an error occurs</li></ul>



**native rtm\_stop(measurement[TRTM\_Data]);**

*Stops the runtime measurement and calculates the time in [ms] since the "rtm\_start()" function was called up and the instructions executed since then. The determined values are written in the ".runtime" and ".instructions" elements of the transferred structure to record the information regarding a runtime measurement.*

<b>Parameter</b>	<b>Explanation</b>
<i>measurement</i>	Structure for storing the information regarding a runtime measurement  <b>Important note:</b> This structure must be persistent from calling up "rtm_start()" to calling up "rtm_stop()".

	<b>Explanation</b>
<i>Return value</i>	<ul style="list-style-type: none"> <li>• OK, if successful</li> <li>• ERROR, if an error occurs</li> </ul>

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

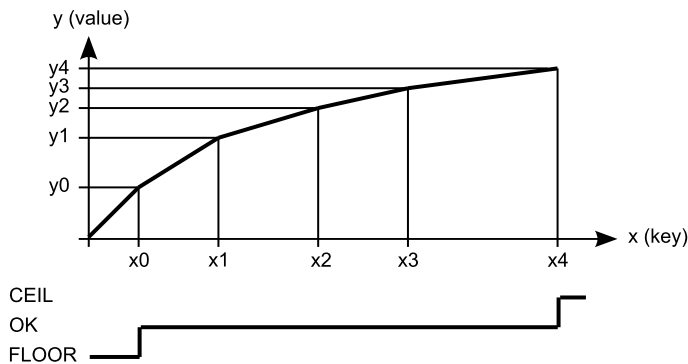
Searches for a certain value in the "key" column of the transferred reference point table and supplies the relevant value from the "value" column in the table. If the searched value is between two reference points, the returned value is interpolated linearly between the two adjacent values in the "value" column (linear equation:  $y = k*x + d$ ). Non-linear characteristic curves (e.g. connection between ADC value -> temperature) can be reproduced with this function.

Parameter	Explanation
key	Value that is used for the search
value	Includes the result of the calculation by the function
table	The table that is searched must be a "TablePoint" type table.
size	Number of rows in the table

	Explanation
Return value	<ul style="list-style-type: none"> <li>• OK, if the relevant value was found</li> <li>• TAB_ERR_FLOOR, if the searched value is lower than the first table entry. "value" contains the first table entry.</li> <li>• TAB_ERR_CEIL, if the searched value is higher than the last table entry. "value" contains the last table entry.</li> <li>• &lt; OK, if another error occurs (see "Return codes for general purposes" in chapter "Constants" on page 106).</li> </ul>

**Note:** Additional explanation on the "table" reference point table

The rows of the table can be displayed in an x/y coordinate system. The values in the "key" column are displayed on the X axis and the associated values in the "value" column are displayed on the Y axis.



Display of the reference point table as an x/y coordinate system

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

The functionality is the same as that of the "CalcTable" function. The difference is that "Float" is the data type for all elements of the "CalcTableF" function.

**native rM2M\_GetId(id[TrM2M\_Id], len=sizeof id);**

*Provides the information to identify the module/device*

<b>Parameter</b>	<b>Explanation</b>
<i>id</i>	<i>Structure for storing the information to identify the module/device (see "TrM2M_Id" in chapter "Arrays with symbolic indices" on page 166)</i>
<i>len</i>	<i>Size (in cells) of the structure to store the information - OPTIONAL</i>

	<b>Explanation</b>
<i>Return value</i>	<ul style="list-style-type: none"> <li>• <i>Used size (in cells) of the structure for storing the information</i></li> <li>• <i>ERROR if the address and/or length of the ID structure are invalid (outside the script data memory)</i></li> <li>• <i>&lt; OK, if another error occurs (see "Return codes for general purposes" in chapter "Constants" on page 106).</i></li> </ul> <p><b>Note:</b> <i>The firmware of a module/device detects if a device logic is being used for which the function only has one transfer parameter (older include file is being used) and for compatibility reasons therefore returns "OK" instead of the size of the structure for storing the information.</i></p>

**native heapSpace();**

*Supplies the free memory capacity to the heap*

	<b>Explanation</b>
<i>Return value</i>	<i>The free memory capacity to the heap. The stack and the heap have a joint memory area, so that this value specifies the number of bytes that remain for the stack or the heap.</i>

**native funcidx(const name[]);**

*Supplies the index of a public function. Used to register callbacks for the runtime environment.*

<b>Parameter</b>	<b>Explanation</b>
<i>name</i>	<i>Name of the public function</i>

	<b>Explanation</b>
<i>Return value</i>	<ul style="list-style-type: none"> <li>• <i>-1, if there is no function with the transferred name</i></li> <li>• <i>Index of the public function</i></li> </ul>

**native numargs();**

*Returns the number of arguments transferred to a function. This is useful within functions with a variable list of arguments.*

	<b>Explanation</b>
<i>Return value</i>	<i>The number of arguments that have been transferred to a function.</i>

---

**native getarg(arg, index=0);**

*This function supplies an argument from a variable argument list. If the argument is an array, the "index" specifies the index of the required array element.*

<b>Parameter</b>	<b>Explanation</b>
<i>arg</i>	<i>The sequence number of the argument. Use 0 for the first argument.</i>
<i>index</i>	<i>Index if "arg" refers to an array</i>

	<b>Explanation</b>
<i>Return value</i>	<i>The value of the argument.</i>

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

*Sets the value of the argument*

<b>Parameter</b>	<b>Explanation</b>
<i>arg</i>	<i>The sequence number of the argument. Use 0 for the first argument.</i>
<i>index</i>	<i>Index if "arg" refers to an array</i>
<i>value</i>	<i>Value to which the argument should be set</i>

	<b>Explanation</b>
<i>Return value</i>	<ul style="list-style-type: none"><li><i>true, if the value could be set</i></li><li><i>false, if the argument or index are invalid</i></li></ul> <p><i>This function sets an argument in a variable argument list. If the argument is an array, the "index" specifies the index of the required array element.</i></p>

**native rand();**

*Returns a random number from the "32-Bit signed Integer" value range. However, value "-1" (ERROR) is reserved for returning an error.*

	<b>Explanation</b>
<i>Return value</i>	<ul style="list-style-type: none"><li><i>Random number from the "32-Bit signed Integer" value range</i></li><li><i>ERROR if no random number generator is available</i></li></ul>

**native delay\_us(us);**

*Blocking delay function. The execution of the device logic is stopped and the following code line is only executed once the delay time has expired.*

<b>Parameter</b>	<b>Explanation</b>
<i>us</i>	<i>Delay time (1...10000 [μs]).</i>

	<b>Explanation</b>
<i>Return value</i>	<ul style="list-style-type: none"> <li>• <i>OK, if successful</i></li> <li>• <i>ERROR, if an error occurs</i></li> </ul>

**13.3.12 Console****native print(const string[]);**

*Prints the specified string to the standard output*

<b>Parameter</b>	<b>Explanation</b>
<i>string</i>	<i>The character string to be issued. This can include escape sequences.</i>

	<b>Explanation</b>
<i>Return value</i>	<i>OK</i>

---

**native printf(const format[], {Float,Fixed,\_}:...);**

Prints the transferred format string to the standard output. The mode of operation of the functions corresponds to that of the standard ANSI-C implementation.

**Note:**

- Characters may get lost if console output buffer overflows.
- Use `sprintf ()` to write to a string buffer instead of the console.

<b>Parameter</b>	<b>Explanation</b>
<i>format[]</i>	<p>The format character string to be used (C-style formatting codes)</p> <p><i>%b</i> : Number in binary radix <i>%c</i> : Character <i>%d</i> : Number in decimal radix <i>%f</i> : Floating point number <i>%s</i> : String <i>%x</i> : Number in hexadecimal radix ... : <i>s32   f32   astr</i> - Additional arguments.</p> <p>Depending on the format string, the function may expect a sequence of additional arguments, each containing a value to be used to replace a format specifier in the format string.</p>

	<b>Explanation</b>
<i>Return value</i>	<ul style="list-style-type: none"><li>• Number of printed characters</li><li>• <i>ERROR</i>, if not successful</li></ul>

**native setbuf(buf{}, size);**

Provides the firmware with a buffer from the RAM area reserved for the device logic that is used to output strings via the "printf()" function. When this function is called up, the system switches from the 256 byte buffer integrated in the firmware to the transferred buffer.

**Important note:** The buffer must be valid during the entire use by the firmware (i.e. it must be defined as a global or static variable).

<b>Parameter</b>	<b>Explanation</b>
<i>buf</i>	Static byte array that should be used as a buffer to output strings
<i>size</i>	Size of the buffer in bytes  <b>Note:</b> If the function is called up again and the size is set to "0" during the process, then the system switches back to the integrated buffer (256 bytes). The transferred static byte array can then be used by the device logic again.

	<b>Explanation</b>
<i>Return value</i>	<ul style="list-style-type: none"><li>• <i>OK</i>, if successful</li><li>• <i>ERROR</i>, if not successful</li></ul>

### 13.3.13 SMS

**Important note:** If the device is in "online" mode no SMS can be processed.

#### 13.3.13.1 Callback functions

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

Function to be provided by the device logic developer, that is called up if an SMS is received

<b>Parameter</b>	<b>Explanation</b>
<i>SmsTel</i>	String that contains the telephone number of the sender of the SMS
<i>SmsText</i>	String that contains the content of the SMS

#### 13.3.13.2 Functions

**native rM2M\_SmsInit(funcidx, config);**

Initialises SMS receipt

<b>Parameter</b>	<b>Explanation</b>
<i>funcidx</i>	Index of the public function that should be called up if an SMS has been received  Type of function: <code>public func(const SmsTel[], const SmsText[]);</code>  <b>Important note:</b> If an SMS that is longer than 160 characters is received, it is discarded immediately. The specified public function is not called up.
<i>config</i>	Reserved for extensions

	<b>Explanation</b>
<i>Return value</i>	<ul style="list-style-type: none"> <li>• OK, if successful</li> <li>• &lt; OK, if an error occurs (see "Return codes for general purposes" in chapter "Constants" on page 106)</li> </ul>

**native rM2M\_SmsClose();**

Deactivates SMS receipt

	<b>Explanation</b>
<i>Return value</i>	OK

---

## 13.3.14 File transfer

### 13.3.14.1 Arrays with symbolic indices

#### TFT\_Info

*Properties of a file entry*

```
// name      Name of the file
// stamp     Time stamp of the file (seconds since 31.12.1999)
// stamp256  Fraction of the next started sec. (resolution 1/256 sec.)
// size      File size in byte
// crc       Ethernet CRC32 of the file
// flags     File flags (see "File flags" in
//           chapter "Constants" on page 176)
#define TFT_Info[ .name{256}, .stamp, .stamp256, .size, .crc, .flags ]
```

### 13.3.14.2 Constants

#### File flags

```
FT_FLAG_READ   = 0x0001, // File can be read by the server.
FT_FLAG_WRITE  = 0x0002, // File can be written by the server.
FT_FLAG_NODE   = 0x0004 /* file nodes (required to entitle the server to
                        create a new file) */
FT_FLAG_SYSTEM = 0x0008 // System file (cannot be used by the device logic)
```

#### File transfer command

```
FT_CMD_NONE    = 0,
FT_CMD_UNLOCK  = 1, // File transfer session terminated. The server
                    releases the block again. */
FT_CMD_LIST    = 2, // The server requests the properties of a file
FT_CMD_READ    = 3, // The server requests a part of a file.
FT_CMD_STORE   = 4, // The server requests a file to be written.
FT_CMD_WRITE   = 5, /* The server provides a block to be written in
                    a file. */
FT_CMD_DELETE  = 6, // The server requests a file to be deleted.
FT_CMD_ENUM    = 7, /* The server requests the properties of a file
                    that are part of a file node */
FT_CMD_RETR    = 8, /* The server requests a file that is part of a
                    file node */
```

### 13.3.14.3 Callback functions

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

*Function to be provided by the device logic developer, that is called up when a file transfer command is received. The callback function must be able to handle all file transfer commands (see "File transfer commands" in chapter "Constants" on page 176).*



<b>Parameter</b>	<b>Explanation</b>																		
<i>id</i>	Unique identification with which the file is referenced (specified during registration)																		
<i>cmd</i>	File transfer command that was received from the system and that has to be processed by the callback function																		
<i>data</i>	<p>This parameter is only relevant when the following file transfer commands are received:</p> <ul style="list-style-type: none"> <li>• <i>FT_CMD_STORE</i>: Array that contains the properties of the file that should be newly created. Structure: <table border="1" data-bbox="571 607 1509 896"> <thead> <tr> <th><b>Offset</b></th> <th><b>Bytes</b></th> <th><b>Explanation</b></th> </tr> </thead> <tbody> <tr> <td>0</td> <td>4</td> <td>Time stamp of the file</td> </tr> <tr> <td>8</td> <td>4</td> <td>File size in byte</td> </tr> <tr> <td>12</td> <td>4</td> <td>Ethernet CRC32 of the file</td> </tr> <tr> <td>16</td> <td>2</td> <td>File flags</td> </tr> <tr> <td>18</td> <td>256</td> <td>File name</td> </tr> </tbody> </table> </li> <li>• <i>FT_CMD_WRITE</i>: Array that contains the data received from the myDatanet server.</li> <li>• <i>FT_CMD_RETR</i>: Name of a file (ASCII) that is part of a file node and was requested by the server</li> </ul>	<b>Offset</b>	<b>Bytes</b>	<b>Explanation</b>	0	4	Time stamp of the file	8	4	File size in byte	12	4	Ethernet CRC32 of the file	16	2	File flags	18	256	File name
<b>Offset</b>	<b>Bytes</b>	<b>Explanation</b>																	
0	4	Time stamp of the file																	
8	4	File size in byte																	
12	4	Ethernet CRC32 of the file																	
16	2	File flags																	
18	256	File name																	
<i>len</i>	<p>This parameter is only relevant when the following file transfer commands are received:</p> <ul style="list-style-type: none"> <li>• <i>FT_CMD_READ</i>: Number of bytes requested by the myDatanet server (max. 4kB)</li> <li>• <i>FT_CMD_STORE</i>: Size of the file property block received from the myDatanet server</li> <li>• <i>FT_CMD_WRITE</i>: Number of bytes received from the myDatanet server</li> <li>• <i>FT_CMD_RETR</i>: Length of the file name (number of characters excluding the final '\0')</li> </ul>																		
<i>ofs</i>	<p>This parameter is only relevant when the following file transfer commands are received:</p> <ul style="list-style-type: none"> <li>• <i>FT_CMD_READ</i>: Byte offset within the file of the data block to be transferred to the myDatanet server</li> <li>• <i>FT_CMD_WRITE</i>: Byte offset within the file of the data block received from the myDatanet server</li> <li>• <i>FT_CMD_ENUM</i>: List index (starting with 0) for when the server requests the properties of a file that belongs to a file node<sup>1)</sup></li> </ul>																		

<sup>1)</sup>Upon receipt of the file transfer command "FT\_CMD\_ENUM", the "FT\_SetPropsExt()" function can be used to set the properties of a file that should be assigned to the current file node. This means that a file is assigned to the file node. Following the first "FT\_CMD\_ENUM" command, the system sends further "FT\_CMD\_ENUM" commands until the device logic developer indicates that they do not want to assign any more files to the current file node. The developer must indicate this by setting the length for the "TFT\_Info" structure (i.e. the "len" parameter) to 0 when setting the file properties via the "FT\_SetPropsExt()" function.

---

### 13.3.14.4 Functions

**native FT\_Register(const name{}, id, funcidx);**

*Registers a file made available by the device logic.*

<b>Parameter</b>	<b>Explanation</b>
<i>name</i>	<i>Unique file name</i>
<i>id</i>	<i>Unique identification with which the file is subsequently referenced (freely selectable)</i>
<i>funcidx</i>	<i>Index of the public function that should be called up if a file transfer command has been received</i>  <i>Type of function: public func(id, cmd, const data{}, len, ofs);</i>  <b><i>Important note:</i></b> <i>All file transfer commands (see "File transfer commands" in chapter "Constants" on page 176) must be handled by this public function.</i>

	<b>Explanation</b>
<i>Return value</i>	<ul style="list-style-type: none"><li>• <i>OK, if successful</i></li><li>• <i>&lt; OK, if an error occurs (see "Return codes for general purposes" in chapter "Constants" on page 106)</i></li></ul>

**native FT\_RegisterEnum(id, funcidx, props[TFT\_Info], len=sizeof props);**

*Registers a file node made available by the device logic. Several files can be managed via a file node.*

<b>Parameter</b>	<b>Explanation</b>
<i>id</i>	<i>Unique identification with which the file node is subsequently referenced (freely selectable)</i>
<i>funcidx</i>	<p><i>Index of the public function that should be called up if a file transfer command has been received</i></p> <p><i>Type of function: public func(id, cmd, const data[], len, ofs);</i></p> <p><b>Important note:</b> <i>All file transfer commands (see "File transfer commands" in chapter "Constants" on page 176) must be handled by this public function.</i></p>
<i>props</i>	<p><i>Structure that contains the properties of a file entry for the file node (see "TFT_Info" in chapter "Arrays with symbolic indices" on page 176)</i></p> <p><i>.name: Those parts of the file node name that are not wildcards, must also appear in the names of the files that should be managed with the file node (wildcard matching), so that the read and write operations are accepted.</i></p> <p><i>.flags: Read/write flags as required; node flag mandatory</i></p>
<i>len</i>	<i>Size (in cells) of the structure that contains the properties of a file entry – OPTIONAL</i>

	<b>Explanation</b>
<i>Return value</i>	<ul style="list-style-type: none"> <li>• <i>OK, if successful</i></li> <li>• <i>&lt; OK, if an error occurs (see "Return codes for general purposes" in chapter "Constants" on page 106)</i></li> </ul>

**native FT\_Unregister(id);**

*Removes a file from the registration. The file is no longer available for the file transfer.*

<b>Parameter</b>	<b>Explanation</b>
<i>id</i>	<i>Unique identification with which the file is referenced (specified during registration)</i>

	<b>Explanation</b>
<i>Return value</i>	<ul style="list-style-type: none"> <li>• <i>OK, if successful</i></li> <li>• <i>&lt; OK, if an error occurs (see "Return codes for general purposes" in chapter "Constants" on page 106)</i></li> </ul>

---

**native FT\_SetProps(id, stamp, size, crc, flags);**

*Sets the properties of a file*

**Important note:** This function must be called up following receipt of a "FT\_CMD\_LIST" command.

**Important note:** Although this function will still be supported for the purpose of downward compatibility, it should no longer be used for new projects. The "FT\_SetPropsExt()" function should be used as an alternative.

<b>Parameter</b>	<b>Explanation</b>
<i>id</i>	<i>Unique identification with which the file is referenced (specified during registration)</i>
<i>stamp</i>	<i>Time stamp of the file (seconds since 31.12.1999)</i>
<i>size</i>	<i>File size in byte</i>
<i>crc</i>	<i>Ethernet CRC32 of the file</i>
<i>flags</i>	<i>File flags (see "File flags" in chapter "Constants" on page 176)</i>

	<b>Explanation</b>
<i>Return value</i>	<ul style="list-style-type: none"><li>• OK, if successful</li><li>• &lt; OK, if an error occurs (see "Return codes for general purposes" in chapter "Constants" on page 106)</li></ul>

**native FT\_SetPropsExt(id, props[TFT\_Info], len=sizeof props);**

*Sets the properties of a file (extended format)*

**Important note:** This function must be called up following receipt of a "FT\_CMD\_LIST" command.

<b>Parameter</b>	<b>Explanation</b>
<i>id</i>	<i>Unique identification with which the file is referenced (specified during registration)</i>
<i>props</i>	<i>Structure that contains the properties of a file entry (see "TFT_Info" in chapter "Arrays with symbolic indices" on page 176)</i>
<i>len</i>	<i>Size (in cells) of the structure that contains the properties of a file entry – OPTIONAL</i>

	<b>Explanation</b>
<i>Return value</i>	<ul style="list-style-type: none"><li>• OK, if successful</li><li>• &lt; OK, if an error occurs (see "Return codes for general purposes" in chapter "Constants" on page 106)</li></ul>

**native FT\_Read(id, const data[], len);**

Transmits the data to the system, to transfer it to the myDatagnet server. The data must be provided by the callback function specified via "FT\_Register()".

**Important note:** This function must be called up following receipt of a "FT\_CMD\_READ" command.

<b>Parameter</b>	<b>Explanation</b>
<i>id</i>	Unique identification with which the file is referenced (specified during registration)
<i>data</i>	Array that contains the data that should be transmitted to the system to be transferred to the myDatagnet server
<i>len</i>	Number of bytes to be transferred

	<b>Explanation</b>
<i>Return value</i>	<ul style="list-style-type: none"> <li>• OK, if successful</li> <li>• &lt; OK, if an error occurs (see "Return codes for general purposes" in chapter "Constants" on page 106)</li> </ul>

**native FT\_Accept(id, newid=-1);**

Accepts the file that the myDatagnet server wants to write. It is a new file if the transferred unique identification number ("id" parameter) refers to a file node. In this case, an unique identification number ("newid" parameter) must be assigned to the new file. The new file must also be registered via the "FT\_Register()" function. The file properties that were transmitted by the system to the callback function (see "Callback functions" on page 176) must be saved via the "FT\_SetProps()" function.

**Important note:** This function must be called up following receipt of a "FT\_CMD\_STORE" command.

<b>Parameter</b>	<b>Explanation</b>
<i>id</i>	Unique identification with which the file is referenced (specified during registration)
<i>newid</i>	Unique identification for the new file that should be created (-1 if it is not a new file) - OPTIONAL

	<b>Explanation</b>
<i>Return value</i>	<ul style="list-style-type: none"> <li>• OK, if successful</li> <li>• &lt; OK, if an error occurs (see "Return codes for general purposes" in chapter "Constants" on page 106)</li> </ul>

---

**native FT\_Written(id, len);**

Confirms that the data received from the myDatanet server has been written. The actual writing process must be executed via the callback function specified via "FT\_Register()". The data that is to be written is transmitted to the callback function by the system (see "Callback functions" on page 176).

**Important note:** This function must be called up following receipt of a "FT\_CMD\_WRITE" command.

<b>Parameter</b>	<b>Explanation</b>
<i>id</i>	Unique identification with which the file is referenced (specified during registration)
<i>len</i>	Number of written bytes

	<b>Explanation</b>
<i>Return value</i>	<ul style="list-style-type: none"><li>• OK, if successful</li><li>• &lt; OK, if an error occurs (see "Return codes for general purposes" in chapter "Constants" on page 106)</li></ul>

**native FT\_Error(id);**

Used to display a file handling error and terminates any file command.

<b>Parameter</b>	<b>Explanation</b>
<i>id</i>	Unique identification with which the file is referenced (specified during registration)

	<b>Explanation</b>
<i>Return value</i>	<ul style="list-style-type: none"><li>• OK, if successful</li><li>• &lt; OK, if an error occurs (see "Return codes for general purposes" in chapter "Constants" on page 106)</li></ul>

## 13.4 Device Logic error codes

If an error occurs while executing the Device Logic, the corresponding error code is entered in the device log and the Device Logic is restarted. If such an error occurs more than three times in 24 hours, the Device Logic is deactivated and error handling is activated (see "Error handling" on page 36). The parameter for all log entries except "SCRIPT\_ERR" contains the 32-bit instruction pointer of the Abstract machine (AMX). Two entries are generated in the device log as only 16-bit values can be saved in the parameter of a log entry. The first entry contains Bit31-Bit16 and the second entry contains Bit15-Bit0 of 32-bit instruction pointer. Instructions on evaluating the device log are included in the chapter "Log tab" (see ""Log" tab" on page 87).

Log entry		Parameter		Description
Code	Plain text	Code	Plain text	
3000	SCRIPT_ERR	0	NO SCRIPT	No valid Device Logic available
		1	SCRIPT UPDATE	New Device Logic received
		2	SCRIPT EXCEPT LOOP	Exception loop detected (4 system starts due to exception within 10 min.).  The Device Logic is deactivated and error handling is activated (see "Error handling" on page 36). The file system is also reformatted. This means that all of the data and log entries recorded to date are deleted. This is indicated by the additional "LOG REFORMATFILE" log entry.
		3	SCRIPT SOFT ERROR 1	First time a runtime error has occurred within 24 hours. Device Logic was restarted.
		4	SCRIPT SOFT ERROR 2	Second time a runtime error has occurred within 24 hours. Device Logic was restarted.
		5	SCRIPT SOFT ERROR 3	Third time a runtime error has occurred within 24 hours. Device Logic was restarted.  The Device Logic is deactivated and error handling is activated if another runtime error should occur within 24 hours (see "Error handling" on page 36).
		6	SCRIPT UPDATE ERROR	Error when installing the device logic received during the download.  The "Interval & Wakeup" connection type was activated and a connection to the server is established every hour.
		7	SCRIPT SYSTEM SHUTDOWN	Reserved for extensions
		8	SCRIPT DOWNLOAD ERROR	Connection aborted while downloading the device logic.  As the existing Device Logic is located in the RAM, it can continue to be executed until the next PowerOn. It can no longer be executed following a PowerOn and the "SCRIPT_ERR, NO SCRIPT" error is entered in the device log.
		9	SCRIPT DELETED	The device logic was deleted manually (e.g. using rapidM2M Studio).  The "Interval & Wakeup" connection type was activated and a connection to the server is established every 24 hours.

Log entry		Parameter		Description
Code	Plain text	Code	Plain text	
3001	AMX_ERR_EXIT	##	---	Abortion e.g. max. number of commands (100,000) per run reached
3002	AMX_ERR_ASSERT	##	---	Assertion failed
3003	AMX_ERR_STACKERR	##	---	Stack/heap collision (insufficient stack size)
3004	AMX_ERR_BOUNDS	##	---	Array index outside the valid range
3005	AMX_ERR_MEMACCESS	##	---	Invalid memory access e.g. mix-up between cell (32-bit element) access [] and byte access {}
3006	AMX_ERR_INVINSTR	##	---	Invalid statement
3007	AMX_ERR_STACKLOW	##	---	Stack underflow
3008	AMX_ERR_HEAPLOW	##	---	Heap underflow
3009	AMX_ERR_CALLBACK	##	---	No (invalid) native callback function
3010	AMX_ERR_NATIVE	##	---	Native function failed
3011	AMX_ERR_DIVIDE	##	---	Division by zero
3012	AMX_ERR_SLEEP	##	---	Sleep mode
3013	AMX_ERR_INVSTATE	##	---	Invalid state
3014	reserved			
3015	reserved			
3016	AMX_ERR_MEMORY	##	---	Out of memory
3017	AMX_ERR_FORMAT	##	---	P-code file format is invalid/not supported
3018	AMX_ERR_VERSION	##	---	File is for a newer version of AMX
3019	AMX_ERR_NOTFOUND	##	---	File or function not found
3020	AMX_ERR_INDEX	##	---	Invalid index parameter (invalid entry point)
3021	AMX_ERR_DEBUG	##	---	Debugger cannot be executed
3022	AMX_ERR_INIT	##	---	AMX not initialised (or initialised twice)



Log entry		Parameter		Description
Code	Plain text	Code	Plain text	
3023	AMX_ERR_USERDATA	##	---	User data field cannot be set (table full)
3024	AMX_ERR_INIT_JIT	##	---	JIT cannot be initialised.
3025	AMX_ERR_PARAMS	##	---	Faulty parameter
3026	AMX_ERR_DOMAIN	##	---	Domain error. The result of the expression is not in the valid range.
3027	AMX_ERR_GENERAL	##	---	General error (invalid or non-specific error)
3028	AMX_ERR_OVERLAY	##	---	Overlays are not supported (JIT) or are not initialised.
3050	LOG_NOSCRIPT_ERR	1	SCRIPT File not available	Meta information on the Device Logic not available
		2	SCRIPT File not fully downloaded	Inconsistency between the meta information on the Device Logic and the Device Logic itself detected
		3	SCRIPT Error reading file	Error detecting the Device Logic type
		4	SCRIPT Marked faulty	Device Logic is marked as faulty (4 runtime errors have occurred within 24 hours)
		5	SCRIPT Invalid type	Invalid Device Logic Typ detected
		6	SCRIPT Error loading program	Error initializing the Device Logic
		7	SCRIPT Exception loop detected	Exception Loop detected (4 system starts due to exception within 10 min.)
		8	SCRIPT Invalid filesize	If the size of the Binaries is $\leq 1$ Byte

---

## 13.5 Syntax

### 13.5.1 General syntax

#### 13.5.1.1 Format

Identifiers, numbers and characters are separated by spaces, tabs, line breaks and "form feed". A series of one or more of these separators is recognised as an empty space.

#### 13.5.1.2 Optional semicolons

Semicolons (used to finish a statement) are optional if they are at the end of a line. Semicolons are required to separate several statements in a line. An expression can be split across several lines, though the postfix operators must be on the same line as the operand.

#### 13.5.1.3 Comments

Text between the `/*` and `*/` symbols (both symbols can be on the same or different lines) and text following `//` (to the end of the line) are comments. Comments must not be nested. The compiler considers comments to be blank space. A documentation comment is a comment that starts with `/**` (two stars and space after the second star) and ends with `*/`. A comment that starts with `///  
/` (three forward slashes and a space after the third slash) is also a documentation comment. The parser can support the documentation comment in different ways, for example, by using it to generate online help.

#### 13.5.1.4 Identifier

Names of variables, functions and constants. Identifier comprises the characters `a...z`, `A...Z`, `0...9`, `_` or `@`. The first character must not be a number. The characters `@` and `_` on their own are not valid identifiers, e.g. `"_Up"` is a valid identifier but `"_"` is not. A distinction is made between upper and lower case. The parser cuts identifiers off after a certain length. By default, only the first 16 characters are referenced for distinguishing purposes.

#### 13.5.1.5 Reserved keywords

Statements	Operator	Directives	Others
assert break case continue default do else exit for goto if return sleep state switch while	defined sizeof state tagof	defined sizeof state tagof	defined sizeof state tagof

### 13.5.1.6 Numerical constants

#### 13.5.1.6.1 Numerical integer constants

##### Binary

*0b followed by a series of 0 and 1*

##### Decimal

*A series of numbers between 0 and 9*

##### Hexadecimal

*0x followed by a series of numbers between 0 and 9 and the letters a to f*

#### 13.5.1.6.2 Numerical floating-point constants

A floating-point number is a number with numbers after the decimal point. A floating-point number starts with one or several numbers, includes a decimal point and has at least one number after the decimal point, e.g. "12.0" and "0.75" are valid floating-point numbers. An exponent can optionally be added. The notation is the letter "e" (lower case) followed by an integer numerical constant. For example, "3.12e4" or "12.3e-3" are valid floating-point numbers with an exponent.

### 13.5.2 Variables

#### 13.5.2.1 Declaration

The keyword "new" declares a new variable. For special declarations, the keyword "new" is replaced with "static" (see "Static local declaration" on page 188). The value of the new variable is zero, provided that is not initialised explicitly.

A variable declaration can appear

- At every position at which an expression is valid - local variable
- At every position at which a function declaration or the implementation of the function is valid - global variables
- In the first expression of a "for" loop (see "For (expression 1; expression 2; expression 3) statement" on page 198) - local variable

Example:

```
new a;          // without initialisation (value is 0)
new b = 3;     // with initialisation (value is 3)
```

#### 13.5.2.2 Local declaration

A local declaration appears within a statement block. A variable can only be accessed within this block and the blocks that it comprises. A declaration within the first expression of a loop instruction is also a local declaration.

#### 13.5.2.3 Global declaration

A global declaration appears outside of a function and a global variable can be used in any function. Global variables can only be initialised with constant expressions.

---

### 13.5.2.4 Static local declaration

A local variable is destroyed if the execution leaves the block in which the variable was created. Local variables in a function only exist during the operating time of the specified function. Each new call up of the function creates and initialises new local variables. The variable will also remain in the memory at the end of the function, if a local variable is declared with the keyword "static" instead of "new". This means that static, local variables provide permanent private storage that can only be accessed by a single function (or block). Static local variables can only be initialised with constant expressions, the same way as global variables.

### 13.5.2.5 Static global declaration

A static global variable acts in the same way as a global variable with the difference that the variable is only valid in the file in which it was declared. Replace the keyword "new" with "static" to declare a global variable as static.

### 13.5.2.6 Floating point values

Floating point values are supported. These can be added at every point at which a variable declaration is valid.

Example:

```
new Float:a;           // without initialisation (value is 0.0)
new Float:b = 3.0;    // with initialisation (value is 3.0)
```

## 13.5.3 Constant variables

It is sometimes necessary to create a variable that is initialised once and is then not meant to be changed again. Such a variable acts in a similar way to a symbolic constant although it is still a variable. To declare a constant variable, place the keyword "const" between the keyword that starts the variable declaration ("new", "static") and the name of the variables.

Example:

```
new const address[4] = { 192, 0, 168, 66 }
static const status /* initialised to zero */
```

Typical situations in which you could use a constant variable, include:

- To create an "array" constant. Symbolic constants cannot be accessed via the index.
- It is a special case when array arguments are marked as "const" in a function. Arrays arguments are always transferred via a reference. If the arguments are declared to be "const", they are protected against unwanted changes. See examples of "const function arguments" in the chapter "Function arguments ("call-by-value" versus "call-by-reference")" on page 201.

## 13.5.4 Array variables

### 13.5.4.1 One-dimensional arrays

The name[constant] syntax declares the name as an array of "constant" elements, where each element is an entry. "name" is a placeholder for the name of the variable and "constant" is a positive value not equal to zero. "constant" is optional and can be omitted. If there is no value between the brackets, the number of elements is equal to the number of initial values. The array index area is "zero-based", which means that the first element is "name[0]" and the last element is "name[constant-1]".

### 13.5.4.2 Initialisation

Data objects can be initialised during their declaration. The initialised value from global data objects must be a constant value. Global or local arrays must also be initialised with constant values. Data that is not initialised are zero by default.

Example:

List: valid declaration

```
new i = 1
new j          /* j is 0 */
new k = 'a'    /* k has the character code of 'a' */
new a[] = [1,4,9,16,25] /* a has 5 elements */
new s1[20] = ['a','b'] /* the remaining 18 elements are 0 */
new s2[] = ''Hello world...'' /* an unpacked string */
```

List: invalid declaration

```
new c[3] = 4          /* An array cannot be set to an individual
                       value */
new i = "Good-bye"   /* only an array can hold a string */
new q[]              /* Unknown size for an array */
new p[2] = { i + j, k - 3 } /* Array initialisers must be constants */
```

### 13.5.4.3 Progressive initialisation for arrays

The point operator continues the initialisation of the arrays based on the last two initialised values. The point operator (three points, "...") initialises the array up to the array limit.

Example: List: array initialisers

```
new a[10] = { 1, ... } // sets all of the elements to 1
new b[10] = { 1, 2, ... } // b = 1, 2, 3, 4, 5, 6, 7, 8, 9, 10
new c[8] = { 1, 2, 40, 50, ... } // c = 1, 2, 40, 50, 60, 70, 80, 90
new d[10] = { 10, 9, ... } // d = 10, 9, 8, 7, 6, 5, 4, 3, 2, 1
```

### 13.5.4.4 Multi-dimensional arrays

(Only arrays with up to three dimensions are supported)

Multi-dimensional arrays are arrays that include references to other sub-arrays. For example, a two-dimensional array is an "array on one-dimensional arrays".

Example for the declaration of two-dimensional 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'' ]
```

---

As the last two declarations (variables "e" and "f") illustrates, the last dimension has an unspecified length. In this case, the length of the sub-array is detected by the associated initialiser. Each sub-array is a different length. In this specific example, "e[1][5]" includes the letter "l" of the word "Cancel". However, "e[0][5]" is invalid as the sub-array e[0] only comprises three entries (the letters "O", "K" and the zero terminator). The difference between the declarations of the "e" and "f" arrays is that we enable the compiler to determine the number of higher dimensions for "f". "sizeof f" and "sizeof e" are both 2 (see "Arrays and the "sizeof" operator" on page 190).

#### 13.5.4.5 Arrays and the "sizeof" operator

The "sizeof" operator returns the number of elements of a variable. The "sizeof" result of a simple (non array) variable is always 1.

An array with one dimension comprises a number of elements and the "sizeof" operator returns this quantity. The code section below would therefore issue "5", as the array comprises four characters and the zero terminator.

```
new msg[] = 'Help'  
printf('%d', sizeof msg);
```

The "sizeof" operator always returns the number of entries even for a "packed" array. The code section below also issues "5", as the variable comprises five entries even though it requires less memory space.

```
new msg{} = "Help"  
printf('%d', sizeof msg);
```

For multi-dimensional arrays, the "sizeof" operator can return the number of elements for every dimension. An element in the last (lowest) dimension is a single entry, while it is a sub-array in the highest dimension. Please note that in the following code section, the "sizeof matrix" syntax returns the number of elements of the higher dimension and that the "sizeof matrix[]" syntax issues the lower dimension of the two-dimensional array. The code section issues three (higher dimension) and two (lower dimension).

```
new matrix[3][2] = { { 1, 2 }, { 3, 4 }, { 5, 6 } }  
printf('%d %d', sizeof matrix, sizeof matrix[]);
```

The application of the "sizeof" operator on multi-dimensional arrays is particularly practical when it is used as a standard value for function arguments.

## 13.5.5 Operators and expressions

### 13.5.5.1 Notational conventions

The use of some operators is dependent on the relevant type of operand. The following notations are therefore used in this chapter:

**e**

*Any expression*

**v**

*Any expression that can be assigned a value ("lvalue" expression - variable)*

**a**

*An array*

**f**

*A function*

**s**

*A symbol - this can be a variable, a constant or a function*

### 13.5.5.2 Expressions

An expression consists of one or several operands with an operator. The operand can be a variable, a constant or another expression. An expression followed by a semicolon is a statement.

Examples of expressions:

```
v++ f(a1, a2)
v = (ia1 * ia2) / ia3
```

### 13.5.5.3 Arithmetic

Operator	Example	Explanation
+	e1 + e2	Result of adding e1 and e2
-	e1 - e2	Result of subtracting e2 from e1
	-e	Result of the arithmetic negation of e (two's complement)
*	e1 * e2	Result of multiplying e1 with e2
/	e1 / e2	Result of dividing e1 by e2. The result is truncated to the closest whole number that is less or equal to the quotient. Positive and negative values are rounded down (negative infinity).
%	e1 % e2	Result is the remainder of the division of e1 by e2. The prefix is the same as that of e2
++	v++	Increases v by 1. The result of the expression is the value before the increase.
	++v	Increases v by 1. The result of the expression is the value following the increase.
--	v--	Decreases v by 1. The result of the expression is the value before the decrease.
	--v	Decreases v by 1. The result of the expression is the value following the decrease.

**Note:** The unary + is not defined. The operators ++ and -- change the operand. The operand must be a "lvalue".

### 13.5.5.4 Bit manipulation

Operator	Example	Explanation
~	~e	The result is the one's complement of e.
>>	e1 >> e2	The result of the arithmetic shift to the right of e1 by e2 bits. The shift is signed: The bit on the far left is copied to the free bits of the result.
>>>	e1 >>> e2	The result of the logical shift to the right of e1 by e2 bits. The shift is unsigned. The free bits of the result are filled with 0.
<<	e1 << e2	Result: Shift to the left of e1 by e2 bits. The free bits of the result are filled with 0. There is no difference between an arithmetic and a logical shift to the left.
&	e1 & e2	The result is the bitwise logical "and" of e1 and e2.
	e1   e2	The result is the bitwise logical "or" of e1 and e2.
^	e1 ^ e2	The result is the bitwise logical "exclusive or" of e1 and e2.

### 13.5.5.5 Assignment

The result of an assignment expression is the value of the operand following the assignment.

Operator	Example	Explanation
=	v = e	Assigns the value of e to the variable v
	v = a	Assigns the array a to variable v. v must be an array of the same size and with the same dimensions as a. a can be a character string or an array.

**Note:** The following operators combine an assignment with an arithmetic or bitwise operation. The result of the expression is the value of the left operand following the arithmetic or bitwise operation.

Operator	Example	Explanation
+=	v += e	Increases v by e
-=	v -= e	Decreases v by e
*=	v *= e	Multiplies v with e
/=	v /= e	Divides v by e
%=	v %= e	Assigns v the remainder of the division of v and e
>>=	v >>= e	Arithmetically shifts v to the right by e bits
>>>=	v >>>= e	Logically shifts v to the right by e bits
<<=	v <<= e	Shifts v to the left by e bits
&=	v &= e	Executes a bitwise "and" from v and e and assigns the result to v
=	v  = e	Executes a bitwise "or" from v and e and assigns the result to v
^=	v ^= e	Executes a bitwise "exclusive or" from v and e and assigns the result to v



### 13.5.5.6 Comparative operators

A logical "false" is represented by an integer value of 0; a logical "true" is represented by a value that is not 0. Results of a comparative expression are either 0 or 1 and the "tag" is set to "bool".

Operator	Example	Explanation
==	e1 == e2	The result is "true" if e1 and e2 are the same.
!=	e1 != e2	The result is "true" if e1 and e2 are not the same.

**Note:** The following operators can be linked, the same as in the expression "e1 <= e2 <= e3". This means that the result is "1" if every single comparison is true and "0" if at least one comparison is false.

Operator	Example	Explanation
<	e1 < e2	The result is a logical "true" if e1 is less than e2.
<=	e1 <= e2	The result is a logical "true" if e1 is less or equal to e2.
>	e1 > e2	The result is a logical "true" if e1 is greater than e2.
>=	e1 >= e2	The result is a logical "true" if e1 is greater or equal to e2.

### 13.5.5.7 Boolean

A logical "false" is represented by an integer value of 0; a logical "true" is represented by a value that is not 0. Results of a comparative expression are either 0 or 1 and the "tag" is set to "bool".

Operator	Example	Explanation
!	!e	The result is a logical "true", if e is logical "false".
	e1    e2	The result is "true", if either e1 or e2 (or both) are logical "true". The expression e2 is only evaluated if e1 is logical "false".
&&	e1 && e2	The result is "true" if e1 and e2 are logical "true". The expression e2 is only evaluated if e1 is logical "true".

### 13.5.5.8 Other

Operator	Example	Explanation
[]	a[e]	Array index: The result is the entry at position e of array a.
{}	a{e}	Array index: The result is the index at position e of "packed" array a:
()	f(e1, e2, ... eN)	The result is the value that is returned by function f. The function is called up with parameters e1, e2, ... eN. The sequence of the evaluation of the parameters is not defined. (The implementation of the script engine may evaluate the parameters in reverse order.)
? :	e1 ? e2 : e3	The result is either e2 or e3, depending on the value of e1. The conditional expression is a composite expression with a two-part operator, "?" and ":". The expression e2 is evaluated if e1 is logical "true"; e3 is evaluated if e1 is logical "false".
:	tagname: e	"Tag" overwritten: The value of the expression does not change, although the "tag" does change.
defined	defined s	Result is "1" if the symbol was defined. The symbol can be a constant or a global or local variable. The "tag" of the expression is "bool"
sizeof	sizeof s	The result is the number of elements of the specified variable. An element is an entry for simple variables and for one dimensional arrays. For multi-dimensional arrays, the result is the number of elements (sub-arrays) in the highest dimension. Add [] to the name of the array to specify a lower dimension. The result is 0 if the size of the variable is not known. If this operator is used in a "default" value of a function, the expression is executed at the time that the function was called up and not at the time that the definition was completed.
tagof	tagof s	The result is a unique number that represents the "tag" of the variables, the constants, the return value of a function or the name of the "tag" title. If this operator is used in a "default" value of a function, the expression is executed at the time that the function was called up and not at the time that the definition was completed.

### 13.5.5.9 Priority of the operators

The following table groups the operators with the same priority, starting with the highest priority at the top of the table.

If the evaluation of an expression is not explicitly justified with brackets, it is categorised by the association rules. For example:  $a*b/c$  is equal to  $(a*b)/c$  based on the left to right association, and  $a=b=c$  is equal to  $a=(b=c)$ .

Operator	Explanation	Reading order
()	Function call	left-to-right
[]	array index (element)	
{}	array index (character)	
!	logical not	right-to-left
~	one's complement	
-	two's complement (unary minus)	
++	increase	
--	decrease	
:	"tag" overwritten	
defined	symbol definition status	
sizeof tagof	symbol size in "elements" unique number of the tag	
*	multiplication	left-to-right
/	division	
%	modulo	
+	addition	left-to-right
-	subtraction	
>>	arithmetic shift to the right	left-to-right
>>>	logical shift to the right	
<<	shift to the left	
&	bitwise "and"	left-to-right
^	bitwise "exclusive or"	left-to-right
	bitwise "or"	left-to-right
<	less than	left-to-right
<=	less or equal to	
>	greater than	
>=	greater or equal to	
==	equal	left-to-right
!=	unequal	
&&	logical "and"	left-to-right
	logical "or"	left-to-right
?:	conditional execution	right-to-left
=	Assignment *= /= %= += -= >>= >>>= <<= &= ^=  =	right-to-left
,	comma	left-to-right

### 13.5.6 Statements

A statement can comprise one or several lines. A line can comprises two or more statements.

Statements for the sequence control (if, if-else, for, while, do-while and switch) can be nested.

#### 13.5.6.1 Statement label

A label consists of an identifier followed by a ":". A label is a "Jump target" of a "goto" statement.

---

Each statement can be marked with a label. The label must be followed by a statement, which can also be an "empty statement".

The scope of a label is the function in which it was declared, i.e. a "goto" statement cannot jump from the current function to another function.

### 13.5.6.2 Composite statements

A composite statement (also known as a block) is a series of zero or several statements that is enclosed by brackets ("{" and "}"). The closing bracket ("}") must not be followed by a semicolon. Each statement can be replaced by a block. A composite statement that does not comprise any statements is a special case and is known as an "empty statement".

### 13.5.6.3 Expression statement

Each expression becomes a statement when a semicolon (";") is added. An expression also becomes a statement, if the expression is only followed by blank spaces to the end of the line, and the expression is not continued in the next line.

### 13.5.6.4 Empty statement

An empty statement does not execute any statements and consists of a block statement without statements, i.e. it consists of the "{}" symbol. Empty statements are implemented in control flow statements without actions (e.g. "while (!iskey()) {}") or if a label is defined exactly before the closing bracket of a block statement. An empty statement does not end with a semicolon.

### 13.5.6.5 Assert expression

The program is aborted with a runtime error if the expression is logical "false"

**Note:** *This expression protects against "impossible" or invalid conditions. In the following example, a negative fibonacci number is invalid. The assert statement marks this error as a programming error. Assert statements should only ever highlight programmer errors and never user inputs.*

Example:

```
fibonacci(n)
{
    assert n > 0

    new a = 0, b = 1
    for (new i = 2; i < n; i++)
    {
        new c = a + b
        a = b
        b = c
    }
    return a + b
}
```

### 13.5.6.6 Break

Terminates and leaves the smallest, encircling "do", "for" or "while" statement at any point in the loop. The "break" statement moves the program flow to the next statement outside the loop.

Example:

```
example(n)
{
    new a = 0

    for(new i = 0; i < n ; i++ )
    {
        a += i

        if(i>10)
            break

        a += 1
    }
    return a
}
```

### 13.5.6.7 Continue

Terminates the current iteration of the smallest encircling "do", "for" or "while" statement and moves the program control to the conditional part of the loop.

Example

```
example(n)
{
    new a = 0

    for(new i = 0; i < n ; i++ )
    {
        a += i

        if(i>10)
            continue

        a += 1
    }
    return a
}
```

---

### 13.5.6.8 Do statement while (expression)

Executes a statement before the conditional part (the "while" condition) is evaluated. The statement is repeated as long as the condition is logical "true". The statement is executed at least once.

Example:

```
example (n)
{
    new a = 0

    do
    {
        a++
    }
    while (n >= 0)

    return a
}
```

### 13.5.6.9 Exit expression

Cancels the program. The expression is optional, however, if present it must start and end on the same line as the "exit" statement. The exit statement returns the expression value or zero to the main application, if no expression is specified.

### 13.5.6.10 For (expression 1; expression 2; expression 3) statement

All three of the expressions are optional.

#### Expression 1:

*Is only evaluated once before entering the loop. This expression can be used to initiate a variable. This expression also includes the variable declaration by means of the "new" syntax. A variable that is declared at this stage is only valid in the loop. It is not possible to combine an expression (with existing variables) and a declaration of new variables in this field. All of the variables must either already exist in this field, or they must all be declared in this area.*

#### Expression 2:

*This expression is executed before every run of the loop and terminates the loop if the expression logical "false" is returned. If this expression is omitted, it is assumed that the result of expression 2 is logical "true".*

#### Expression 3:

*This expression is executed each time the statement is completed. The program control moves from expression 3 to expression 2 for the next (conditional) iteration of the loop.*

Example:

```
example(n)
{
    new a = 0

    for(new i = 0; i < n; i++)
    {
        a++
    }

    return a
}
```

The "for (; ;)" statement is the same as the "while (true)" statement.

#### 13.5.6.11 Goto label

Moves the program control (unconditionally) to the statement that follows the specified label. The label must be within the same function as the "goto"-statement (a "goto"-statement cannot jump out of a function).

#### 13.5.6.12 If (expression) statement 1 else statement 2

Executes statement 1 if the results of the expression is logical "true". The "else" clause of the "if" statement is optional. If the result of the expression is logical "false" and there is an "else" clause, the statement that is associated with the "else" clause (statement 2) is executed.

Example:

```
example(n)
{
    if(n < 0)
        return -1
    else if (n == 0)
        return 0
    else
        return 1
}
```

#### 13.5.6.13 Return expression

Terminates the current function and moves the program control to the next statement following the function call. The expression value is returned as the function result. The expression can be an array or a character string. The expression is optional, however, if present it must start on the same line as the "return" statement. Zero is returned if no expression is specified.

#### 13.5.6.14 switch (expression) {case list}

Transfers the sequence control to the various statements within the "switch", depending on the value of the "switch" expression. The main part of the "switch" statement is a composite statement that comprises a series of "case" clauses. Each "case" clause starts with the keyword "case" followed by a list of constants and a statement. The list of constants is a series of expressions separated by commas, each of which is evaluated as a constant value. This list ends with a colon. To specify an area in this list, separate the lower and upper limit of the area with a double point (".."). An example for an area is: "case 1..9:".

---

The "switch" statement shifts the sequence control to a "case" clause if a value from the list corresponds to the value of the "switch" expression.

The "default" clause consists of the "default" keyword and a double point. The "default" clause is optional, however, if it is specified it must be included as the last entry in the "case" list. The "switch" statement shifts the sequence control to the "default" clause if none of the "case" clauses comply with the "switch" expression.

Example:

```
example(n)
{
    new a = 0

    switch (n)
    {
        case 0..3:
            a = 0
        case 4,6,8,10:
            a = 1
        case 5,7:
            a = 2
        case 9:
            a = 3
        default:
            a = -1
    }

    return a
}
```

### 13.5.6.15 While (expression) statement

Evaluates the expression and executes the statement if the result of the expression is logical "true". The program control returns to the expression again once the statement has been executed. The statement is therefore executed as long as the expression is logical "true".

Example:

```
example(n)
{
    new a = 0

    while(n >= 0)
    {
        a++
    }

    return a
}
```



### 13.5.7 Functions

A function declaration specifies the name of the function and the formal parameters enclosed in brackets. A function can also return a value. A function must be defined globally, i.e. declared outside of another function and is globally available.

If the function declaration is followed by a semicolon (instead of a statement), this is a forward declaration of a function.

The "return" statement sets the return value of the function. For example, the return value of the "sum" function (see below) is the sum of both parameters. The "return" expression is optional.

```
sum(a, b)
{
    return a + b
}
```

The arguments of a function are (declared implicitly) local variables for this function. The function call specifies the values of the arguments. Another example of a complete definition of a function is "leap year" that indicates "true" or "false" for the relevant year.

```
leapyear(y)
{
    return y % 4 == 0 && y % 100 != 0 || y % 400 == 0
}
```

Details of the statements used in this example are provided in the chapter "Operators and expressions" on page 191.

Generally, functions include local variable declarations and consist of a block statement.

**Note:** In the next example, the "assert" statement prevents negative values for the exponent.

```
power(x, y)
{
    /* returns xy */
    assert y >= 0

    new r = 1
    for (new i = 0; i < y; i++)
        r *= x

    return r
}
```

A function can comprise several "return" statements, for example, one is used to quickly terminate a function if invalid parameters are transferred, or when it becomes apparent that the function has nothing to do. If a function returns an array, all of the "return" statements must return an array with the same number of entries.

#### 13.5.7.1 Function arguments ("call-by-value" versus "call-by-reference")

The "faulty" function in the next example has a parameter that is used in the loop to calculate the factorial of this number. It must be noted that the function modifies the argument.

---

```

main()
{
    new v = 5
    new f = faculty(v)
}

faculty(n)
{
    assert n >= 0

    new result = 1
    while (n > 0)
        result *= n--

    return result
}

```

Regardless of what (positive) value the "n" variable has at the start of the "while" loop, "n" will equal zero at the end of the function. In the "faculty" function, for example, the parameter is transferred as a value ("by value"), which means that changes to the "n" variable are only valid locally in the "faculty" function. In other words, the "v" variable in the "main()" function has the same value before and after the function is called up.

Arguments can be transferred as a value ("by value") or as a reference ("by reference"). A function argument that is to be transferred as a reference must have the "&" prefix preceding the name. The arguments are transferred to the function as a value by default.

Example:

```

swap(&a, &b)
{
    new temp = b
    b = a
    a = temp
}

```

To transfer an array to a function, add a pair of brackets ("[]") to the name of the argument. The number of entries can also be specified. This improves the error detection of the compiler's parser.

Example:

```

addvector(a[], const b[], size)
{
    for (new i = 0; i < size; i++)
        a[i] += b[i]
}

```

Arrays are always transferred as a reference.

**Note:** The "b" array in the above-mentioned example is not changed in the function. This function argument was declared as a "const" to make this explicit. In addition to the improved error detection, it also enables the compiler to generate a more efficient code.

The following code example calls up the "addvector" function and adds five to each element of the "vect" variables:

```
new vect[3] = [ 1, 2, 3 ]

addvector(vect, [5, 5, 5], 3)

/* vect[] now comprises the values 6, 7 and 8 */
```

### 13.5.7.2 Named parameters versus fixed parameters

In the previous examples, the order of the parameters in a function call were important as each parameter was copied to the same position of the function parameter. For example, in the "weekday" function (defined below), the expression "weekday(12,31, 1999)" would be used to get the weekday of the last day of the last century.

```
weekday(month, day, year)
{
  /* returns the day of the week: 0=Saturday, 1=Sunday, 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
}
```

The date format changes depending on the culture and country, while the USA use the month/day/year format, European countries frequently use the day/month/year format and technical publications use the year/month/day (ISO/IEC 8824) format. In other words, the sequence of the parameters is not "standardised" or "normal". For this reason, there is an alternative way of transferring parameters to a function, by using "named parameters". These are illustrated in the next example (the function was declared in the same way as the previous example).

```
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)
```

In "named parameters", a dot (".") precedes the name of the argument. The argument of the function can be set to any expression that is valid for the argument. In the event of a named parameter, the equals sign ("=") does not refer to an allocation but instead links the expression with a function argument.

Fixed and named parameters can be mixed together, although the fixed parameters must be specified before the named parameters.

### 13.5.7.3 Standard values of function arguments

A function argument can have a standard value. The standard value of a function argument must be a constant. To specify a standard value, add an equals sign ("=") and the value to the name of the parameter.

The standard value is adopted if a placeholder is specified instead of a valid function parameter during a function call. The placeholder is the underscore character ("\_"). The argument placeholder is only valid for parameters with a standard value.

The right argument placeholders can be removed from the list of arguments.

---

For example, if the "increment" function is defined as follows:

```
increment(&value, incr=1)
{
    value += incr
}
```

The following function calls are all the same:

```
increment(a)
increment(a, _)
increment(a, 1)
```

Standard values for arguments that are transferred as a reference are helpful in making these parameters optional. For example, if the "divmod" function was written to return the quotient and the rest as a parameter.

```
divmod(a, b, &quotient=0, &remainder=0)
{
    quotient = a / b
    remainder = a % b
}
```

Based on the previous definition of the "divmod" function, the following function calls are all valid:

```
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
```

The next example adds the value of an array to another one. The values of the array are increased by one if only one parameter is specified:

```
addvector(a[], const b[] = {1, 1, 1}, size = 3)
{
    for (new i = 0; i < size; i++)
        a[i] += b[i]
}
```

## 13.6 Differences to C

- The input mechanism of C is not present. It is an "integer-only" variant of C. There are no structures or unions. Floating point support must be implemented with user-defined operators and the help of native functions.
- The syntax for floating point values is stricter than that in C. In contrast to C, values such as ".5" and "6." are not accepted. It is mandatory to write "0.5" and "6.0". The decimal point is optional in C. If an exponent is included, then you can write "2E8" in C. The capital letter "E" is not accepted. Use the lower case letter "e". In addition, it requires a comma: e.g. "2.0e8" (see "Numerical constants" on page 187).

- "pointers" are not supported. A "reference" argument to transfer function parameters as a reference (see "Function arguments ("call-by-value" versus "call-by-reference")" on page 201) is included.. The "placeholder" argument replaces some applications of the ZERO pointer (see "Standard values of function arguments" on page 203).
- Numbers can be specified in a hexadecimal, decimal or binary format. The octal format is not supported (see "Numerical constants" on page 187). Hexadecimal numbers must start with "0x" ("x" in lower case). The prefix "0X" is invalid.
- "Cases" in a "switch"-statement are not "fall through". At least one statement must follow the "case" label. You must create a composite statement (with { }) to execute several statements (see "switch (expression) {case list}" on page 199). The "switch" statement should be considered as a structured "if". However, in C/C++ the "switch" statement is a "conditional goto".
- A "break" statement only terminates loops. In C/C++, the "break" statement also terminates a "case" in a "switch" statement.
- "array assignments" are supported with the limitation that both of the arrays must be the same length. For example, if "a" and "b" arrays have six lines, the expression "a=b" is valid. In addition to character strings, literal arrays and thus expressions such as "a = {0,1,2,3,4,5}" are also supported where "a" is an array variable with six elements.
- "defined" is an operator and not a preprocessor directive. The "defined" operator works with constants (declared with "const"), global variables, local variables and functions.
- The "sizeof" operator returns the size of the variables in "elements" and not in "bytes". An element is an entry or sub-array. Further details are provided in the chapter "Other" on page 194.
- An empty statement is an empty block (with { }) and not a semicolon (see "Composite statements" on page 196). This change prevents frequent errors.
- A division is completed in such a way that the remainder of the division has (or ought to have) the same prefix as the denominator. Divisions (operator "/") are always rounded down to the smaller whole number (whereby -2 is smaller than -1). For example,  $5/2 = 2$  (2.5 is rounded down to 2),  $-5/2 = -3$  (-2.5 is rounded down to -3). The "%" operator always generates a positive result regardless of the prefix of the numerator (see "Operators and expressions" on page 191).
- There is no unary "+" operator as it is a "no-operation" operator anyway ("a = +1" is not valid; correct: "a = 1").
- Three bit by bit operators have different priorities than in C. The priority level of the "&", "^" and "|" operator is higher than the relational operators. Dennis Ritchie explains that these operators were assigned a low priority level in C as early C compilers did not yet include the logical "&&" and "||" operators so that bit by bit "&" and "|" were used instead.
- The keyword "const" implements the "enum" functionality of C.

- 
- In most cases, the forward declarations of functions (i.e. prototypes) are not necessary as a two-pass compiler is used. It detects all of the functions during the first cycle and uses them during the second. User-defined operators must however be declared before use. If available, forward declarations must be exactly the same as the definition of the function. The parameter names in the prototypes and the definitions of the functions must be identical. Due to the "named parameter" function, the parameter names in the prototype are of significance. Prototypes are used to call up the forward declared functions. To use these with the named parameters during this process, the compiler must already know the names of the parameters (and their position in the parameter list). The parameter names in the prototypes must therefore match those in the definitions.
  - Variables are automatically initialised using „0“. It is therefore not necessary to explicitly set them to „0“.

---

# Chapter 14 Data Descriptor

The basic principle of the myDatalogC33x is "storage-2-storage" data transmission. For this type of data transmission, neither the myDatalogC33x nor the server must know about the logical content of the data blocks. Therefore, the only aim is to transport a block of data from A to B.

The data transferred from the myDatalogC33x to the sever can therefore be selected freely. There are 1023 Byte available per data record that can be used as required. There are also another 10 independent memory blocks each with 4000 Byte for the configuration data that can be used as required.

The content of the data block or configuration block must be described on the server so that the data and configurations received from the myDatalogC33x can also be used within the myDatanet interface (reports, visualisations, graphics, etc.). The Data Descriptor contains the tool for describing the data as well as the correct provision of the data for use within the interface.

## 14.1 Data structure

The following containers are available for the different types of data (measurement data, configurations, etc.):

- **Measurement data:** "#histdata0" - "#histdata9"
- **Configurations:** "#config0" - "#config9"
- **Configurations only available on the server:** "#configA" - "#configC"
- **Alarm messages:** #alerts
- **Aloha data:** #aloha

This section describes how the structured measurement data channels ("#histdata0" - "#histdata9"), configuration memory blocks ("#config0" - "#config9") and the aloha data ("aloha") are split into individual data fields for use on the myDatanet server. The structure of the alarm messages ("#alerts") is fixed firmly in the system and does not need to be specified/cannot be changed by the user.

**Important note:** *If a structured measurement data channel, a configuration block or the aloha data are to be available on the myDatanet server or via the REST API then all data fields need to be defined by means of the Data Descriptor .*

An extended example, in which most of the available attributes are used, is provided in chapter "Example" on page 215.

---

### 14.1.1 Division of a structured measurement data channel into individual data fields

```
#histdata0 Measurements up
BatVoltage      s16  title="Battery Voltage"  decpl=2  units="V"   vscale=0.001
InputVoltage    s16  title="USB Voltage"    decpl=2  units="V"   vscale=0.001
```

The first line in the example above specifies the container to be used for the measurement data:

- #histdata0:** The measurement data should be stored in histdata channel 0.
- Measurements:** "Measurements" should be used as the name for the histdata channel.
- up:** The data is only transmitted from the device to the server.

*Note: After specifying the direction of transmission other attributes (e.g. "title") could be added.*

The second line in the example above describes the first measurement value in the measurement data container used:

- BatVoltage:** "BatVoltage" should be used as the name for the measurement value.
- s16:** The data type used for the measurement value should be a 16-bit signed integer.
- title:** Name of the measurement value that is displayed on the server
- decpl:** Number of decimal places that should be displayed
- units:** Unit of the measurement value that is displayed on the server
- vscale:** Virtual scaling of the value (see "Attributes of the field definition" on page 210)

*Note: Name and data type must always be specified. Attributes are optional. Further attributes can also be added.*

The third line in the example above describes the second measurement value in the measurement data container used.



---

## 14.1.2 Division of a configuration memory block into individual data fields

```
#config0 BasicCfg down title="Basic configuration"  
RecordItv      u32  title="Record Interval"      units="sec"  min=10  default=10  
TransmissionItv u32  title="Transmission Interval"  units="min"  min=10  default=60
```

The first line in the example above specifies the container to be used for the configuration:

**#config0:** The parameters should be stored in configuration memory block 0.  
**BasicCfg:** "BasicCfg" should be used as the name for the configuration memory block.  
**down:** The configuration memory block is only transmitted from the server to the device.  
**title:** Name of the configuration section that is displayed on the server

***Note:** Name and direction of transmission must always be specified. Attributes are optional. Further attributes (e.g. "edit" or "view") can also be added.*

The second line in the example above describes the first parameter in the configuration memory block:

**RecordItv:** "RecordItv" should be used as the name for the parameter  
**u32:** The data type used for the parameter should be a 32-bit signed integer.  
**title:** Name of the parameter that is displayed on the server  
**units:** Unit of the parameter that is displayed on the server  
**min:** smallest valid value for the parameter  
**default:** Default value for the parameter

***Note:** Name and data type must always be specified. Attributes are optional. Further attributes (e.g. "vscale") can also be added.*

The third line in the example above describes the second parameter in the configuration memory block.

---

### 14.1.3 Division of the aloha data into individual data fields

```
#aloha up
BatVoltage      s16  title="Battery Voltage"  decpl=2  units="V"  vscale=0.001
InputVoltage    s16  title="USB Voltage"          decpl=2  units="V"  vscale=0.001
```

The first line in the example above specifies the container to be used for the aloha data:

**#aloha:** The measurement data should be stored in the aloha data container.  
**up:** The data is only transmitted from the device to the server.

*Note: After specifying the direction of transmission other attributes (e.g. "title") could be added.*

The second line in the example above describes the first measurement value in the aloha data container used:

**BatVoltage:** "BatVoltage" should be used as the name for the measurement value.  
**s16:** The data type used for the measurement value should be a 16-bit signed integer.  
**title:** Name of the measurement value that is displayed on the server  
**decpl:** Number of decimal places that should be displayed  
**units:** Unit of the measurement value that is displayed on the server  
**vscale:** Virtual scaling of the value (see "Attributes of the field definition" on page 210).

*Note: Name and data type must always be specified. Attributes are optional. Further attributes can also be added.*

The third line in the example above describes the second measurement value in the aloha data container used.

### 14.1.4 Attributes of the field definition

#### title

*Alpha-numeric. Title of the field. This title is then used in all of the reports, graphics, etc. for this channel. The length of the title should not exceed 16 characters if possible, otherwise this can cause display problems on the interface.*

#### units

*Alpha-numeric. Units of a value. The length of the units should not exceed eight characters if possible, otherwise this can cause display problems on the interface.*

#### bitmask

*Hexadecimal, without leading "0x". Bit mask to mask the actual bits to be used out of the data block, hexadecimal. The extracted value is aligned to the LSB following the masking process. Example: An u16 field is generated out of two bytes with the 0xF3A7 content. 0FF0 is specified as the bit mask. The bits are extracted and aligned with the LSB. The subsequent HEX value is therefore 0x3A (=58 decimal).*

---

**editmask**

Format statements for displaying the field content on the interface of the myDatenet server or input via the interface of the myDatenet server.

- usable for strings (data type "astr", "nstr", "wstr" and "ustr", however not for "cstr")

<b>Format statements</b>	<b>Explanation</b>
<b>"%COLORPICKER%"</b>	Creates a button that displays the currently selected colour. When clicking the button, a dialog field to set the desired colour appears. The dialog field returns the selected colour as a string in "RRGGBB" format, with the colour components being specified in hex.
<b>"%HEX%"</b>	Creates an input field in which the string is displayed in hex format. Each byte of the string is therefore represented by two characters. For example, the letter 'a' is represented as "61".
<b>"%MASKED%"</b>	Creates an input field whose content is masked with "asterisk" (e.g. for passwords and tokens). A button (eye icon) for switching between plain text and masking is displayed next to the input field.
<b>"%MULTILINE%30%75"</b>	Creates a text field with 30 lines and 75 characters per line

- usable for numeric fields (data type "u8", "s8", "f32", ....)

<b>Format statements</b>	<b>Explanation</b>
"%2.x0"	<p>Creates an input field in which the numeric value is displayed in hex format. Each byte of the data type is therefore represented by two characters. The number after "%" must match the number of characters required to represent the data type (e.g. '4' for a "u16").</p> <p><b>Note:</b> When using this format statement, the "decpl" attribute must also be set to "-1" (i.e. decpl=-1).</p>
"0=off;1=on"	<p>A dropdown list is created in which the text following the "=" is displayed for each entry instead of the value. Entries must be separated using a ";".</p> <p><b>Note:</b> The entries <b>MUST NOT</b> start with '+' or '~'</p>
"%CHECKBOX%"	<p>A checkbox is created.</p> <p>1 = tick set not equal to 1 = tick not set</p> <p><b>Note:</b> If the checkbox is used to display data and the device returns a value not equal to 1, the tick is also displayed as "not set". If the checkbox is used for data entry and the tick is not set, the checkbox returns 0.</p>
"%CHECKBOX%5;10"	<p>A checkbox is created the same as for editmask="%CHECKBOX%", except that the values for "tick set" (in this example 10) and "tick not set" (in this example 5 or not equal to 10) can be chosen. The note above is also valid in a similar way.</p>
"%TIME%s%hh:mm:ss"	<p>Creates an input field in which the numeric value is displayed in the specified time format (e.g. hh:mm:ss)</p> <p>After the segment "%TIME" it must be specified whether the value is saved in seconds (%s) or in minutes (%m). The time format follows after this specification. "%hh:mm:ss", "%hh:mm" or "%mm:ss" are possible formats. Caution "%mm:ss" is not valid if the value is to be saved in minutes (%m).</p> <p><b>Note:</b> It is advisable to use the "units" attribute to specify the time format (e.g. hh:mm:ss) in which the value is displayed.</p>

- usable for timestamps (data type "stamp32" and "stamp40")

<b>Format statements</b>	<b>Explanation</b>
<code>%DATETIMEPICKER%</code>	Creates an input field in which the timestamp is displayed as date/time. When clicking on the input field, a dialog field to select date and time appears.

### **decpl**

*Numeral, integer positive. Number of decimal places that should be displayed.*

### **vscale**

*Numeral, floating point. Virtual scaling of the value. The extracted value is multiplied by this factor and only then can it continue to be used.*

*This value represents value k from the formula  $k*x+d$ .*

### **vofs**

*Numeral, floating point. Virtual offset of the value. The extracted value is multiplied by vscale, and vofs is then added to the value.*

*This value represents value d from the formula  $k*x+d$ .*

### **min**

*The minimum value for the subsequent display on the server (e.g. graphic).*

### **max**

*The maximum value for the subsequent display on the server (e.g. graphic). This value is used for the length of the character string in the string (or char) data type. This means that the specification of the max. value for the string (or char) is mandatory.*

### **chmode**

*The channel mode. This selected setting affects any further processing and display of the channel in the individual server modules.*

*The following channel modes are available:*

<b>Mode</b>	<b>Name</b>	<b>Explanation</b>
1	Digital	The channel is processed as a digital channel. To prevent any problems from occurring, the included value should be 0 or 1.
2	Day counter	A counter for which the value is reset once daily. This reset must be completed by the control program.
3	Interval meter	A meter that is reset every time a measurement data record is created. This reset must be completed by the control program.
6 (standard)	Analogue	A "simple" measurement value, e.g. the temperature
12	Infinite meter	A meter for which the value is never reset, e.g. water or electricity meter

### **index**

*Is used for the user-defined sorting of the fields in the selection lists. The standard value for the 1000 channels that are available is -1, which ensures that the channel is hidden.*

*As soon as a channel is used in the data structure description (a simple field tag with the attribute name will suffice), the background index value is automatically set to the field index (e.g. index=2 for ch2).*

*This standard sorting can be overridden by specifying the index field.*

---

**view**

*Numeral, integer positive. Specifies from which user level, the field is visible on the interface of the myDatenet server.*

**edit**

*Numeral, integer positive. Specifies the user level that is required to be able to change the field content via the interface of the myDatenet server. If this attribute is not specified or if the specified value is lower than that of the "view" attribute, the user level specified for the "view" attribute is required to change the field content.*

*If changing the field content via the REST-API should be prevented, the value for this attribute must be set to 99.*

---

## 14.2 Example

```
#histdata0 Measurements up
Delay u16 title="Delay" units="sec" min=10 max=2000 vofs=10 chmode=3 index=1
Height f32 title="Height" decpl=2 units="cm" min=0 max=2000 vscale=0.01 chmode=6 index=0
Pump u8 view=99 edit=99
@Pump
Pump_MSK u8 dlorw=skip title="Pump" bitmask=$01 min=0 max=1 chmode=1
Info astr.50 title="Info" index=10
```

The first line specifies the container to be used for the measurement data:

- #histdata0:** The measurement data should be stored in histdata channel 0.
- Measurements:** "Measurements" should be used as the name for the histdata channel
- up:** The data is only transmitted from the device to the server

The second line describes the first measurement value "Delay" in the measurement data container used:

- Delay:** "Delay" should be used as the name for the measurement value.
- u16:** The data type used for the measurement value should be a 16-bit unsigned integer (i.e. 2 bytes).
- title:** Name of the measurement value that is displayed on the server
- units:** Unit of the measurement value that is displayed on the server
- min:** Minimum value for the further display on the server (e.g. graphic)
- max:** Maximum value for the further display on the server (e.g. graphic)
- vofs:** Virtual offset of the value (see "Attributes of the field definition" on page 210). In the current example 10 is added to the extracted value.
- chmode:** Channel mode  
3 = ^ interval counter (counter that is reset every time a measurement data record is created)
- index:** Is used for the user-defined sorting of the fields in the selection lists

The third line describes the second measurement value "Height" in the measurement data container used:

- Height:** "Height" should be used as the name for the measurement value.
- f32:** The data type used for the measurement value should be a 32-bit float (i.e. 4 bytes).
- title:** Name of the measurement value that is displayed on the server
- decpl:** Number of decimal places that should be displayed
- units:** Unit of the measurement value that is displayed on the server
- min:** Minimum value for the further display on the server (e.g. graphic)

---

<b>max:</b>	Maximum value for the further display on the server (e.g. graphic)
<b>vscale:</b>	Virtual scaling of the value (see "Attributes of the field definition" on page 210). In the current example the extracted value is multiplied by 0.01.
<b>chmode:</b>	Channel mode 6 =^ analogue channel (a "simple" measurement value, e.g. the temperature)
<b>index:</b>	Is used for the user-defined sorting of the fields in the selection lists

Lines 4-6 describe the third measurement value "Pump" in the measurement data container used:

<b>Pump:</b>	"Pump" should be used as the name for the measurement value.
<b>u8:</b>	The data type used for the measurement value should be an 8-bit unsigned integer (i.e. 1 byte).
<b>view:</b>	Specifies from which user level the field is visible on the interface of the server 99 =^ field is not visible to anyone (required as in the current example the "Pump_MS" shadow field should be used instead of the "Pump" field. Shadow fields can be used to divide one byte field up into several bit fields).
<b>edit:</b>	Specifies the user level that is required to be able to change the field content via the interface of the server 99 =^ field cannot be changed by anyone (required as in the current example the "Pump_MS" shadow field should be used instead of the "Pump" field).
<b>@Pump:</b>	Specifies that the "Pump_MS" shadow field defined in line 6 should use the memory area of the "Pump" field defined in line 4
<b>Pump_MS:</b>	"Pump_MS" should be used as the name for the shadow field.
<b>dlorw=skip:</b>	The shadow field is not considered for the access functions to the container which are automatically generated for the device logic (in the current example histdata channel 0). I.e. within the device logic writing/reading of the shadow field must be done via manual masking of the desired bit in the "Pump" field.
<b>bitmask:</b>	Bit mask to mask the actual bits to be used out of the data block, hexadecimal  In the current example, the least significant bit (LSB) is masked out of the "Pump" field.
<b>min:</b>	Minimum value for the further display on the server (e.g. graphic)
<b>max:</b>	Maximum value for the further display on the server (e.g. graphic)
<b>chmode:</b>	Channel mode. 1 =^ digital



The 7th line describes the fourth measurement value "Info" in the measurement data container used:

- Info:** "Info" should be used as the name for the measurement value.
- astr.50:** The data type used for the measurement value should be an ANSI string. After the dot the number of characters (50 in the current example) are specified.
- title:** Name of the measurement value that is displayed on the server
- index:** Is used for the user-defined sorting of the fields in the selection lists.

index is not specified for "Pump", thus "Pump" automatically receives index 2. The sorting order of the channels is therefore "Height", "Delay", "Pump", "Info".

### 14.3 Special values of the data types

Each numerical data type supports special states, such as NAN (Not a Number). If such a value is detected on the server, the standard display and further processing in myDatenet is applied.

#### Overview of the possible values (unsigned):

Value/type	u8 (byte)	u16 (word)	u32 (dword)
NaN	0xFF	0xFFFF	0xFFFFFFFF
OF	0xFE	0xFFFE	0xFFFFFFFFE
UF	0xFD	0xFFFD	0xFFFFFFFFD
OL	0xFC	0xFFFC	0xFFFFFFFFC
SC	0xFB	0xFFFB	0xFFFFFFFFB

#### Overview of the possible values (signed):

Value/type	s8 (bint)	s16(wint)	s32 (dint)	s64 (qint)
NaN	127	32767	2147483647	0x7FFFFFFFFFFFFFFF
OF	126	32766	2147483646	0x7FFFFFFFFFFFFFFFE
UF	-126	-32766	-2147483646	0x8000000000000002
OL	-127	-32767	-2147483647	0x8000000000000001
SC	-128	-32768	-2147483648	0x8000000000000000

#### Overview of the possible values (float):

Value/type	f32 (float32)	f64 (float64)
NaN	0x7F800001	0x7FF0000000000001
OF	0x7F800002	0x7FF0000000000002
UF	0x7F800003	0x7FF0000000000003
OL	0x7F800004	0x7FF0000000000004
SC	0x7F800005	0x7FF0000000000005



# Chapter 15 API

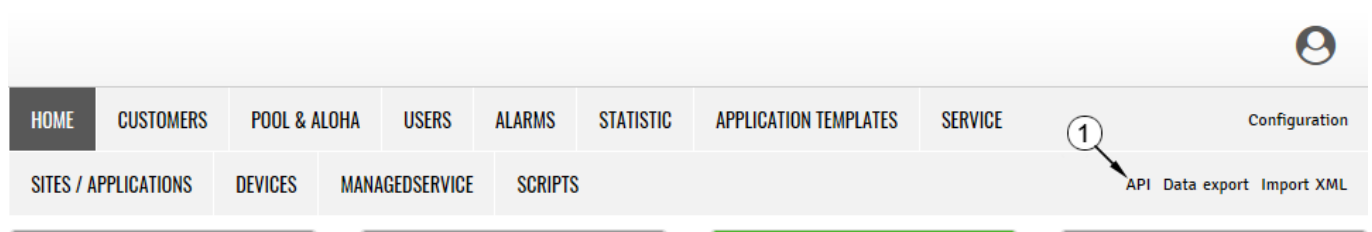
**Important note:** The relevant licences are required on the myDatanet server to use the API (Application Programming Interface). For future information contact your responsible sales partner.

## 15.1 Backend API

The API is provided to export data from and import data to the myDatanet server. However, this is not just limited to the pure measurement data but includes all of the data provided by myDatanet server (e.g. configurations). It is therefore possible for the customer to completely dispense with the interface of the myDatanet server and to create his own user interface. A specially developed PC program or web interface can, for example, be used for this purpose.

## 15.2 rapidM2M Playground

The rapidM2M Playground enables you to familiarise yourself with the API of the myDatanet server and to test the provided functions. One click on the "API" button will take you to rapidM2M Playground .



1 Opens the rapidM2M Playground

## 15.2.1 Overview

### rapidM2M Playground

1	Input field for the user name
2	Input field for the password
3	List of the available HTTP commands. The HTTP commands are grouped according to their fields of application.
4	Depending on the selected HTTP command, the drop down lists for selecting the customer, user and site that should replace the corresponding wild cards (" \$CID "...customer , "\$UID"...user, "\$SID"...site) in the resource path of the HTTP command are displayed.
5	Button to execute the HTTP command
6	Opens the website "http://rapidm2m.com/" that includes additional information for developers
7	Opens the quick guide for the API
8	Button for displaying the menu that contains the global settings
9	Button to change the colour scheme of the rapidM2M Playground
10	Window displaying the selected HTTP command
11	Response code sent by the myDatanet server as an answer to the HTTP command
12	Copies the JSON object generated as a response to the HTTP command on to the clipboard
13	Window displaying the documentation for the selected HTTP command. Depending on the selected command, this includes a description of the action being executed, information that must be observed and a description of the request body and response body.
14	Window displaying the JSON object that is generated as a response to the HTTP command
15	Window displaying the last executed HTTP commands

# Chapter 16 Maintenance

**Important note:** To prevent any damage to the device, the work described in this section of the instructions must only be performed by qualified personnel.

*The device must be deenergised before any maintenance, cleaning and/or repair work.*

## 16.1 General maintenance

- Regularly check the myDatalogC33x for mechanical damage.
- Check all cables for mechanical damage at regular intervals.
- Clean the myDatalogC33x with a soft, moist cloth. Use a mild cleaning agent, if necessary.

## 16.2 Fuse replacement



**DANGER:**

***Risk of fire. An incorrect fuse can cause injuries, damages or emissions. The fuse is located inside the housing. The housing may only be opened by the manufacturer.***

If you believe that the fuse of the myDatalogC33x is defective (see "Troubleshooting and repair" on page 225), the device must be sent back in its original packaging to the manufacturer (see "Return" on page 42).



# Chapter 17 Removal/disposal

## Incorrect disposal can cause environmental hazards.

Dispose of the device components and packaging material in accordance with the locally valid environmental regulations for electronic products.

1. Disconnect any charging voltage that has been used.
2. Disconnect any connected cables using a suitable tool.



### Logo of the EU WEEE Directive

This symbol indicates that the requirements of Directive 2012/19/EU regarding the scrap disposal of waste from electric and electronic equipment must be observed. Microtronics Engineering GmbH supports and promotes recycling and environmentally friendly, separate collection/disposal of waste from electric and electronic equipment in order to protect the environment and human health. Observe the local laws and regulations on disposal of electronic waste at all times.

Microtronics Engineering GmbH releases goods brought onto the market in Austria from the obligations via ERA, which means that collection points that cooperate with ERA Elektro Recycling Austria GmbH (<https://www.era-gmbh.at/>) can be used for disposal in Austria.

**The device contains a lithium button cell that has been soldered on. It must be removed before disposal or the disposal service must be informed that batteries are still located in the device.**

**The device includes a battery or rechargeable battery (lithium) that must be disposed of separately.**





# Chapter 18 Troubleshooting and repair

## 18.1 General problems

Problem	Cause/solution
The device does not respond	<ul style="list-style-type: none"> <li>• Check the cable connections (see "Connecting the sensors, actuators and power supply" on page 46)</li> </ul>
Communication problems	<ul style="list-style-type: none"> <li>• Load the device log from the myDatalogC33x using the DeviceConfig (see "myDatanetDeviceConfig Manual " 805004).</li> </ul>
Not all or no data is available on the server.	<ul style="list-style-type: none"> <li>• The connection was aborted during the transmission, which is indicated by a time-out entry in the connection list (see "myDatanet Server Manual " 805002). Solution: Initiate a transmission or wait for the next cyclical transfer.</li> <li>• The Data Descriptor was not configured correctly (see "Data Descriptor " on page 207).</li> <li>• The assignment of the device and site is not correct (see "Site" on page 74).</li> </ul>
Data at universal input is not plausible	<ul style="list-style-type: none"> <li>• Check the cable connections (see "Connecting the sensors, actuators and power supply" on page 46)</li> <li>• Check whether the output signal of the sensor that you are using is compatible with the electrical characteristics of the universal inputs (see "Technical details about the universal inputs" on page 59).</li> <li>• The Data Descriptor was not configured correctly (see "Data Descriptor " on page 207).</li> </ul>
The data of the RS485 interface is not plausible.	<ul style="list-style-type: none"> <li>• Check the cable connections (see "Connecting the sensors, actuators and power supply" on page 46)</li> <li>• Check whether the sensor that you are using is compatible with the electrical characteristics of the interface (see "Technical details about the RS485 interface" on page 60).</li> <li>• The Data Descriptor was not configured correctly (see "Data Descriptor " on page 207).</li> </ul>
The data received via the CAN interface is not plausible	<ul style="list-style-type: none"> <li>• Check the cable connections (see "Connecting the sensors, actuators and power supply" on page 46)</li> <li>• Check whether the machines, sensors and actuators you are using are compatible with the characteristics of the CAN interface (see "Technical details about the CAN interface" on page 61).</li> <li>• The Data Descriptor was not configured correctly (see "Data Descriptor " on page 207).</li> </ul>
The data of the RS232 interface is not plausible.	<ul style="list-style-type: none"> <li>• Check the cable connections (see "Connecting the sensors, actuators and power supply" on page 46)</li> <li>• Check whether the sensor that you are using is compatible with the electrical characteristics of the interface (see "Technical details about the RS232 interface" on page 63).</li> <li>• The Data Descriptor was not configured correctly (see "Data Descriptor " on page 207).</li> </ul>
The isolated switch contact is not working.	<ul style="list-style-type: none"> <li>• Disruption to the voltage that is conducted via the isolated switch contact</li> </ul>

<b>Problem</b>	<b>Cause/solution</b>
The device logic is not being executed correctly.	<ul style="list-style-type: none"> <li>• Check that the correct device logic type was selected during the configuration of the control (see "Control" on page 75).</li> <li>• Load the device log from the myDatalogC33x using the DeviceConfig (see "myDatagnetDeviceConfig Manual " 805004). A list of all the possible device logic error codes is included in the chapter "Pawn script error codes" (see "Device Logic error codes" on page 182).</li> <li>• The previous device logic has been replaced with that of the newly assigned site/application due to a context change (assignment of a different site/application).</li> <li>• By assigning a new or different site/application, the device logic installed via the USB has been replaced with that of the newly assigned site/application.</li> </ul>

## 18.2 Log entries and error codes

Log entry		Parameter		Description
Code	Plain text	Code	Plain text	
1000	POWER ON	0	---	Restart following a power failure
		4	---	Watchdog reset (e.g. because of an exception)
		6	---	Reset was initiated by the device itself (e.g. in event of firmware update)
		##	--	Restart for another reason. There may be a hardware problem if the "POWER ON" log entry with a parameter code that is not equal to 0 or 6 is contained in the device log several times. Contact the manufacturer in this case (see "Contact information" on page 247).
1030	UV LOCKOUT	---	---	The device switches to energy saving mode and terminates all of the operations as the rechargeable battery or battery voltage is too low. Only the charge controller, if present, remains active.
1031	UV RECOVER	---	---	The rechargeable battery or battery voltage once again suffices to guarantee reliable operation. The device resumes normal operation in accordance with the configuration.
1034	CONTROLLER UPDATE	##	---	Controller firmware update was completed successfully  This entry is always duplicated in the device log. In the first entry, the parameter specifies the major version number (e.g. 3 for 03v011), while in the second entry it specifies the minor version number (e.g. 11 for 03v011).
1035	EXCEPTION	##	---	An internal system error was detected that caused the device to restart. The parameter specifies the type of system error. Contact the manufacturer if the device log contains this error with the same parameter code several times (see "Contact information" on page 247).
1038	UV MODEM LOCKOUT	---	---	The device deactivates the modem because the rechargeable battery or battery voltage is too low. A connection cannot be established now.
1039	UV MODEM RECOVER	---	---	The rechargeable battery or battery voltage once again suffices to guarantee a stable connection.

Log entry		Parameter		Description
Code	Plain text	Code	Plain text	
1161	LOG REFORMATFILE	##	---	Errors in file system have been resolved. This can result in data being lost (data and/or log entries). The parameter contains more information on the problem. Contact the manufacturer if the device log contains this error with the same parameter code several times (see "Contact information" on page 247).
1192	FUTURE TIMESTAMP	##	---	Internal error Contact the manufacturer if the device log includes this error several times (see "Contact information" on page 247).
1200	MODEM ERROR			Modem error (see "Modem error" on page 232)
1201	MODEM NOT FOUND	---		Internal error Contact the manufacturer if the device log includes this error several times (see "Contact information" on page 247).
1202	MODEM CMME ERROR	##	---	The GPRS modem indicates a +CME error. The parameter specifies the type of error.
1203	SELECTED NETWORK	##	---	A new GSM network was selected. The parameter specifies the MCC (Mobile Country Code) and the MNC (Mobile Network Code) of the selected GSM network.
1207	GSM NETWORK REGISTRATION	0	NOT REGISTERED	Not registered, modem is currently not looking for any new operators to register
		1	HOME	Registered, home network
		2	SEARCHING	Not registered, but the modem is currently looking for a new operator with which it can register
		3	DENIED	Registration denied
		4	UNKNOWN	Unknown (e.g. outside the GERAN/UTRAN/E-UTRAN cover)
		5	ROAMING	Registered, roaming

Log entry		Parameter		Description
Code	Plain text	Code	Plain text	
1208	GPRS NETWORK REGISTRATION	0	NOT REGISTERED	Not registered, modem is currently not looking for any new operators to register
		1	HOME	Registered, home network
		2	SEARCHING	Not registered, but the modem is currently looking for a new operator with which it can register
		3	DENIED	Registration denied
		4	UNKNOWN	Unknown (e.g. outside the GERAN/UTRAN/E-UTRAN cover)
		5	ROAMING	Registered, roaming
1212	ERROR MODEM IRREGULAR OFF	##	---	Indicates a faulty connection. The parameter includes a counter that indicates how many consecutive connections have not worked.
1219	LTE NETWORK REGISTRATION	0	NOT REGISTERED	Not registered, modem is currently not looking for any new operators to register
		1	HOME	Registered, home network
		2	SEARCHING	Not registered, but the modem is currently looking for a new operator with which it can register
		3	DENIED	Registration denied
		4	UNKNOWN	Unknown (e.g. outside the GERAN/UTRAN/E-UTRAN cover)
		5	ROAMING	Registered, roaming
1252	MODEM TO CON	##	---	Timeout while a connection is being established. The parameter specifies the reason for the timeout. Contact the manufacturer if the device log contains this error with the same parameter code several times (see "Contact information" on page 247).
1281	ZLIB STREAMPROCESS ERR	##	---	Internal error  Contact the manufacturer if the device log includes this error several times (see "Contact information" on page 247).
1282	ZLIB STREAMFINISH ERR	##	---	Internal error  Contact the manufacturer if the device log includes this error several times (see "Contact information" on page 247).
1300	USB CONNECTED	---	---	USB connection to a PC established.
1310	USB DISCONNECTED	---	---	USB connection was disconnected.

Log entry		Parameter		Description
Code	Plain text	Code	Plain text	
1335	LOG_SHT2X_STATE	0	SHT2X SENSOR OK	The internal temperature and air humidity sensor is returning valid values again
		1	SHT2X RH ERROR	A communication error occurred when reading the air humidity value from the internal temperature and air humidity sensor.
		2	SHT2X TEMP ERROR	A communication error occurred when reading the temperature value from the internal temperature and air humidity sensor.
		3	SHT2X RH+TEMP ERROR	A communication error occurred when reading the measurement value from the internal temperature and air humidity sensor.
		4	SHT2X PLAUSIBILITY ERROR	The values received from the internal temperature and air humidity sensor are not plausible (rH <0% rH or >100% rH or temperature <-40°C or >125°C)
1336	SHT2X COM ERR	---	---	Communication with the internal temperature and air humidity sensor is not possible (sensor not present or faulty)
1337	SHT2X COM ERR1	---	---	Starting the internal temperature measurement failed
1338	SHT2X COM ERR2	---	---	Starting the internal air humidity measurement failed
1339	SHT2X TEMP RAW	##	---	Temperature raw value (register value from the internal temperature and air humidity sensor) if a plausibility error (SHT2X PLAUSIBILITY ERROR) was detected
1340	SHT2X RH RAW	##	---	Air humidity raw value (register value from the internal temperature and air humidity sensor) if a plausibility error (SHT2X PLAUSIBILITY ERROR) was detected
1601	SIM_STATE	0	NONE	SIM state was changed to "NONE" (initial state).
		1	PRODUCTION	SIM state was changed to "PRODUCTION" (a new device is in stock).
		2	HOT	SIM state was changed to "HOT" (valid contract).
		3	COLD	SIM state was changed to "COLD" (end of contract or fair use policy violated).
		4	DISCARDED	SIM state was changed to "DISCARDED" (device has been decommissioned).

Log entry		Parameter		Description
Code	Plain text	Code	Plain text	
1910	ACCU 0 E2PROM ERROR	0	---	Rechargeable battery not available
		1	---	Invalid length of the data structure in the EEPROM of the rechargeable battery
		2	---	No charging profile available in the EEPROM (only with Li-ion rechargeable batteries)
		3	---	Error when reading the SoC-value
		4	---	Error when writing the SoC-value
		5	---	The charging profiles of the rechargeable batteries inserted do not match (only with devices that support the simultaneous use of multiple rechargeable batteries)
		6	---	<ul style="list-style-type: none"> <li>• Permissible charging time exceeded</li> <li>• When restarting the device, it was recognised that the rechargeable battery currently in use has already exceeded the permissible charging time once.</li> </ul> <p>The battery is probably defective and should be checked by the manufacturer.</p>
2000 - 2199	MODULE ERR	##	---	Area for customer-specific critical error codes, that can be written in the device log by means of the "rM2M_WriteLog()" function
2200 - 2399	MODULE WARNING	##	---	Area for customer-specific non-critical error codes, that can be written in the device log by means of the "rM2M_WriteLog()" function
2400 - 2599	MODULE INFO	##	---	Area for customer-specific information about the current operating state, that can be written in the device log by means of the "rM2M_WriteLog()" function
2600 - 2799	MODULE DEBUG	##	---	Area for customer-specific debug information, that can be written in the device log by means of the "rM2M_WriteLog()" function
3000 - 3099	SCRIPT ERROR	##	--	Error codes of the script execution (see "Device Logic error codes" on page 182).

---

### 18.2.1 Modem error

Log entry		Parameter		Description
Code	Plain text	Code	Plain text	
GPRS error				
1200	BEARER GPRS FAILED	-988	---	GPRS setup error <ul style="list-style-type: none"><li>• Try to improve the position of the antenna.</li><li>• Check whether the device is in the coverage area (<a href="http://www.microtronics.com/footprint">www.microtronics.com/footprint</a>).</li></ul>
1200	BAND SEL FAILED	-969	---	A network could not be found on the GSM900/1800 or on the GSM850/1900 band. <ul style="list-style-type: none"><li>• Try to improve the position of the antenna.</li><li>• Check whether the device is in the coverage area (<a href="http://www.microtronics.com/footprint">www.microtronics.com/footprint</a>).</li></ul>

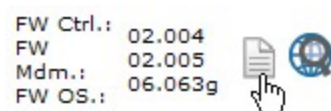


Log entry		Parameter		Description
Code	Plain text	Code	Plain text	
1200	NETLOCK ERROR	-966		Error when selecting the network. Check whether the device is in the coverage area.  Internal SIM chip: see <a href="http://www.microtronics.com/footprint">www.microtronics.com/footprint</a>
TCP channel error				
1200	CHANNEL ABORTED	-965	---	An attempt is being made to write to/read a TCP client that is no longer available.  Try again later
	TCP DNS FAILURE	-958	---	The name could not be resolved in an IP address.  Internal error
	CHANNEL REFUSED	-955	---	The TCP connection has been refused by the server.  Try again later
	CHANNEL HOST UNREACHABLE	-954	---	No route to the host.  Try again later
	CHANNEL NETWORK UNREACHABLE	-953	---	No network available  Try again later
	CHANNEL PIPE BROKEN	-952	---	TCP connection interrupted  Try again later
	CHANNEL TIMEOUT	-951	---	Timeout (DNS request, TCP connection, ping response, etc.)  Try again later
	MODEM POSITION UPDATE ERROR	-943	---	Timeout during determination of the GSM position data

## 18.3 Evaluating the device log

### 18.3.1 Evaluating the device log on the myDatenet server

The last 300 log entries on the myDatenet server can be called up via the button shown below that is located in the measurement device list. As the log entries are sent to the server in the transmission cycle in the same way as the measurement data, only the log entries up to the last server connection are available.



---

The manual for the server ("myDatatnet Server Manual " 805002) includes a detailed description of the evaluation of the device log on the myDatatnet server.

### **18.3.2 Evaluating the device log using DeviceConfig**

The DeviceConfig program can be used to read all of the stored log entries, including those that have not yet been transferred to the myDatatnet server, directly from the myDataLogC33x via the USB interface.

A more detailed description about the evaluation of the device log using DeviceConfig is included in chapter ""Log" tab" on page 87.

# Chapter 19 Spare parts and accessories

## 19.1 Assembly sets

Description	Quantity	Order number
Housing for C3xx	1	301219
Wall mounting set for housing 300x200x150mm	1	301185

## 19.2 Antennas

Description	Quantity	Order number
Extension cable for antenna SMA-M/SMA-F 2,5m	1	206.807
Flat antenna Disc SMA-M 2,5m	1	206.816
Angle adapter SMA-M/SMA-F	1	300318
Rod antenna multi band 2G/3G SMA-M	1	301075
Flat antenna Disc Multi Band 2xSMA-M 2m	1	301090
Dome antenna multi band SMA-M 3m	1	301212
Combi Antenna Cellular/Wifi SMA-M 3m	1	300924

## 19.3 Power supply

Description	Quantity	Order number
Power supply 24V 0,63A for top-hat rail mounting <sup>1)</sup>	1	301066
Power supply 24V 1,5A 36W for DIN rail mounting	1	301251

<sup>1)</sup> not recommended when using extension modules

## 19.4 Adapter

Description	Quantity	Order number
Gender changer 9-pin D-Sub male/male	1	206.684
Null modem adapter 9-pin D-Sub female/male <sup>1)</sup>	1	206.686

<sup>1)</sup> Please ensure that pin 9 on the adapter is not looped through

---

## 19.5 Extension modules

Description	Quantity	Order number
Input extensions		
myDatalogC3e 12UI/2Rel	1	301058
Output extensions		
myDatalogC3e 3mA/6Rel	1	301112

## 19.6 Other accessories

Description	Quantity	Order number
myDatamet Tool Pen	1	206.646

# Chapter 20 Document history

Rev.	Date	Changes
01	21.08.2020	<b>First version</b>
02 (1/2)	25.03.2021 (1/2)	<p><b>Chapter "Declaration of Conformity" on page 11</b> <i>Declaration of Conformity updated</i></p> <p><b>Chapter "Specifications" on page 13</b> <i>Specification of the system-related overhead per data record adjusted from 10bytes to 11bytes</i> <i>Specification of the bandwidth of the WiFi channels added</i> <i>Specification of supported WiFi channels added</i></p> <p><b>Chapter "Registration memory blocks" on page 37</b> <i>The explanations of the registry entries "latestAppVersion", "installedAppVersion" and "appld", which are related to the application templates, have been removed. The application templates are no longer further developed, but are replaced by the IoT apps.</i> <i>Explanations for the registry entries "pipAppld" and "pipAppVer" have been adapted to the new use in connection with the IoT apps</i></p> <p><b>Chapter "Site" on page 74</b> <i>Added explanation of the "Application Version" field, which specifies the version number of the IoT application currently installed on the site</i></p> <p><b>Chapter "Alarm settings" on page 75</b> <i>Explanation of the "Offline Alarm After" configuration parameter added</i></p> <p><b>Chapter "DeviceConfig" on page 79</b> <i>Chapter added</i></p> <p><b>Chapter "Overview" on page 91</b> <i>Chapter added</i></p> <p><b>Chapter ""Customer" area" on page 92</b> <i>Chapter added</i></p> <p><b>Chapter ""Sites / Applications" area at customer level" on page 94</b> <i>Chapter added</i></p> <p><b>Chapter "Creating the site" on page 95</b> <i>The reference to the user manual of the server has been replaced by a detailed description of the steps to be carried out.</i></p> <p><b>Chapter "rapidM2M Studio" on page 99</b> <i>More detailed breakdown of the components of the rapidM2M Studio added</i></p> <p><b>Chapter "Project dashboard" on page 101</b> <i>Chapter added</i></p> <p><b>Chapter "CODEbed" on page 102</b> <i>Chapter added</i></p> <p><b>Chapter "TESTbed" on page 103</b> <i>Chapter added</i></p> <p><b>Chapter "Data structure" on page 207</b> <i>Chapter added</i></p>

<b>Rev.</b>	<b>Date</b>	<b>Changes</b>
02 (2/2)	25.03.2021 (2/2)	<p><b>Chapter "Example" on page 215</b> <i>Chapter added</i></p> <p><b>Chapter "Special values of the data types" on page 217</b> <i>Chapter added</i></p> <p><b>Chapter "Removal/disposal" on page 223</b> <i>Note regarding the lithium button cell included in the device and the associated effects on recycling and environmentally friendly disposal of the device added</i></p> <p><b>Chapter "Glossary" on page 245</b> <i>Explanations of the terms "App Center", "App Model", "IoT App" and "rapidM2M Store" added</i></p>
03 (1/3)	19.08.2022 (1/3)	<p><b>Chapter "Specifications" on page 13</b> <i>Note regarding the reset button added</i> <i>Specified max. size of a data record adjusted from 1024 bytes to 1023 bytes</i> <i>Specifications regarding the WiFi interface removed</i> <i>Specifications on the supported frequency bands for rapidM2M C32x 2G/M1/NB1 World, rapidM2M C32x 3G World and rapidM2M C32x 2G/4G EU added.</i> <i>Specifications regarding the rapidM2M C32x WiFi/3G EU removed</i></p> <p><b>Chapter "Overview" on page 20</b> <i>"Button" changed to "MDN Button"</i></p> <p><b>Chapter "System architecture" on page 21</b> <i>Chapter added</i></p> <p><b>Chapter "Block diagram" on page 22</b> <i>Chapter added</i></p> <p><b>Chapter "Functionality of the internal data memory" on page 31</b> <i>Chapter added</i></p> <p><b>Chapter "Memory organisation" on page 33</b> <i>Chapter added</i></p> <p><b>Chapter "Procedure in case of connection aborts" on page 34</b> <i>Chapter added</i></p> <p><b>Chapter "Timeout monitoring in online mode" on page 35</b> <i>Chapter added</i></p> <p><b>Chapter "Determining the GSM/UMTS/LTE signal strength" on page 36</b> <i>Chapter added</i></p> <p><b>Chapter "Determining the GSM position data" on page 36</b> <i>Chapter added</i></p> <p><b>Chapter "File transfer" on page 38</b> <i>Chapter added</i></p> <p><b>Chapter "Connecting the extension modules" on page 51</b> <i>Chapter added</i></p> <p><b>Chapter "Technical details about the universal inputs" on page 59</b> <i>Chapter added</i></p> <p><b>Chapter "Technical details about the RS485 interface" on page 60</b> <i>Chapter added</i></p> <p><b>Chapter "Technical details about the CAN interface" on page 61</b> <i>Chapter added</i></p>

Rev.	Date	Changes
03 (2/3)	19.08.2022 (2/3)	<p><b>Chapter "Technical details about the RS232 interface" on page 63</b> <i>Chapter added</i></p> <p><b>Chapter "Technical details about the outputs" on page 65</b> <i>Chapter added</i></p> <p><b>Chapter "Technical details about the integrated rechargeable buffer battery" on page 65</b> <i>Chapter added</i></p> <p><b>Chapter "Technical details about the energy supply" on page 67</b> <i>Chapter added</i></p> <p><b>Chapter "Technical details about the system time" on page 67</b> <i>Chapter added</i></p> <p><b>Chapter "Operating elements" on page 73</b> <i>"Button" changed to "MDN Button"</i></p> <p><b>Chapter "Basic settings" on page 76</b> <i>Explanation of the parameter for selecting the report template used to display the data has been revised (if no report template has been selected, the symbol to display the measurement data is not displayed in the list of sites/applications.)</i></p> <p><b>Chapter "Measurement instrument" on page 77</b> <i>Explanation of the "Modem Version" and "OS Version" fields that are no longer used removed</i></p> <p><b>Chapter ""Customer" area" on page 92</b> <i>Screenshots of the user interface of the myDatanet server adapted to version 50v007</i></p> <p><b>Chapter ""Sites / Applications" area at customer level" on page 94</b> <i>Screenshots of the user interface of the myDatanet server adapted to version 50v007</i></p> <p><b>Chapter "Constants" on page 106</b> <i>Explanation of the return codes "ERROR_SENSOR_DISABLED" revised</i></p> <p><b>Chapter "Uplink" on page 112</b> <i>Explanation of the arrays with symbolic indices "TrM2M_GSMInfo" extended to include the description of the "act", "lac" and "cid" elements</i> <i>Explanation of the array with symbolic indices "TrM2M_TxIlfStats" added</i> <i>Explanation of the constants for the mobile radio AcT (access technology) added</i> <i>Explanation of the constants for the signal strength measurement flags added</i> <i>Explanation of the function, that must be provided by the device logic developer and that is called up from the internal flash memory after reading a data record, extended to include the description of the parameter "timestamp256"</i> <i>Explanation of the "rM2M_TxIlfGetStats()" and "rM2M_SetTCPKeepAlive()" functions added</i> <i>Explanation of the "rM2M_GSMGetRSSI()" and "rM2M_GetRSSI()" functions extended to include the description of the "flags" parameter</i></p> <p><b>Chapter "Position" on page 139</b> <i>Explanation of the function "rM2M_PosUpdate()" added</i></p> <p><b>Chapter "Char &amp; String" on page 157</b> <i>Note added to the "strchr", "strchr", "strspn", "strcspn", "strpbrk", "strstr", "strtol" and "atof" functions, indicating that strings &gt; 128 bytes are not supported</i> <i>Explanation of the "memcpy_native", "memset_native" and "memcomp_native" functions added</i></p>

Rev.	Date	Changes
03 (3/3)	19.08.2022 (3/3)	<p><b>Chapter "Various" on page 166</b>  <i>Explanation of the "delay_us()" function added</i></p> <p><b>Chapter "File transfer" on page 176</b>  <i>Explanation of the "FT_CMD_ENUM" and "FT_CMD_RETR" file transfer commands added</i>  <i>Explanation of the callback function, that must be provided by the script developer, extended to include a description of how the file transfer commands "FT_CMD_ENUM" and "FT_CMD_RETR" should be handled</i>  <i>Explanation of the "FT_RegisterEnum()" function added</i></p> <p><b>Chapter "Device Logic error codes" on page 182</b>  <i>Explanation of the error codes "SCRIPT_ERR, SCRIPT UPDATE ERROR" revised</i>  <i>Explanation of the error codes "SCRIPT_ERR, SCRIPT SYSTEM SHUTDOWN", "SCRIPT_ERR, SCRIPT DOWNLOAD ERROR" and "SCRIPT_ERR, SCRIPT DELETED" added</i>  <i>Explanation of the error codes "LOG_NOSCRIPT_ERR, SCRIPT xxx" added</i></p> <p><b>Chapter "Data Descriptor " on page 207</b>  <i>Specified max. size of a data record adjusted from 1024 bytes to 1023 bytes</i></p> <p><b>Chapter "rapidM2M Playground " on page 219</b>  <i>Screenshot and description of the rapidM2M playground updated ("System Console" button removed, button for the global settings added)</i></p> <p><b>Chapter "Log entries and error codes" on page 227</b>  <i>Explanation of the error codes "GSM NETWORK REGISTRATION", "GPRS NETWORK REGISTRATION", "LTE NETWORK REGISTRATION", "SHT2X SENSOR OK", "SHT2X RH ERROR", "SHT2X TEMP ERROR", "SHT2X RH+TEMP ERROR", "SHT2X PLAUSIBILITY ERROR", "SHT2X COM ERR", "SHT2X COM ERR1", "SHT2X COM ERR2", "SHT2X TEMP RAW" and "SHT2X RH RAW" added</i></p> <p><b>Chapter "Assembly sets" on page 235</b>  <i>Chapter added</i></p> <p><b>Chapter "Antennas" on page 235</b>  <i>List of accessories revised</i></p> <p><b>Chapter "Power supply" on page 235</b>  <i>List of accessories revised</i></p> <p><b>Chapter "Adapter" on page 235</b>  <i>Chapter added</i></p> <p><b>Chapter "Extension modules" on page 236</b>  <i>Chapter added</i></p> <p><b>Chapter "Glossary" on page 245</b>  <i>Explanation of the terms "Device logic", "Hardware ID string", "Product revision" and "rapidM2M timestamp" added</i></p>



Rev.	Date	Changes
04	26.09.2022	<p><b>Chapter "Declaration of Conformity" on page 11</b>  <i>Declaration of conformity of the variant myDatalogC33x WIFI/2G/3G/4G World added</i></p> <p><b>Chapter "Specifications" on page 13</b>  <i>Specifications on the supported frequency bands for myDatalogC33x WIFI/2G/3G/4G World added.</i></p> <p><b>Chapter "Overview" on page 20</b>  <i>Image of the front of the device updated (VIN and GND were interchanged)</i></p> <p><b>Chapter "Block diagram" on page 22</b>  <i>Load resistances of the CAN interface added to the block diagram</i></p> <p><b>Chapter "Device labelling" on page 25</b>  <i>Type plate updated</i></p> <p><b>Chapter "Scope of supply" on page 41</b>  <i>Selection of the available variants updated</i></p> <p><b>Chapter "Dimensions" on page 43</b>  <i>Image updated (VIN and GND were interchanged)</i></p> <p><b>Chapter "Connecting the sensors, actuators and power supply" on page 46</b>  <i>Image updated (VIN and GND were interchanged)</i>  <i>Note added indicating that a parallel connection of a PhotoMOS relay (high switching frequencies) and a mechanical relay (high switching currents) is in place for the outputs</i></p> <p><b>Chapter "Connection examples" on page 49</b>  <i>Image updated (VIN and GND were interchanged)</i></p> <p><b>Chapter "Connection of the GSM antenna" on page 50</b>  <i>Reference to the antennas to be used as an alternative in the event of a low radio signal strength changed to the Dome antenna multi band SMA-M 3m(301211).</i></p> <p><b>Chapter "Technical details about the CAN interface" on page 61</b>  <i>Error in the table that indicates which switches are used to switch the load resistances has been fixed. With S1 and S2, the 2k load resistance is activated rather than the 120Ω load resistance.</i></p> <p><b>Chapter "Operating elements" on page 73</b>  <i>Image updated (VIN and GND were interchanged)</i></p> <p><b>Chapter "Assembly sets" on page 235</b>  <i>Wall mounting set for housing 300x200x150mm (301185) added</i></p>
05	17.10.2022	<p><b>Chapter "Connection of the GSM antenna" on page 50</b>  <i>Order number of theDome antenna multi band SMA-M 3mcorrected from (301211) to (301212)</i></p>

Rev.	Date	Changes
06 (1/2)	22.09.2023 (1/2)	<p><b>Chapter "Declaration of Conformity" on page 11</b>  <i>Variant rapidM2M C33x WIFI/3G World removed</i>  <i>Variant rapidM2M C33x WIFI/2G/4G EU removed</i></p> <p><b>Chapter "Specifications" on page 13</b>  <i>Variant rapidM2M C33x WIFI/3G World removed</i>  <i>Variant rapidM2M C33x WIFI/2G/4G EU removed</i></p> <p><b>Chapter "Scope of supply" on page 41</b>  <i>Variant rapidM2M C33x WIFI/3G World removed</i>  <i>Variant rapidM2M C33x WIFI/2G/4G EU removed</i></p> <p><b>Chapter "Installing the myDatalogC33x " on page 43</b>  <i>Note added indicating that the device is not certified for operation in closed sewage systems.</i></p> <p><b>Chapter "Constants" on page 106</b>  <i>Returncode "ERROR_SENSOR_DISABLED" added</i></p> <p><b>Chapter "Timer, date &amp; time" on page 107</b>  <i>Explanation of the array with symbolic indices "TrM2M_DateTime" extended</i>  <i>Explanations of the "rM2M_TimerAdd()" and "rM2M_TimerAddExt()" functions extended</i></p> <p><b>Chapter "Uplink" on page 112</b>  <i>Explanation of the "rM2M_CfgRead()" function extended</i>  <i>Explanation of the return value of the "rM2M_CfgWrite()" function corrected</i>  <i>List of functions that can be returned by the "rM2M_TxGetStatus()" function extended to include the error codes "RM2M_TXERR_MODEM_RESETLOOP", "RM2M_TXERR_MODEM_UNDERVOLTAGE" and "RM2M_TXERR_MODEM_OVERHEAT".</i></p> <p><b>Chapter "Encoding" on page 128</b>  <i>Explanation of the "rM2M_SetPacked()" function corrected, now indicating that in connection with signed data types problems can arise and not in connection with the formerly indicated unsigned data types.</i>  <i>Explanation of the "rM2M_Pack()" function extended</i></p> <p><b>Chapter "Registry" on page 134</b>  <i>Explanation of the constants for the "Indices of the registration memory blocks" extended</i>  <i>Explanations of the "rM2M_RegGetString()", "rM2M_RegGetValue()", "rM2M_RegSetString()", "rM2M_RegSetValue()" and "rM2M_RegOnChg()" functions extended</i></p> <p><b>Chapter "64 bit signed integer" on page 154</b>  <i>Chapter added</i></p>

Rev.	Date	Changes
06 (2/2)	22.09.2023 (2/2)	<p><b>Chapter "Char &amp; String" on page 157</b>  <i>Explanations of the "sprintf()", "strcat()", "strcmp()", "strcspn()", "strpbrk()", "strtol()" and "atof()" functions extended</i>  <i>Note indicating that strings &gt; 128 bytes are not supported removed from the explanations of the "strchr()", "strchr()", "strspn()", "strcspn()", "strpbrk()" and "strtsr()" functions</i></p> <p><b>Chapter "CRC &amp; hash" on page 165</b>  <i>Exrxplanation of the "MD5()" function extended</i></p> <p><b>Chapter "Various" on page 166</b>  <i>Explanations of the "getapilevel()", "exists()", "rtm_start()", "funcidx()", "numargs()" and "getarg()" functions extended</i></p> <p><b>Chapter "Console" on page 173</b>  <i>Explanation of the "printf()" function extended</i>  <i>Note added to the "setbuf()" function, indicating that the buffer "buf" has to be valid throughout the use by the firmware.</i></p> <p><b>Chapter "Differences to C" on page 204</b>  <i>Note added indicating that variables are automatically initialised with "0"</i></p> <p><b>Chapter "Data structure" on page 207</b>  <i>Explanation of the attribute "editmask" revised</i></p> <p><b>Chapter "Log entries and error codes" on page 227</b>  <i>Explanation of error code "MODEM NOT FOUND" added</i>  <i>Explanation of error code "ACCU 0 E2PROM ERROR" added</i></p>



---

# Chapter 21 Glossary

## App centre

Area of the myDatanet server for the installation and management of the IoT apps. The app models that serve as a basis for the IoT apps are obtained via the rapidM2M Store . When installing an IoT app on the myDatanet server the default settings defined when developing the app models are initially applied. These default settings can then be adjusted. Any number of IoT apps can be created based on a single app model by setting the appropriate default settings.

## App model

An app model is developed in the rapidM2M Studio and forms the basis for creating IoT apps. It essentially contains the executable program files (device logic, backend logic, portal view, etc.) from which an IoT is created by adding the default settings. Distribution to the individual myDatanet servers is carried out via the rapidM2M Store . The available app models are displayed in the app centre of the respective myDatanet server.

## Footprint

The manufacturer's devices are equipped with subscriber identity modules (SIM) ex-works for the purpose of mobile data transmission. The footprint describes those countries and regions where a mobile connection is available (see [www.microtronics.com/footprint](http://www.microtronics.com/footprint)).

## Device logic

The device logic is the intelligence installed on the device that determines the local functionality of the device. The device logic is part of the app model and is created in a C-like scripting language built on "PAWN".

## Hardware ID string

Specifies the hardware platform installed in the device and its hardware version (e.g. rapidM2M M2 HW1.4). The part of the hardware ID string, that specifies the hardware version, is only increased if changes relevant to the rapidM2M firmware have been made to the hardware platform. When developing an app model, it can be specified on which hardware platform the app model can be installed and which version of the hardware platform is required as a minimum. The hardware ID string is displayed in the TESTbed of the rapidM2M Studio or in the "Identification" field of the input screen for configuring the device.

## IoT app

IoT apps form the basis for creating sites. They consist of an app model and corresponding default settings that are applied as default values for the site when the site is created. The app centre can be used to create any number of IoT apps based on a single app model by setting the appropriate default settings. This makes sense if several use cases need to be covered by a single app model and they each require a different default site configuration (e.g. if a data logger with different external sensors is to be sold as a package).

## NaN value

The myDatanet uses special encoding to display different error statuses in the measurement values, for example. By setting a measurement value to "NaN", it is clearly marked as invalid and is thus not used for any further calculations. In the measurement value graphs, a measurement value that has been set to "NaN" is indicated by an interruption in the graph. When downloading the data, a measurement value set to "NaN" is indicated by an empty data field.

## Product revision

Specifies the revision of the product. The revision is increased every time the product is modified (i.e. electronic system, mechanics, etc.) and is marked on the type plate of the product.

---

**rapidM2M Store**

*Is responsible for distributing the app models to the individual myDataneT servers. When installing and updating IoT apps the myDataneT server access the app models provided in the rapidM2M Store . The developer of the respective app model defines which myDataneT servers are allowed to access an app model via the rapidM2M Studio .*

**rapidM2M timestamp**

*Depending on the required accuracy, one of two special encodings can be used for the time stamp in rapidM2M. If the accuracy requirements are moderate, the "stamp32" data type (seconds since 1999-12-31 00:00:00 UTC) can be used. If a higher accuracy is required, the "stamp40" data type (1/256 seconds since 1999-12-31 00:00:00 UTC) can be used. Converting the "stamp32" data type into the UNIX timestamp (seconds since 1970-01-01 00:00:00 UTC) can be achieved by adding 946598400.*

## Chapter 22 Contact information

### **Support & Service:**

Microtronics Engineering GmbH  
Hauptstrasse 7  
3244 Ruprechtshofen  
Austria, Europe  
Tel. +43 (0)2756 7718023  
support@microtronics.com  
www.microtronics.com

### **Microtronics Engineering GmbH (Headquarters)**

Hauptstrasse 7  
3244 Ruprechtshofen  
Austria, Europe  
Tel. +43 (0)2756 77180  
Fax. +43 (0)2756 7718033  
office@microtronics.com  
www.microtronics.com



Certified by TÜV AUSTRIA: EN ISO 9001:2015, EN ISO 14001:2015, ISO/IEC 27001:2013, EN ISO 50001:2011 for myDatenet | TÜV SÜD: ATEX Directive 2014/34/EU

© Microtronics Engineering GmbH. All rights reserved. Photos: Microtronics, shutterstock.com

**Microtronics Engineering GmbH | [www.microtronics.com](http://www.microtronics.com)**

Hauptstrasse 7 | 3244 Ruprechtshofen | Austria | +43 2756 77180 | [office@microtronics.com](mailto:office@microtronics.com)

301085 | Rev.06